

ネットワークコンピューティング 第1回（後半）

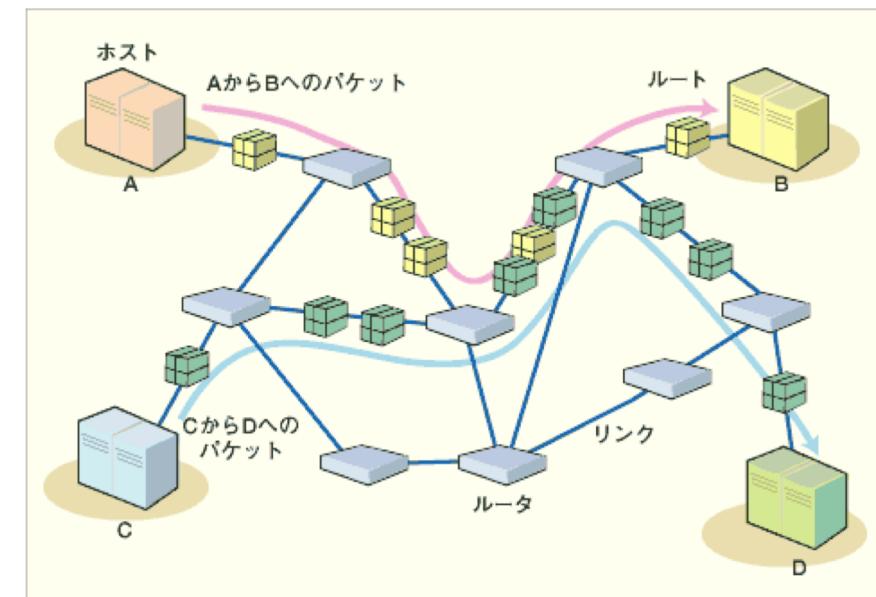
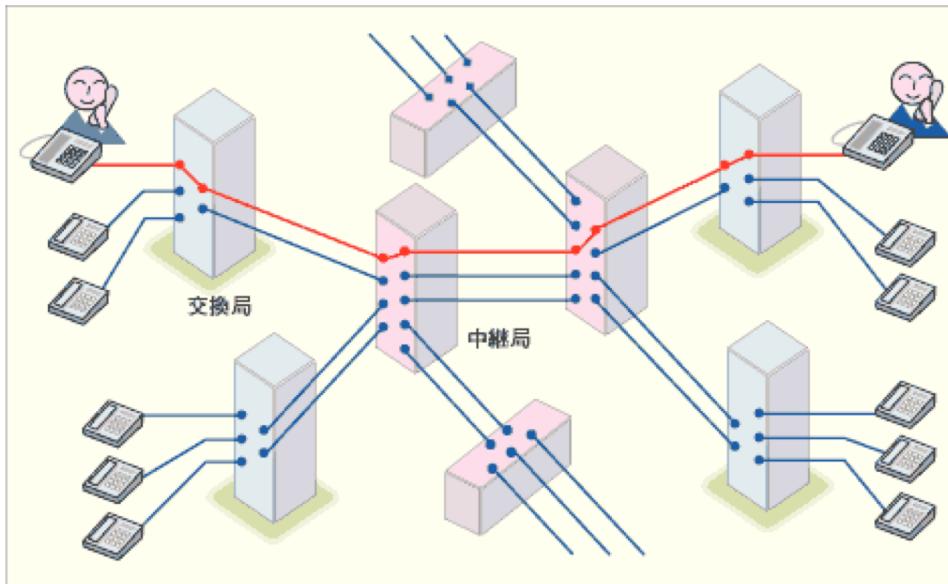
中山 雅哉 (m.nakayama@m.cnl.t.u-tokyo.ac.jp)
関谷 勇司 (sekiya@nc.u-tokyo.ac.jp)

OSI 参照モデル

- OSI : Open Systems Interconnection
- ISO (国際標準化機構) によって策定された、コンピュータの通信を階層的に表したモデル
- その昔 (1970年代) まだインターネットも TCP/IP も無かった時代
 - いろいろな通信プロトコルが考え始められた
 - プロトコル = 手順 / ルール
 - SNA (IBM)、FNA (富士通)、HNA (日立)、DCNA (NTT)
 - これらをまとめるために標準化機構が策定したのが OSI
 - 結果として TCP/IP が生き残った
 - OSI は利害関係のため策定が遅れた

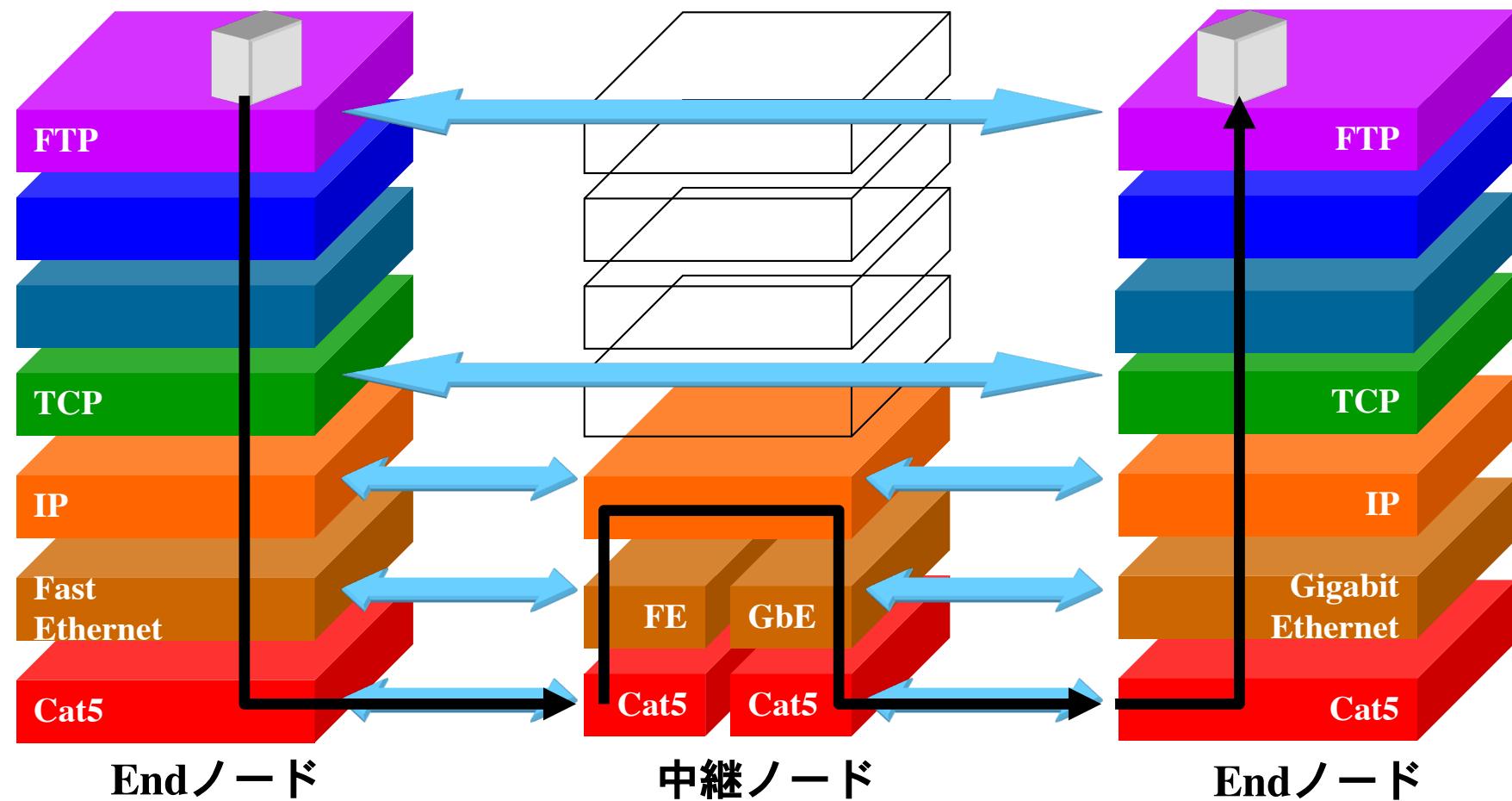
回線交換 v.s. パケット交換

- 回線を共有して使用
- 段階的に集約ポイントが存在する
- パケット単位でデータを交換するノンブロッキング通信
 - 衝突は発生する
 - 経路は動的に変更される



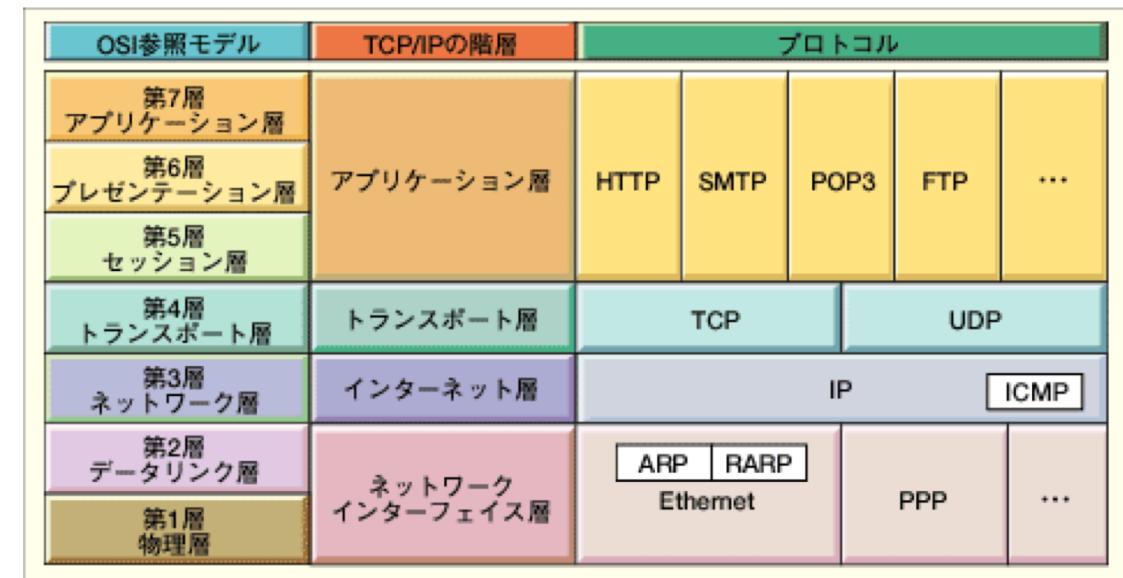
出典 : <http://www.atmarkit.co.jp/fwin2k/network/tcpip001/tcpip04.html>

TCP/IPによるデータ伝送の流れ



階層構造アーキテクチャ

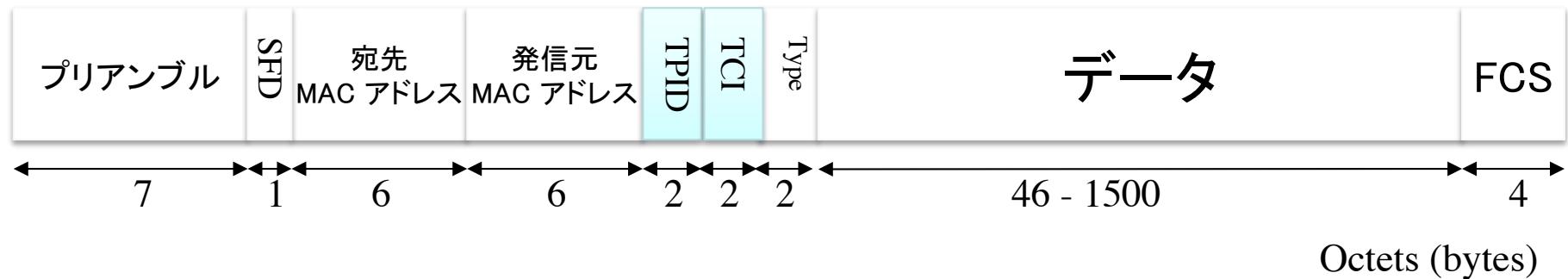
- 階層構造の利点
 - それぞれの層にて役割を分割することが可能
 - 分割した役割毎に、部品を入れ替えることが可能
- 物理層
 - 有線 LAN / 無線 LAN / ADSL / 光ファイバー
- ネットワーク層
 - IPv4 / IPv6 / AppleTalk



Ethernet (イーサネット)

- XeroxのPARC (Palo Alto Research Center)によって発明
 - 1980年にリリース
 - Xerox, Intel, DECによってEthernet 2.0を制定 (1982年)
 - それぞれの頭文字をとって DIX Ethernet と呼ばれる
 - IEEE802.3 CSMA/CD として標準化 (1983年)
- フレーム交換による通信
 - Ethernetフレームによる通信
- 伝送帯域 : 10Mbps ~ 100Gbps
- もっとも一般的な LAN (Local Area Network)構築メディア

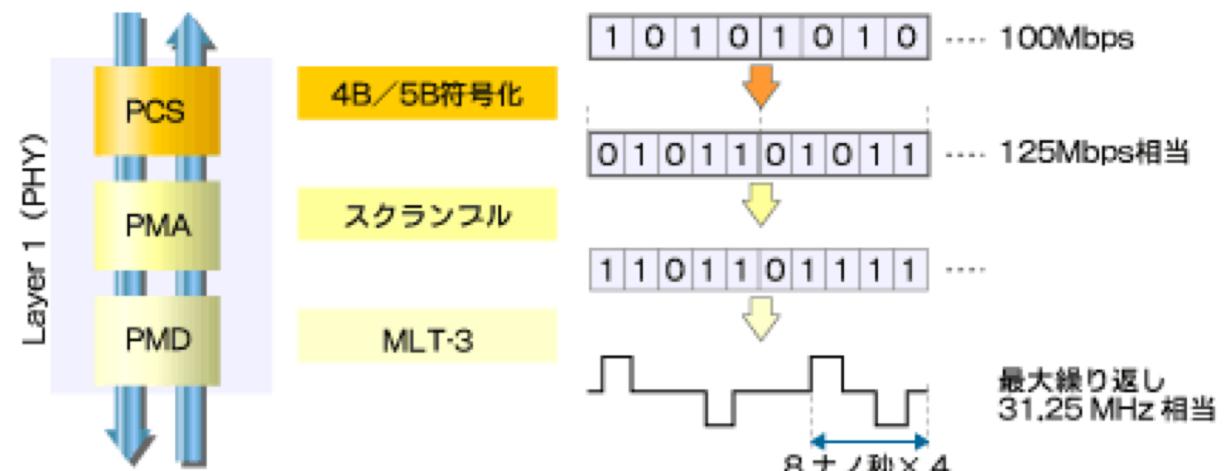
Ethernet Frame Format (DIX Ethernet)



- フレーム内には、宛先アドレスと発信元アドレス (MACアドレス)が含まれる
 - SFD (Start Frame Delimiter)
 - FCS (Frame Check Sequence)
- プロトコルタイプフィールドで上位層を識別

符号化 (100base-TX)

- 10base-T ~ 100base-TX
 - 4B/5B 符号化の後
MLT-3 として送出
- 4B/5B
 - 4bit を 5bit で表す
 - 少なくとも5bit に
ひとつは 1が入る
- スクランブル
 - 電磁妨害対策
- MLT-3
 - +1, 0, -1



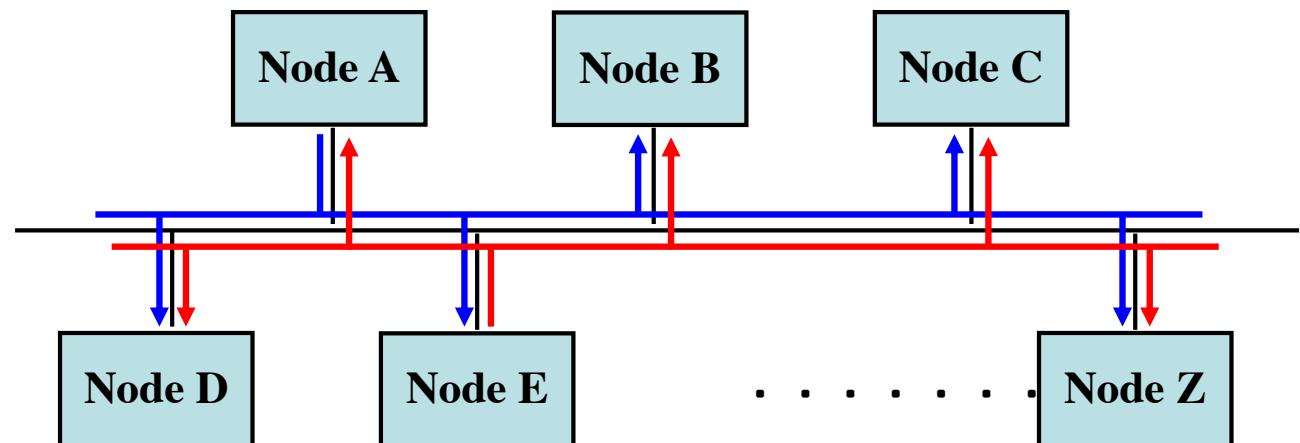
出典 : <http://www.atmarkit.co.jp/fnetwork/rensei/eth02/01.html>

符号化 (1000base-T)

- 8B/1Q4(8 binary to 1 Quinary 4)
 - 8bit → 5値4組の符号に変換
 - 2進数8桁の符号を5進数4桁に変換
 - -2, -1, 0, 1, 2 の5値
- 4D-PAM5 にて送出(4 dimensional-phase amplitude modulation 5)
 - -2, -1, 0, 1, 2 の5値を使用
 - 2本の銅線を1対として、4対の信号線を使用

Broadcast Multiple Access

- 複数のノードが同一リンクに接続する
 - フレームはリンク上の全ノードにブロードキャストされる
 - ノードは自分宛のパケットのみ受信
 - リンク上のノードに識別にMACアドレスを必要とする

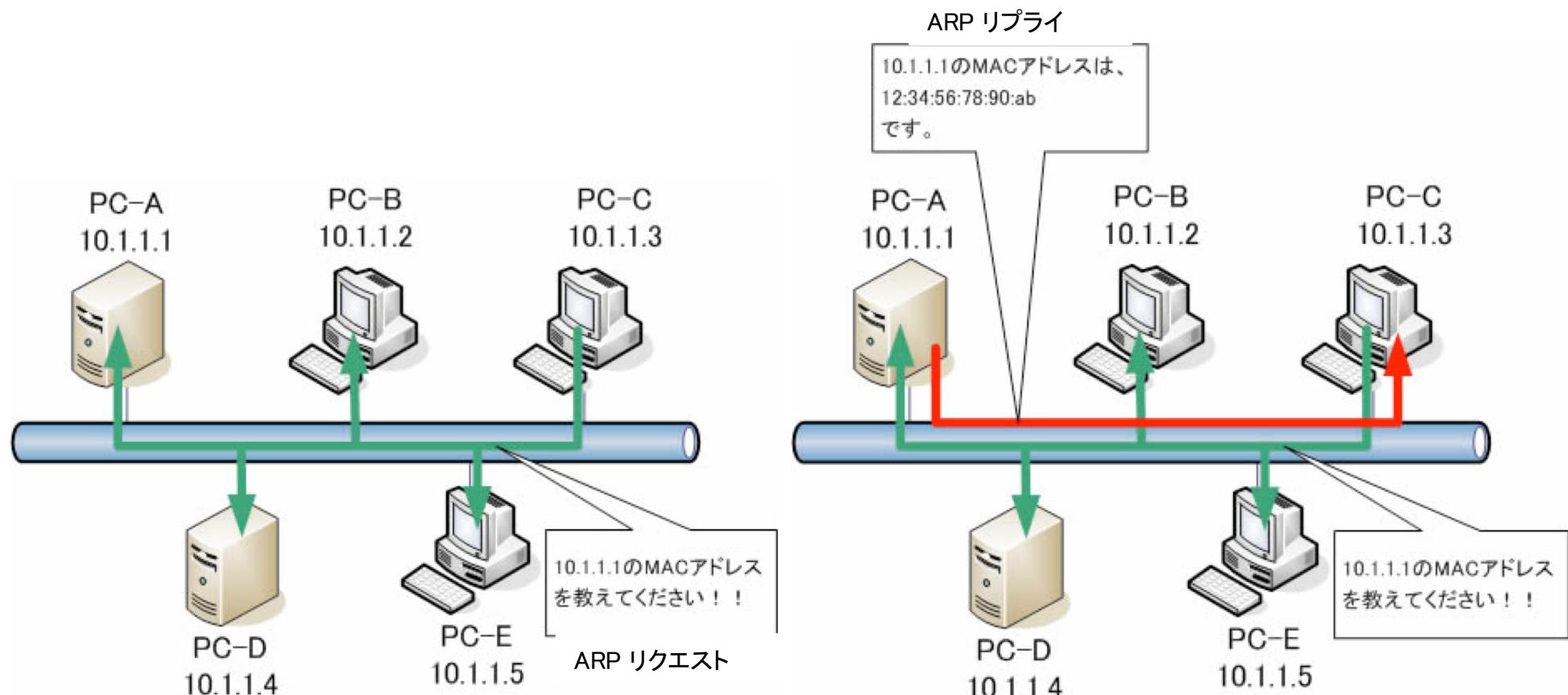


ARP

- Address Resolution Protocol
 - 下位層アドレスと上位層アドレスを関連づける
 - MAC address と IP address の対応付け

```
en0: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
inet6 fe80::217:f2ff:fe0e:d6d0%en0 prefixlen 64 scopeid 0x5
    inet 130.69.251.130 netmask 0xffffffff broadcast 130.69.251.255
        inet6 2001:200:180:299:217:f2ff:fe0e:d6d0 prefixlen 64 autoconf
            ether 00:17:f2:0e:d6:d0
            media: autoselect (1000baseT <full-duplex,flow-control>) status: active
```

ARP の動作概要



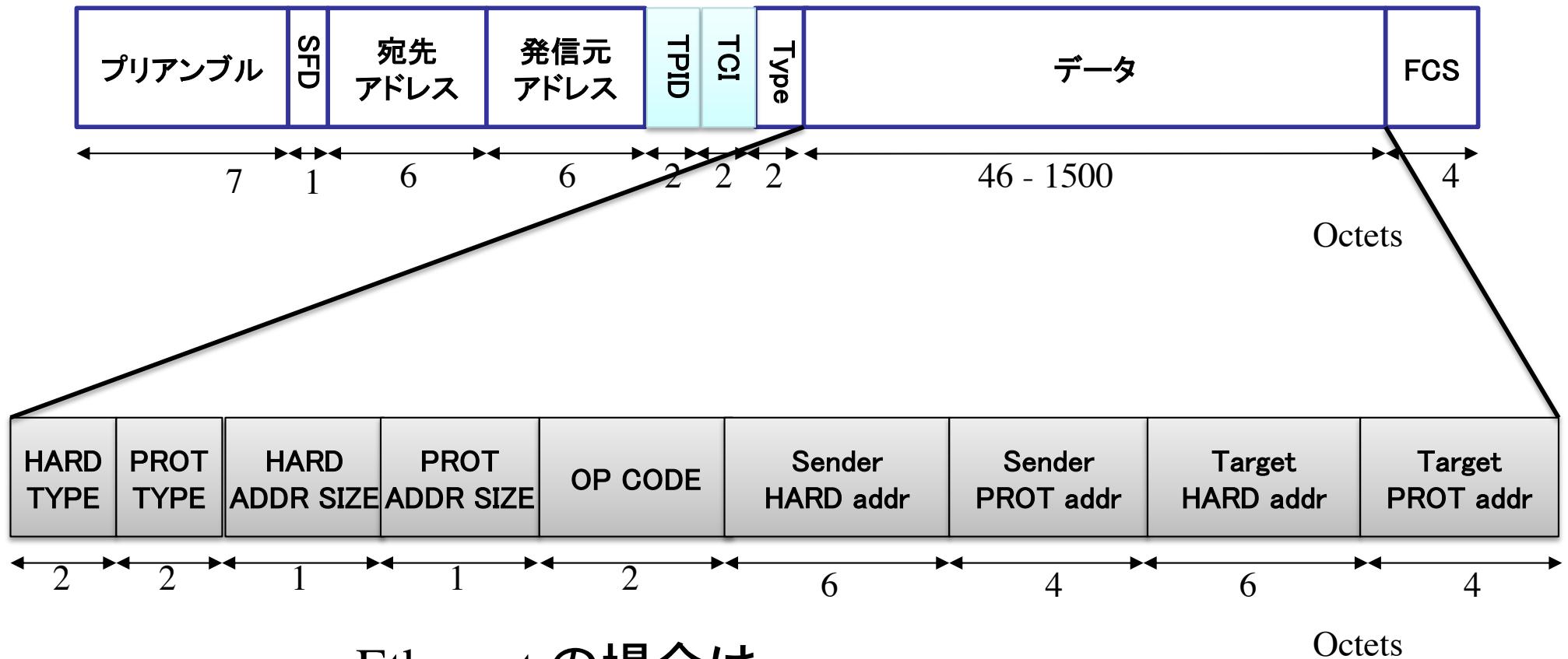
全員が認識するよう

宛先 MAC address = 00:00:00:00:00:00

送信元 MAC address = 自分の MAC address

出典 : <http://www.itbook.info/study/arp2.html>

ARP フォーマット



Ethernet の場合は

HARD addr = MAC addr

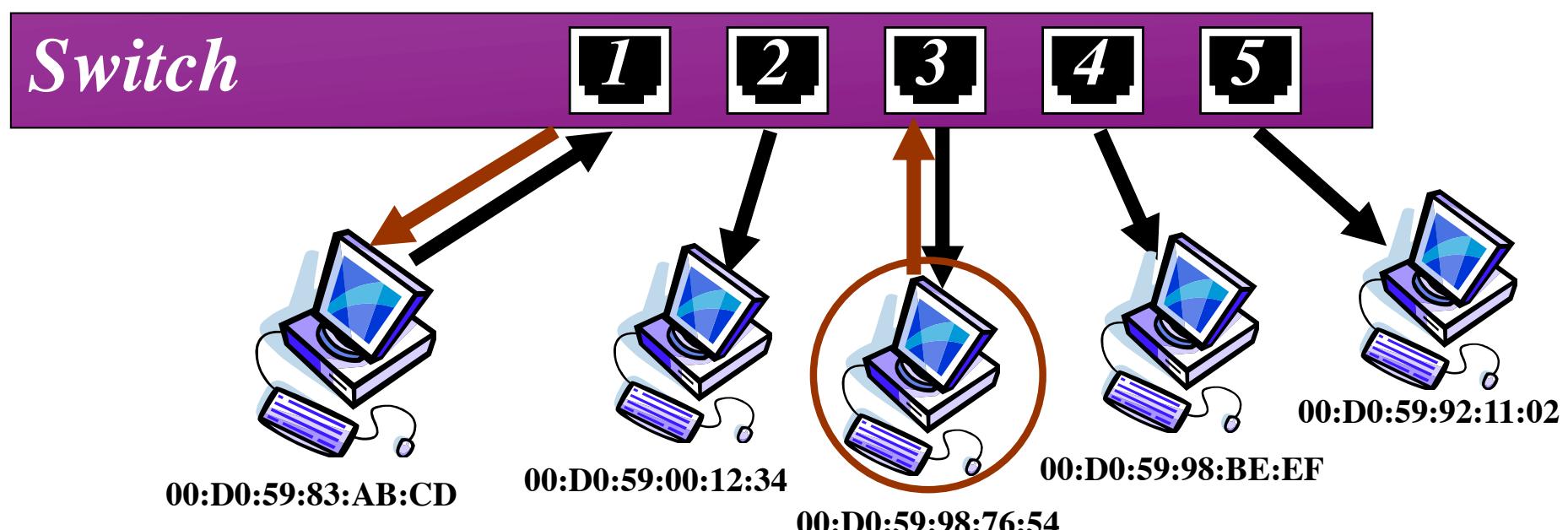
PROT addr = IP addr

ブリッジ(スイッチ)の動作

- フレームの発信元MACアドレスと、受信ポートを記憶
- フレーム転送時の処理
 - キャッシュになければ、受信ポート外の全ポートに転送
 - 宛先アドレスが存在するポートがキャッシュされていれば、そこにフレームを転送
- 一定時間マッチするトラフィックがなければ、キャッシュエントリを削除

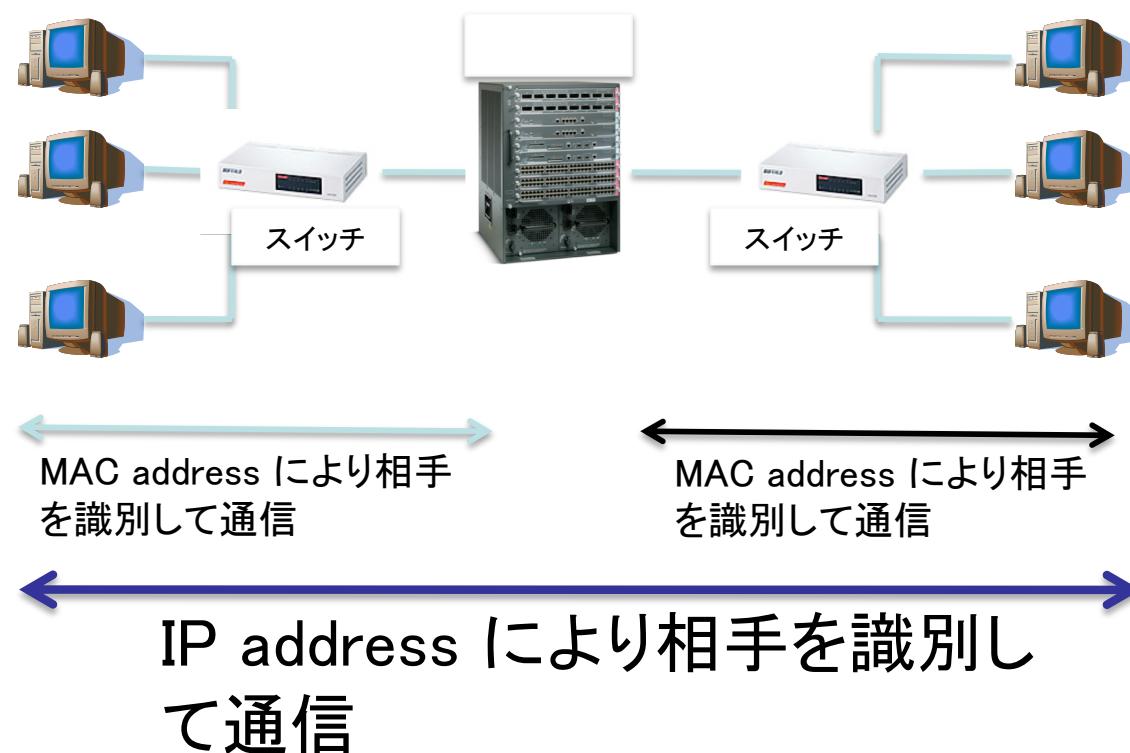
Port	MACアドレス
1	00:D0:59:83:AB:CD

3	00:D0:59:98:76:54
---	-------------------



MAC address / IP address

- 通信先は常に IP address によって指定される
- ARP/NDP によって IP address → MAC address 変換が行われる



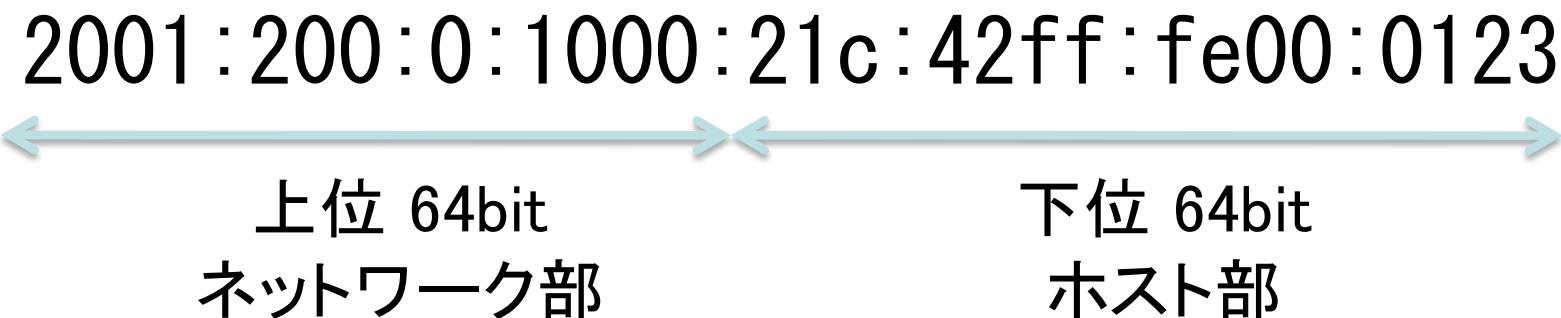
IPv4アドレスの構造

- ネットワーク部とホスト部
 - サブネット上のホストには、同じネットワーク部を付ける。
 - ネットワーク部の「長さ」を **サブネットマスク** で示す。
 - 下の例では25ビット分
- 130.69.251.130/25



IPv6アドレスの構造

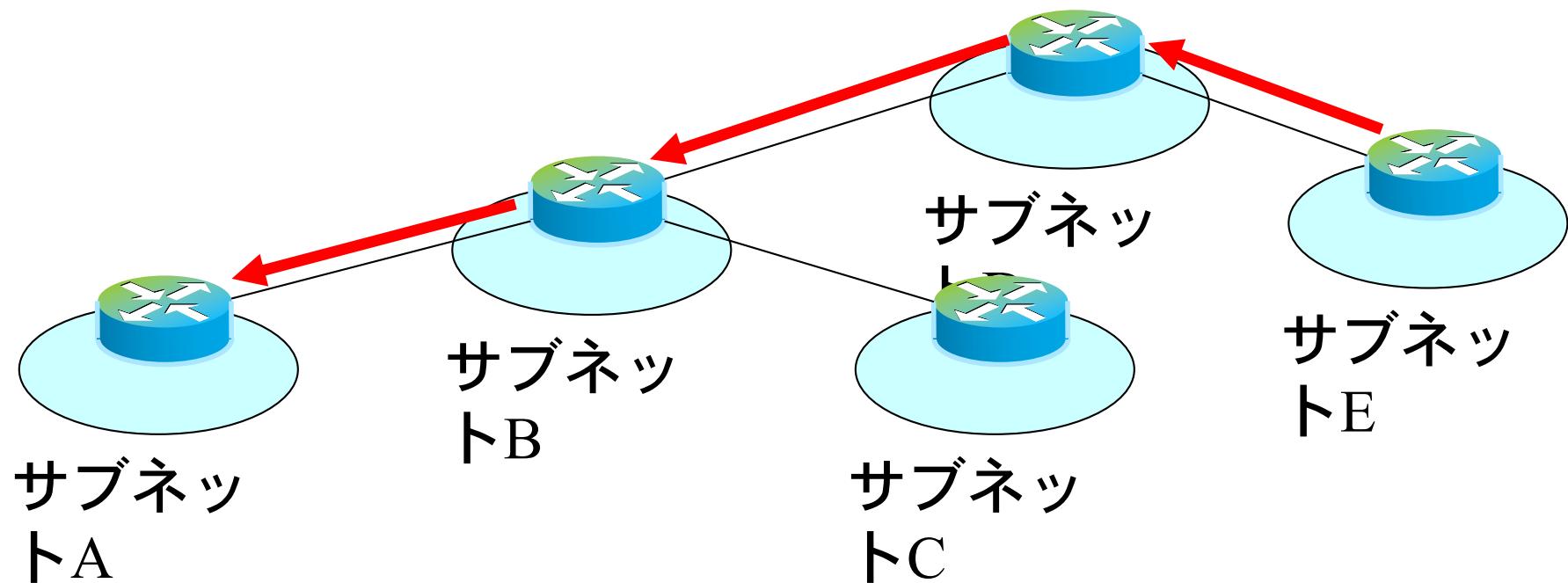
- 概念は IPv4 と同じ
- IPv4 との違い
 - アドレス長が長い (128bit)
 - ネットワーク部とホスト部が固定
 - いつも /64
 - ホスト部のアドレス空間は 2^{64} 個



IP パケットの転送 (フォワーディング)

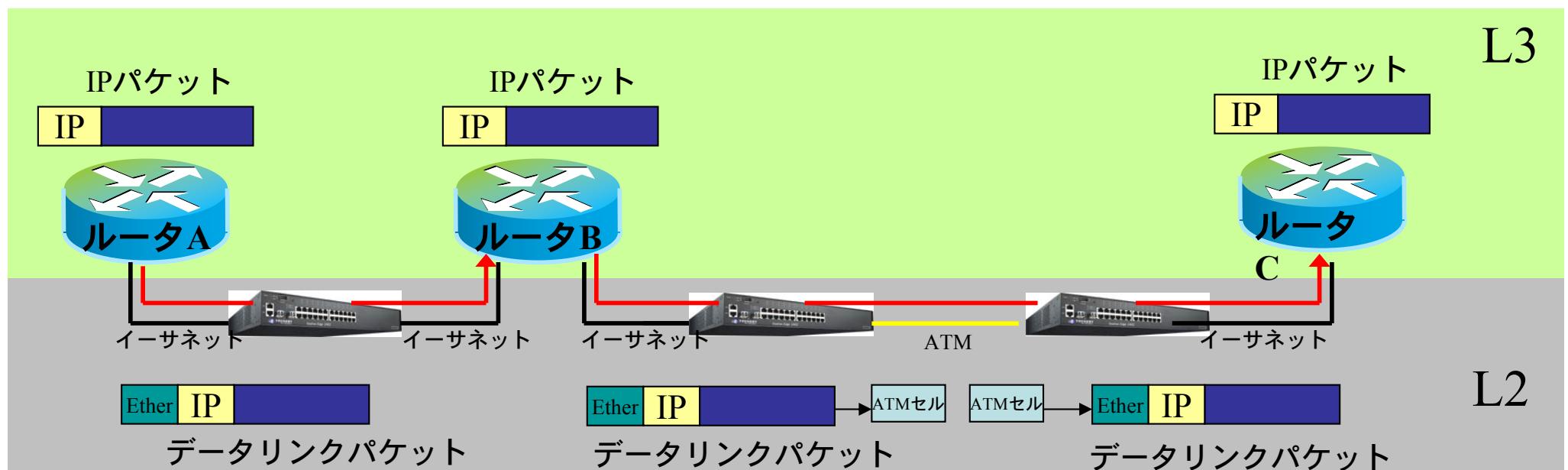
- ・ インターネットは無数のサブネットを相互接続
- ・ あるサブネットから別のサブネットに到達するためには複数のルータが中継

➡ IP フォワーディング



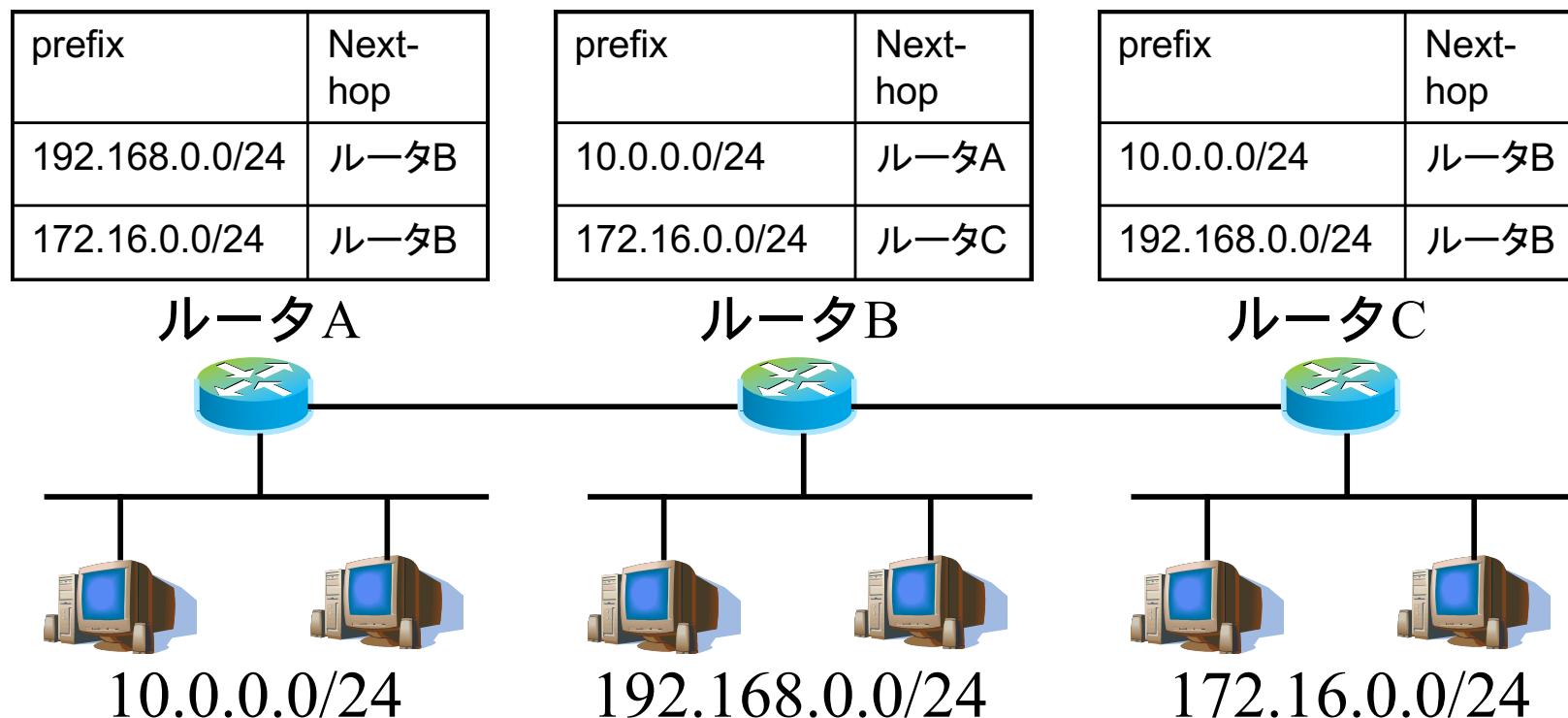
IPによるパケットの伝送

- 複数のデータリンクを相互に接続
 - 大規模なネットワークを構築
 - ルータによって各ネットワーク(セグメント)間を接続
- IPはデータリンクのトポロジを意識しない
 - データリンクは1本の通信回線に見える



経路表

- 全てのインターネットノードは経路表を持つ
 - ホストは default route (ルータの IP アドレス) だけを知っている
 - ルータは多くの経路を保持

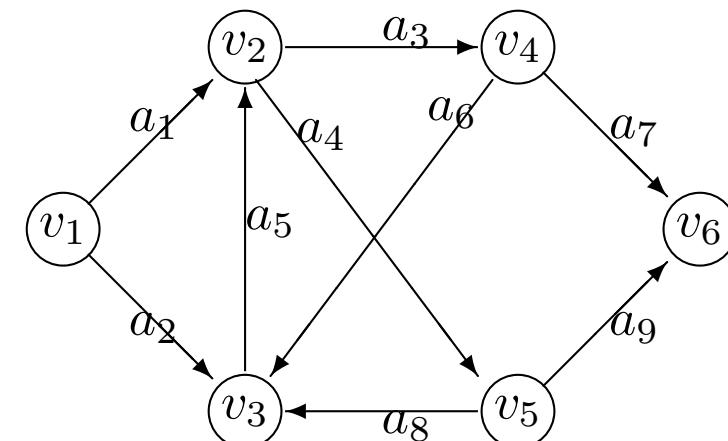


経路制御プロトコル

- 距離ベクトル型 (Distance Vector Model)
 - RIP
- リンク状態型 (Link State Model)
 - OSPF
 - IS-IS
- パスベクトル型 (Path Vector Model)
 - BGP

グラフ理論

- グラフとは
 - 邊 (path) と頂点をもつ
 - 路 (経路) : ある頂点からある頂点までの道筋
- 向きを有するか
 - 有向グラフ
 - 無向グラフ
- 最適化問題例
 - 最短経路問題 : 最短な経路
 - 最小木問題 : 無向グラフにおける最短経路
 - 巡回セールスマン問題 : 全ての頂点を通り、コストが最小となる経路
 - 最大流問題 : 入口から出口に流れる量を最大にする
 - 最小費用流問題 : 単位フローあたりの費用を考慮した流量



距離ベクトル型の特徴

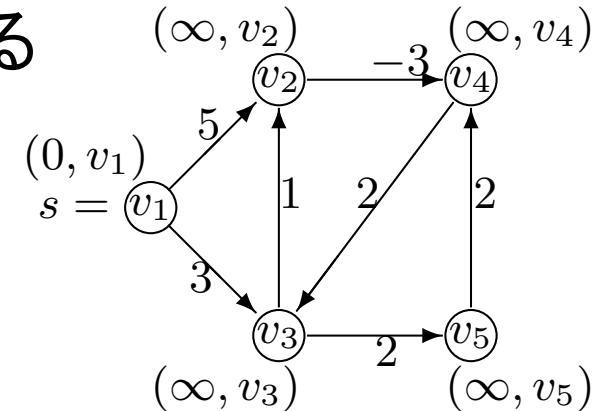
- 代表的なプロトコル: RIPv1, RIPv2
- プロトコルの動作が簡単
 - プログラムも簡単
 - 小型ルータ/エントリクラスのL3スイッチでも搭載
- 経路の収束にかかる時間が長い
 - 小規模なネットワークなら高速
 - 収束
 - 全てのルータが正しい経路表が作成し、それ以上更新しなくなるまでの時間
- ループが発生する可能性
 - 確率は低いが、完全にループフリーではない
- スケーラビリティ
 - 大規模なネットワークほど、交換する経路情報が増す

距離ベクトル型のアルゴリズム

- ベルマン-フォード法 (Bellman-Ford Algorithm)
 - グラフ理論による最短経路解決アルゴリズム
 - この分散型 (各ノードが自律的に行う) を用いて距離ベクトル型ルーティングプロトコルを実現
- 各ルータは、経路情報を隣接ルータに広告
 - 自分が到達可能なプレフィックス
 - そこへの距離(例えばホップ数)
- 経路情報を受け取った場合の処理
 - 自分が知らない経路なら、採用
 - 自分が知っている経路より、短い経路なら採用
- 新しい経路情報を隣接ルータに広告

Bellman – Ford Algorithm

- 単一始点最短経路問題の解法のひとつ
 - 負の重みをもつ辺を扱うことができる
- 計算量
 - $O(V \cdot E)$ V : 頂点数, E : 辺数



	v_1	v_2	v_3	v_4	v_5
$k = 1$	<p>Initial state: $(0, v_1)$, (∞, v_2), (∞, v_3), (∞, v_4), (∞, v_5) Edges: (v_1, v_2) weight 5, (v_1, v_3) weight 3, (v_1, v_4) weight infinity, (v_1, v_5) weight infinity (v_2, v_3) weight 1, (v_2, v_4) weight -3, (v_3, v_4) weight 2, (v_3, v_5) weight 2, (v_4, v_5) weight 2, (v_5, v_4) weight 2</p>	<p>After 1st iteration: $(0, v_1)$, $(5, v_2)$, $(3, v_3)$, (∞, v_4), (∞, v_5) Edges: (v_1, v_2) weight 5, (v_1, v_3) weight 3, (v_1, v_4) weight infinity, (v_1, v_5) weight infinity (v_2, v_3) weight 1, (v_2, v_4) weight -3, (v_3, v_4) weight 2, (v_3, v_5) weight 2, (v_4, v_5) weight 2, (v_5, v_4) weight 2</p>	<p>After 1st iteration: $(0, v_1)$, $(5, v_2)$, $(3, v_3)$, $(4, v_4)$, (∞, v_5) Edges: (v_1, v_2) weight 5, (v_1, v_3) weight 3, (v_1, v_4) weight infinity, (v_1, v_5) weight infinity (v_2, v_3) weight 1, (v_2, v_4) weight -3, (v_3, v_4) weight 2, (v_3, v_5) weight 2, (v_4, v_5) weight 2, (v_5, v_4) weight 2</p>	<p>After 1st iteration: $(0, v_1)$, $(5, v_2)$, $(3, v_3)$, $(4, v_4)$, $(5, v_5)$ Edges: (v_1, v_2) weight 5, (v_1, v_3) weight 3, (v_1, v_4) weight infinity, (v_1, v_5) weight infinity (v_2, v_3) weight 1, (v_2, v_4) weight -3, (v_3, v_4) weight 2, (v_3, v_5) weight 2, (v_4, v_5) weight 2, (v_5, v_4) weight 2</p>	<p>After 1st iteration: $(0, v_1)$, $(5, v_2)$, $(3, v_3)$, $(4, v_4)$, $(5, v_5)$ Edges: (v_1, v_2) weight 5, (v_1, v_3) weight 3, (v_1, v_4) weight infinity, (v_1, v_5) weight infinity (v_2, v_3) weight 1, (v_2, v_4) weight -3, (v_3, v_4) weight 2, (v_3, v_5) weight 2, (v_4, v_5) weight 2, (v_5, v_4) weight 2</p>
$k = 2$	<p>After 2nd iteration: $(4, v_3)$, $(2, v_2)$, $(3, v_1)$, (∞, v_4), $(5, v_3)$ Edges: (v_1, v_2) weight 5, (v_1, v_3) weight 3, (v_1, v_4) weight infinity, (v_1, v_5) weight infinity (v_2, v_3) weight 1, (v_2, v_4) weight -3, (v_3, v_4) weight 2, (v_3, v_5) weight 2, (v_4, v_5) weight 2, (v_5, v_4) weight 2</p>	<p>After 2nd iteration: $(4, v_3)$, $(1, v_2)$, $(3, v_1)$, (∞, v_4), $(5, v_3)$ Edges: (v_1, v_2) weight 5, (v_1, v_3) weight 3, (v_1, v_4) weight infinity, (v_1, v_5) weight infinity (v_2, v_3) weight 1, (v_2, v_4) weight -3, (v_3, v_4) weight 2, (v_3, v_5) weight 2, (v_4, v_5) weight 2, (v_5, v_4) weight 2</p>	<p>After 2nd iteration: $(4, v_3)$, $(1, v_2)$, $(5, v_1)$, (∞, v_4), $(5, v_3)$ Edges: (v_1, v_2) weight 5, (v_1, v_3) weight 3, (v_1, v_4) weight infinity, (v_1, v_5) weight infinity (v_2, v_3) weight 1, (v_2, v_4) weight -3, (v_3, v_4) weight 2, (v_3, v_5) weight 2, (v_4, v_5) weight 2, (v_5, v_4) weight 2</p>	<p>After 2nd iteration: $(4, v_3)$, $(1, v_2)$, $(5, v_1)$, $(1, v_4)$, $(5, v_3)$ Edges: (v_1, v_2) weight 5, (v_1, v_3) weight 3, (v_1, v_4) weight infinity, (v_1, v_5) weight infinity (v_2, v_3) weight 1, (v_2, v_4) weight -3, (v_3, v_4) weight 2, (v_3, v_5) weight 2, (v_4, v_5) weight 2, (v_5, v_4) weight 2</p>	<p>After 2nd iteration: $(4, v_3)$, $(1, v_2)$, $(5, v_1)$, $(1, v_4)$, $(5, v_3)$ Edges: (v_1, v_2) weight 5, (v_1, v_3) weight 3, (v_1, v_4) weight infinity, (v_1, v_5) weight infinity (v_2, v_3) weight 1, (v_2, v_4) weight -3, (v_3, v_4) weight 2, (v_3, v_5) weight 2, (v_4, v_5) weight 2, (v_5, v_4) weight 2</p>

リンク状態型の特徴

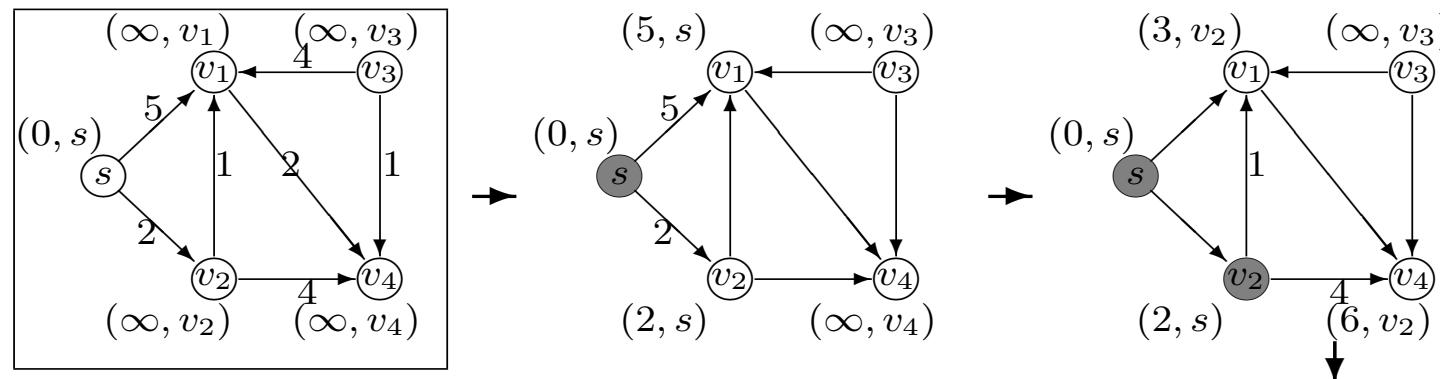
- 代表的なリンク状態型経路制御プロトコル
 - OSPF (Open Shortest Path First)
 - IS-IS (Integrated System to Integrated System)
- 大規模なネットワークに適している
 - Loop Free
 - いったん経路が収束した後は、経路制御メッセージによるトラフィック量が少ない
 - ネットワークの規模が大きくなっても、恒常に流れる経路制御メッセージの増加量は少ない
- 距離ベクトル型に比べ、処理が複雑
 - 安価なルータ, L3スイッチには搭載されていない
- 大規模なネットワークでも経路の収束が早い

リンク状態型のアルゴリズム

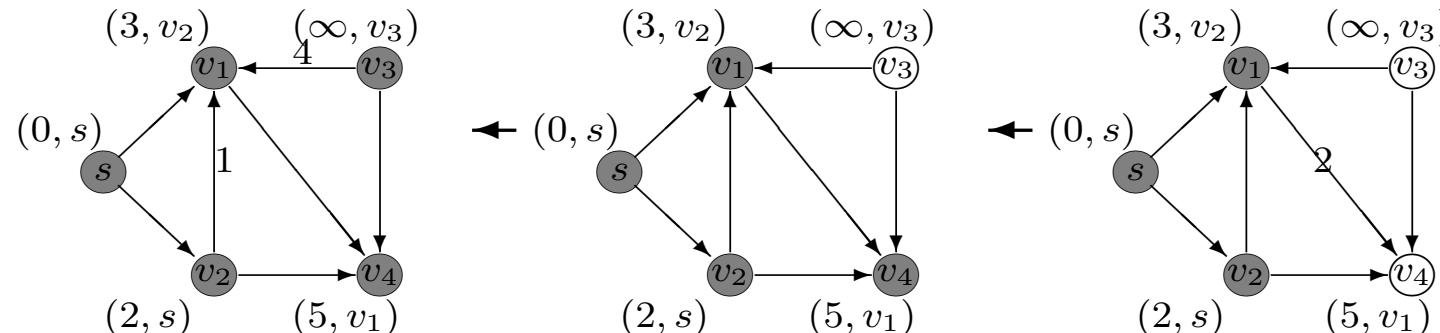
- ダイクストラ法 (Dijkstra Algorithm) を用いている
- 各ルータのリンク(インターフェース)情報をネットワーク全体で共有
 - 例: R1はR2と繋がっている
 - 例: R1には133.27.4.0/24が繋がっている
- ネットワーク全体にflooding
 - ネットワーク内のすべてのルータにリンク情報が伝わる
 - 全ルータは同一のリンク情報データベースを持つ
 - 状態に変化があったリンクの情報だけを伝える
- 各ルータが、リンク情報からトポロジを再構成
 - リンク情報を基に、自分がルート(根)となるツリーを作成
 - ツリーにすれば、ある宛先までの最短経路がわかる
 - 全ルータが同一の計算方法

Dijkstra Algorithm

- 2点間の最短経路を求めるアルゴリズム
 - インターネットの経路制御やカーナビ等にも応用
- 手法
 - 始点から常に最短となる隣接頂点を見つけていき、その距離をラベリングすることで最短経路を発見する

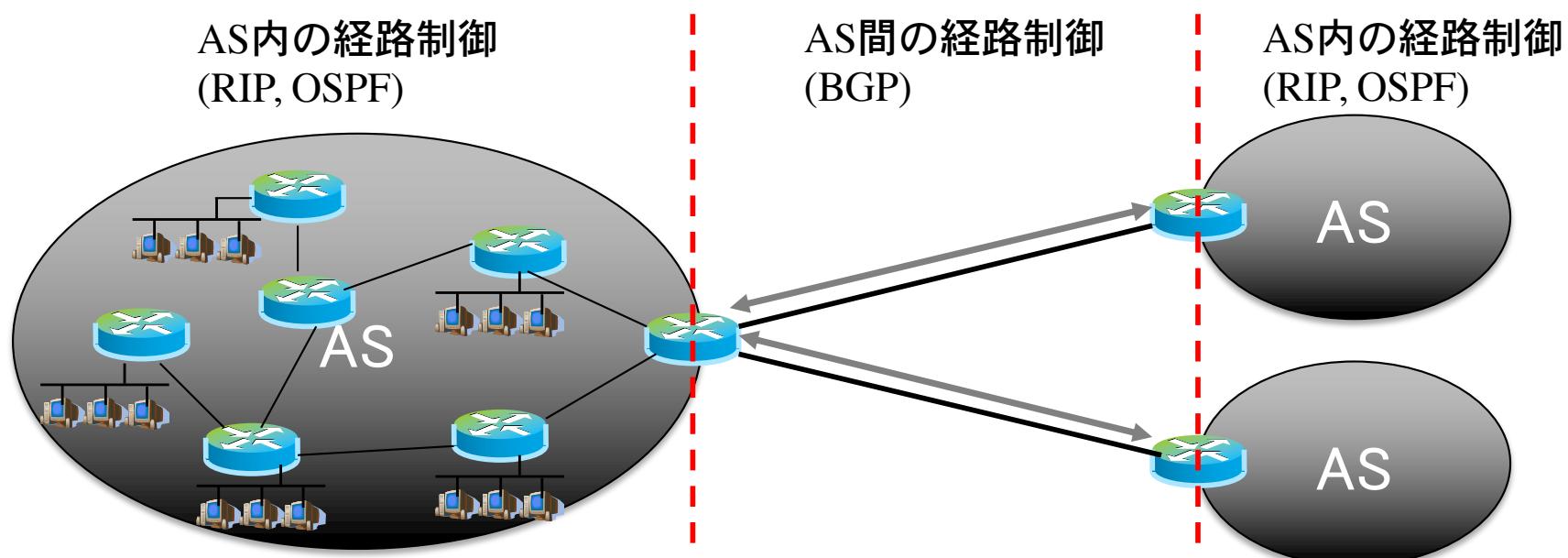


問題例と初期ラベル



経路制御の階層化

- 規模性の問題
 - 全世界を一つのRIPやOSPFドメインで接続することはできない
- 障害範囲の限定
 - AS内の障害が世界中に伝播しない
- 経路数の軽減
 - 経路をAS外に広告する場合、経路を集約できる



TCP と UDP

- IP (IPv4/IPv6) の上で動くトランスポートプロトコル
- TCP (Transmission Control Protocol)
 - RFC793
 - 一対一での信頼性のある通信を行う
 - パケットが届いたかどうかの確認を行う
- UDP (User Datagram Protocol)
 - RFC768
 - 宛先に対して投げっぱなし
 - 到達確認は行わない

TCP/UDP パケットの位置

Ethernet フレーム

Ethernet ヘッダ

データ(IP パケット)

IP パケット

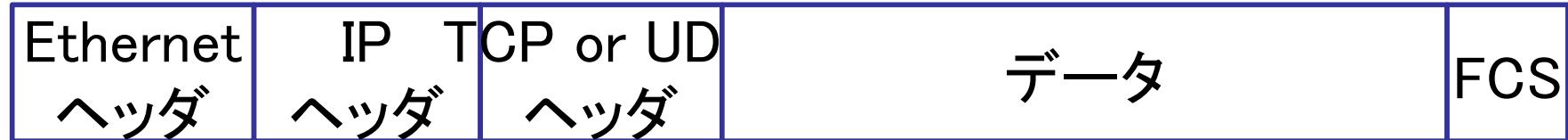
IP ヘッダ

データ(TCP/UDP)

TCP/UDP
ヘッダ

データ

データフォーマット



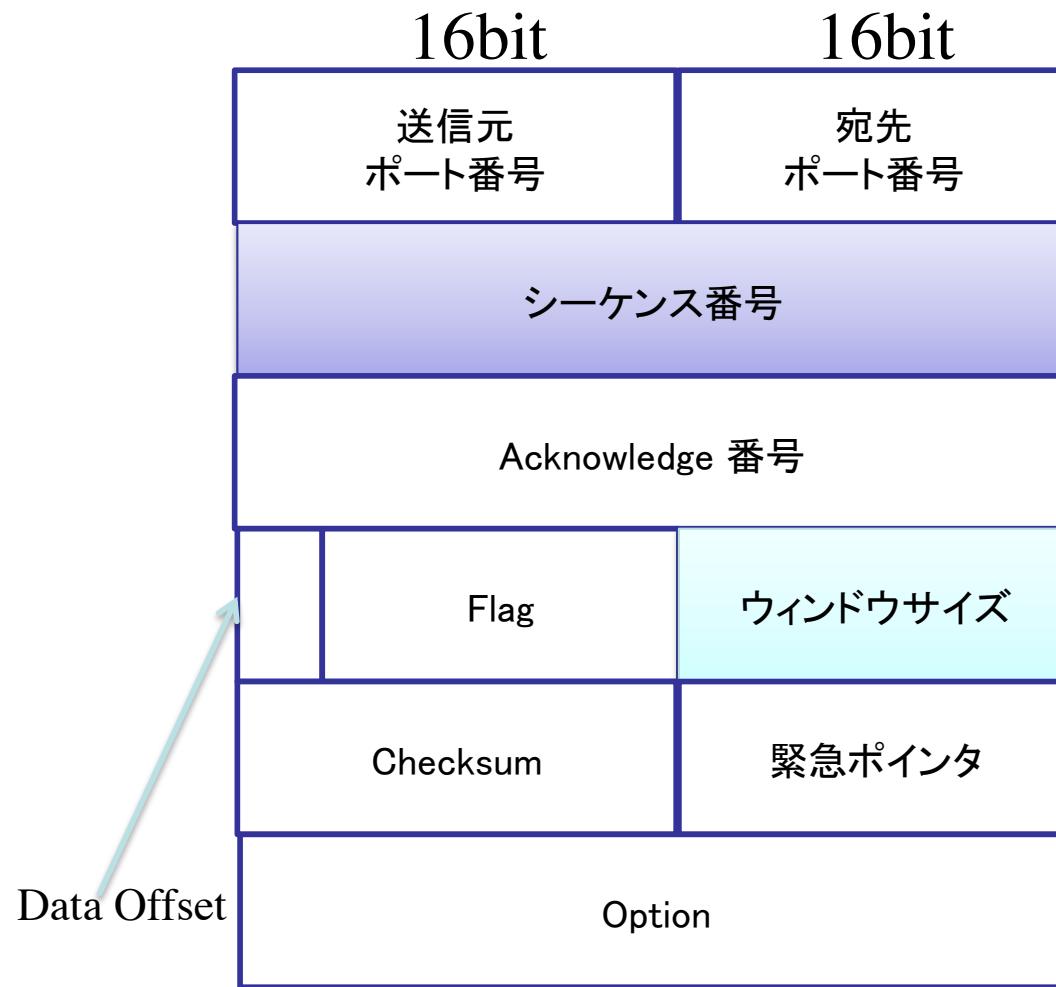
- UDP ヘッダ

送信元 ポート番号	宛先 ポート番号
長さ	Checksum

- TCP ヘッダ

送信元 ポート番号	宛先 ポート番号
シーケンス番号	
Acknowledge 番号	
Flag	ウィンドウサイズ
Checksum	緊急ポインタ

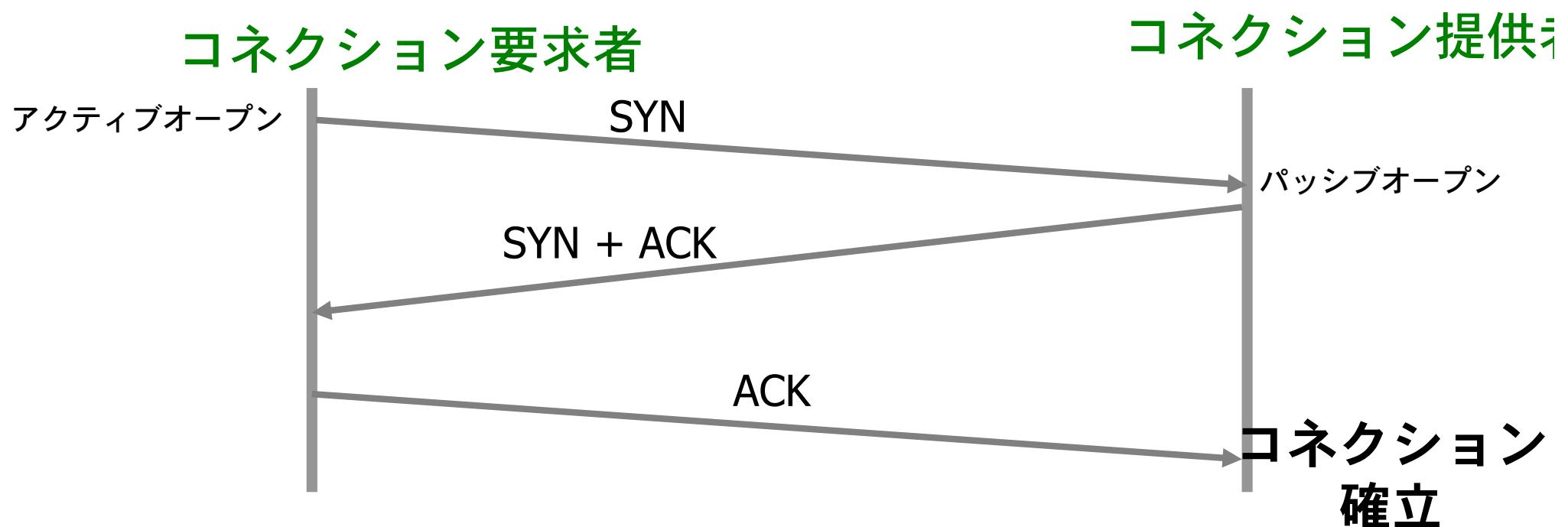
TCP ヘッダ



- シーケンス番号
 - 送信するデータの番号
- Acknowledge 番号
 - 受信したデータの番号
- Data Offset
 - 実際のデータが始まる位置
- Flag
 - URG / ACK / PSH / RST / SYN / FIN
- ウィンドウサイズ
 - 受信確認を待たずとも送付可能なデータの最大サイズ
- 緊急ポインタ
 - 緊急データの位置
- Option
 - 主に MSS (Maximum Segment Size) を伝えるのに使う

TCPの仕組み（コネクションの確立方法）

- 3 way handshake
 - SYN: コネクションを初期化する
 - ACK: 確認応答

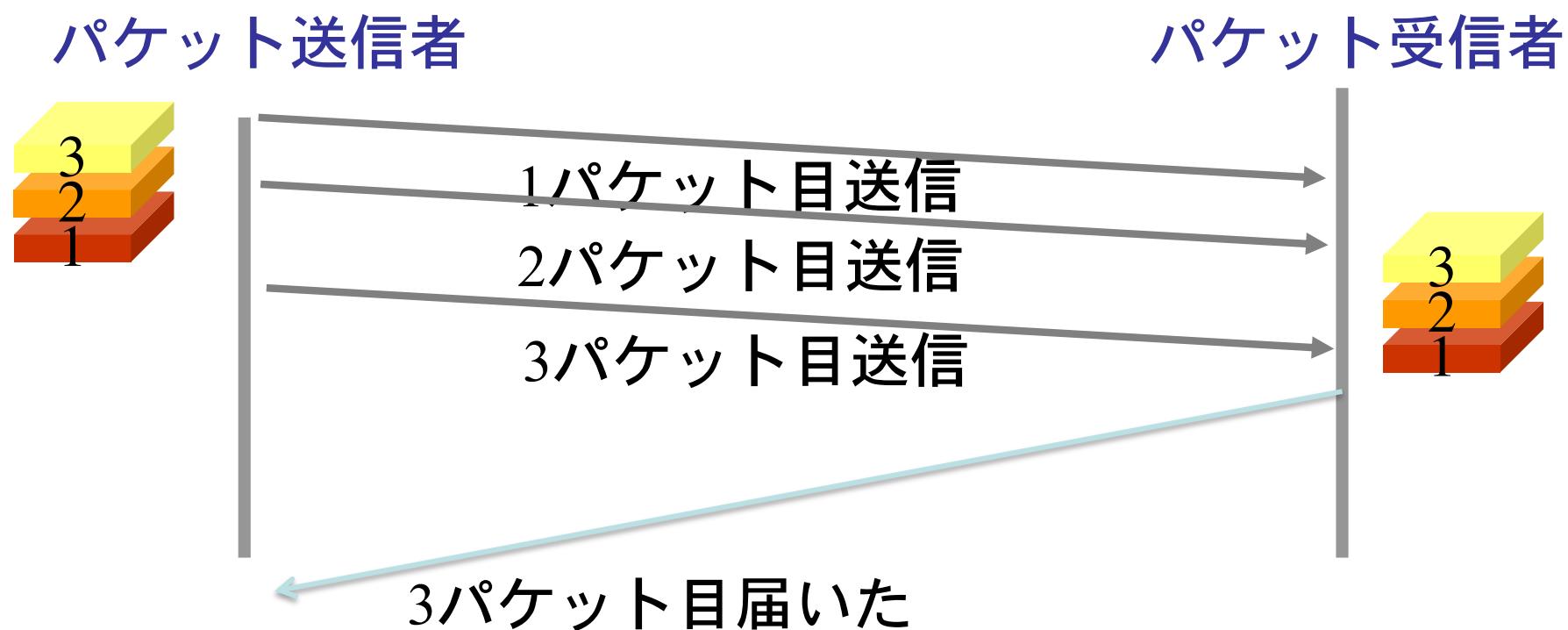


TCPの仕組み(再転送)



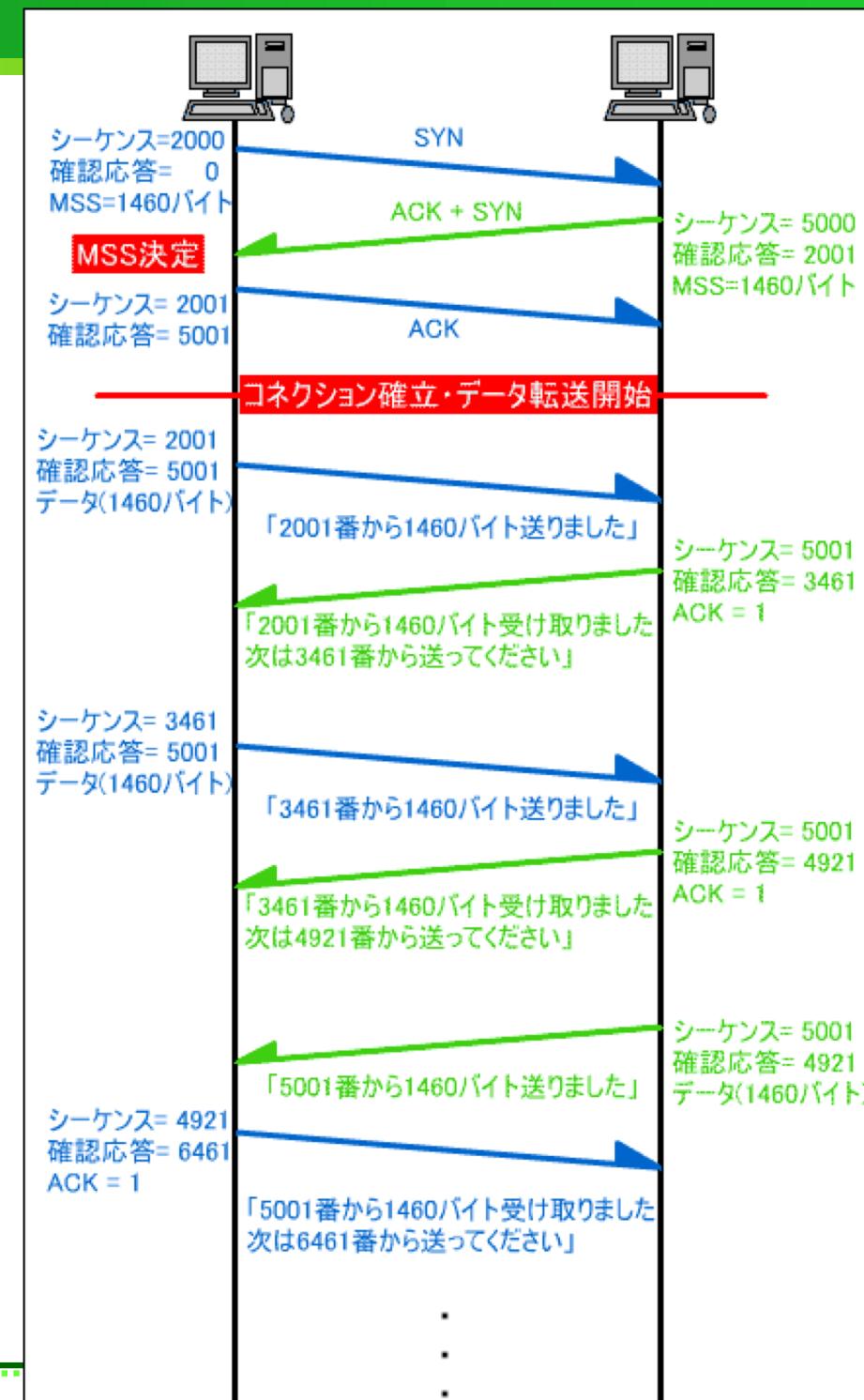
TCPの仕組み(まとめ送り)

- Ack を待たずに送り出す
- まとめて送れるデータ量 = Window Size
- 1パケットの最大サイズ = MSS (Maximum Segment Size)



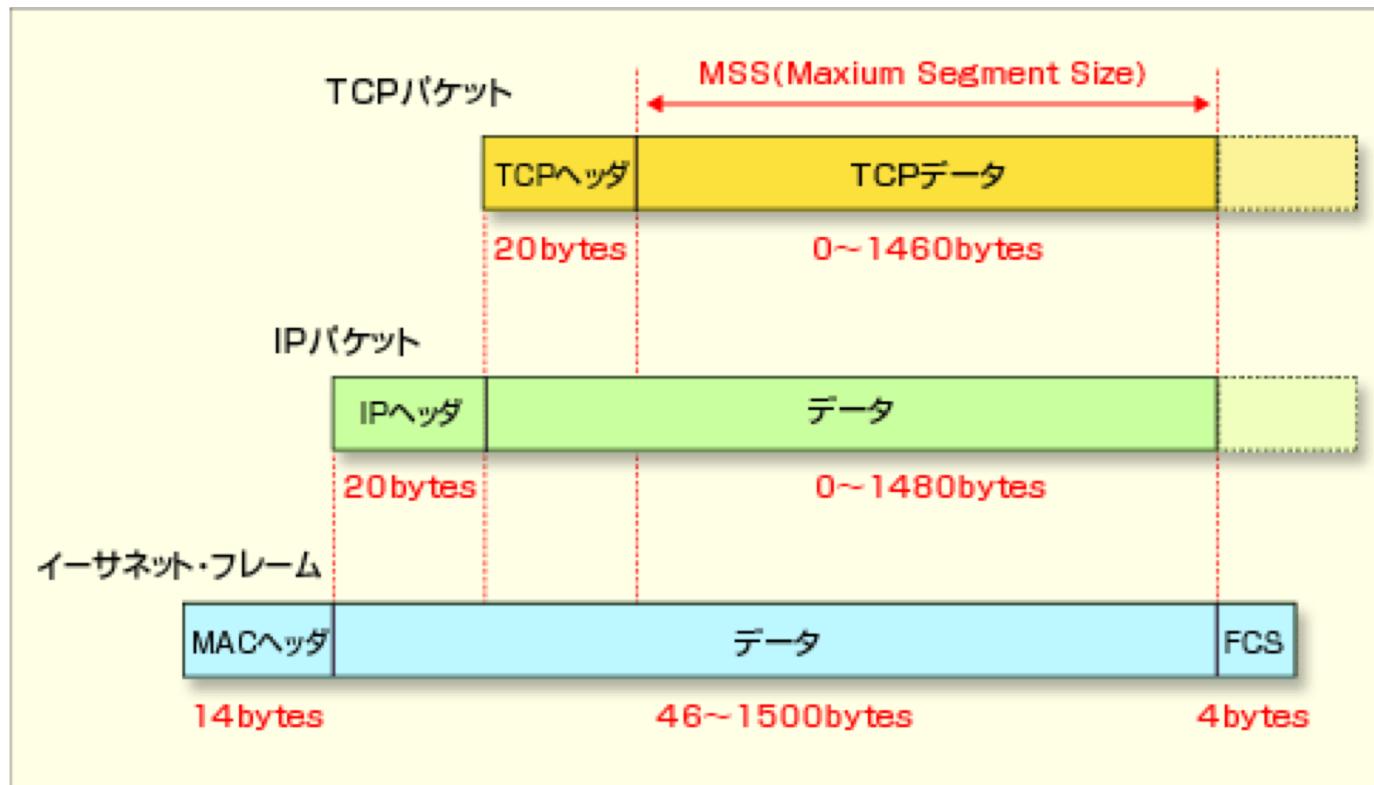
シーケンス番号

- データ列の番号
- 初期値はランダム
 - 右の場合は 2000 と 5000
- 送った byte 分だけ番号を増加させる
- Ack は受け取った番号 + 1 の番号で返す



MSS (Maximum Segment Size)

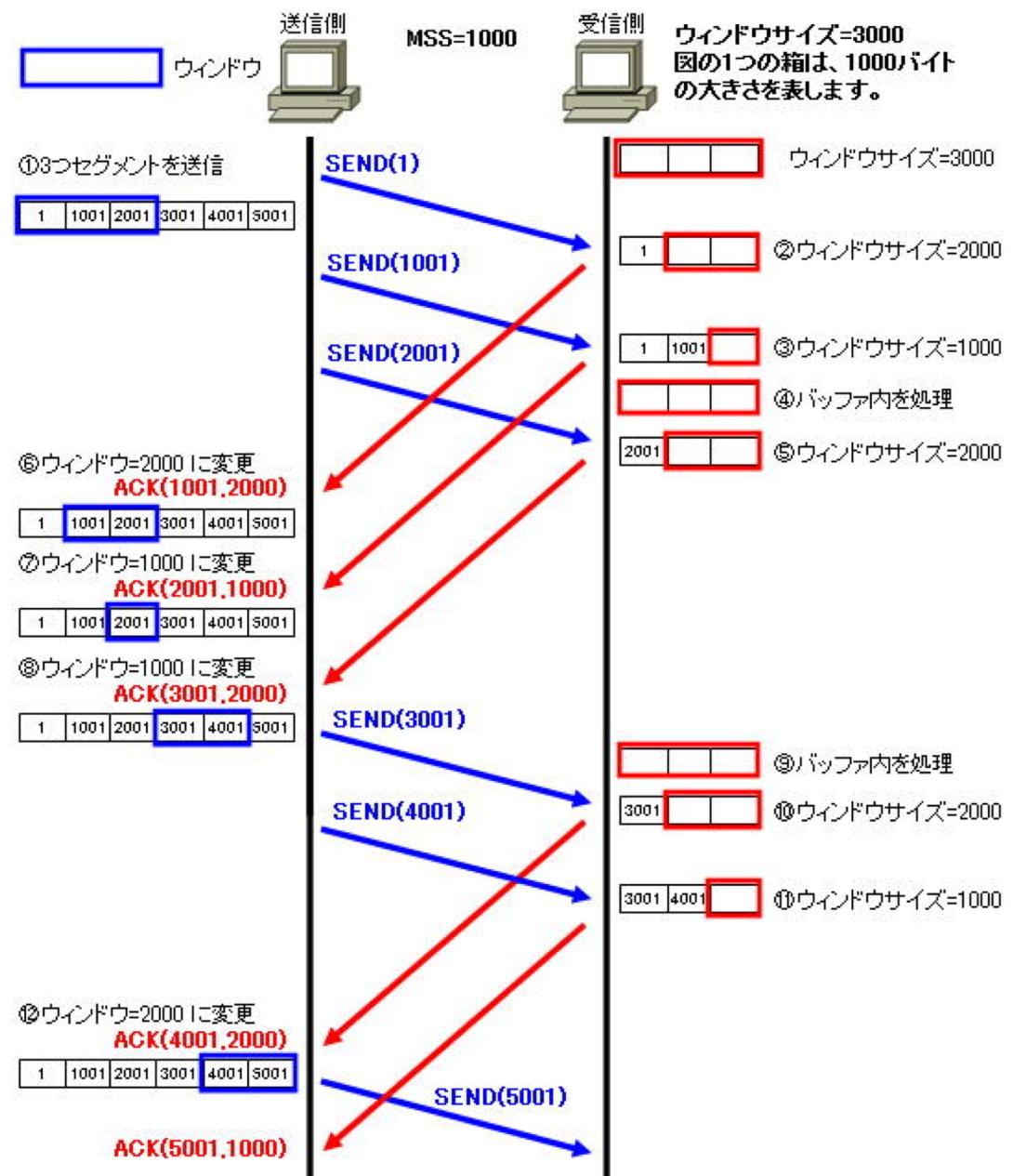
- 1パケットで送れる最大実データ量
- TCP のオプションとして通知可能



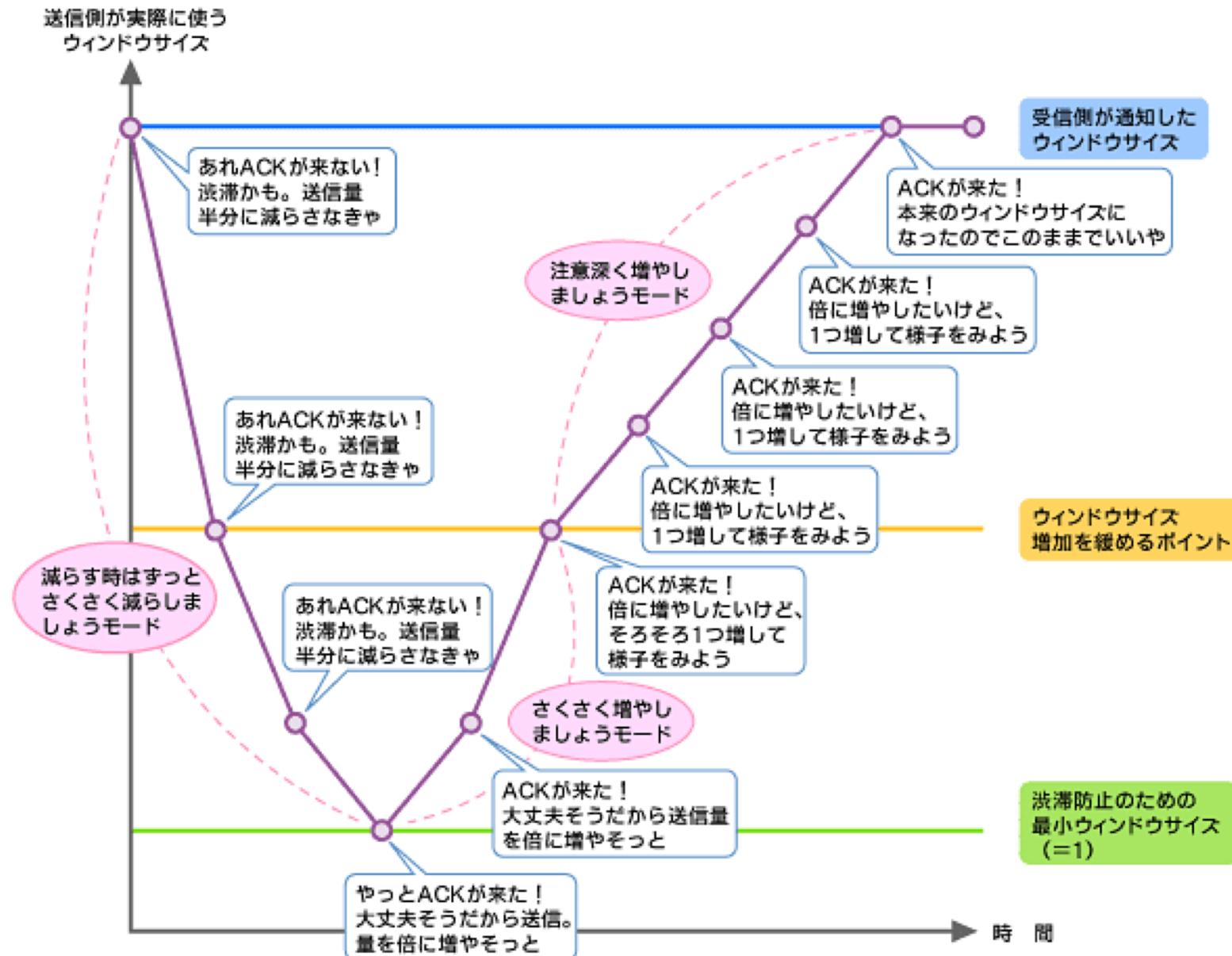
出典 : http://www.atmarkit.co.jp/fwin2k/network/baswinlan016/baswinlan016_02.html

Windows Size

- Window Size
 - 連続して送受信可能なデータの最大サイズ
 - 受信状況によって、刻々と Window Size は変化する

出典 : <http://atnetwork.info/tcpip/tcpip104.html>

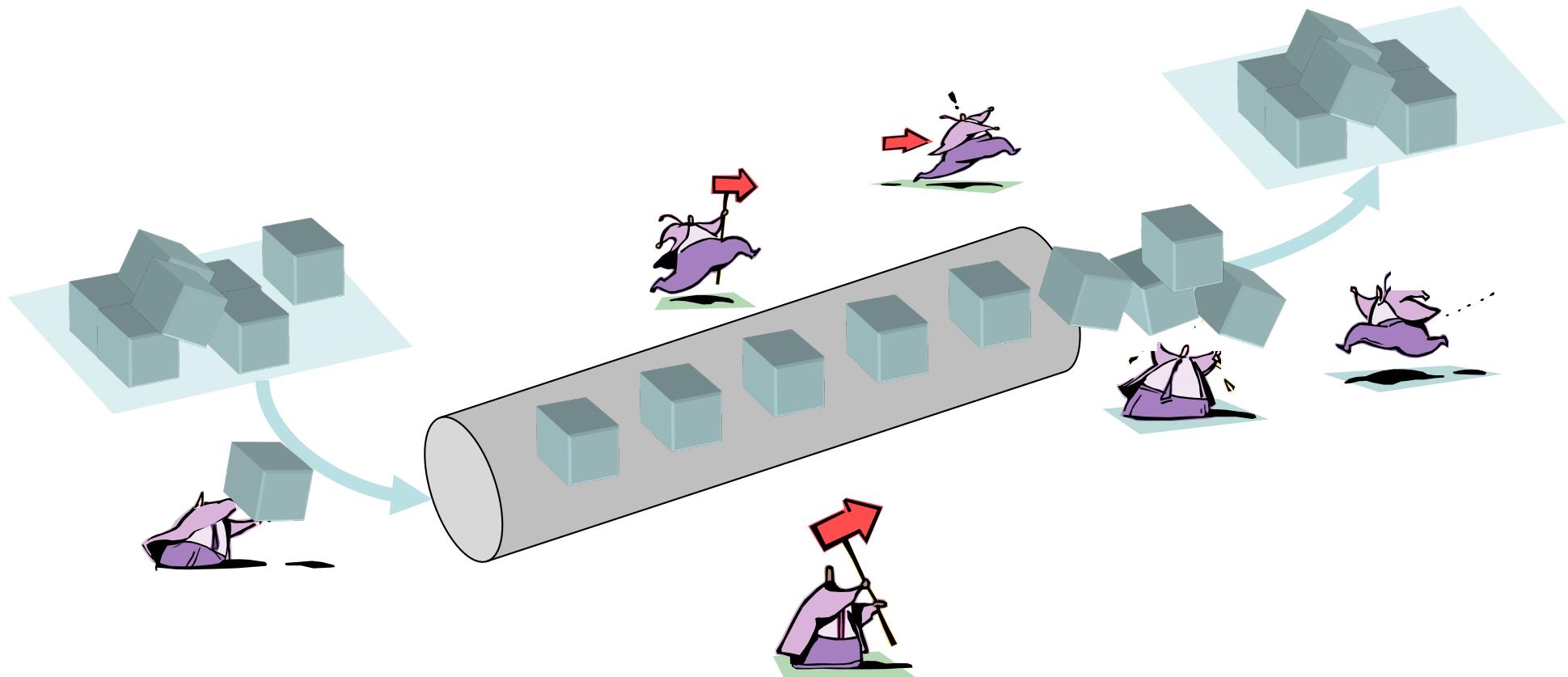
輻輳制御の仕組み



出典 : <http://www.atmarkit.co.jp/fnetwork/rensai/tcp11/02.html>

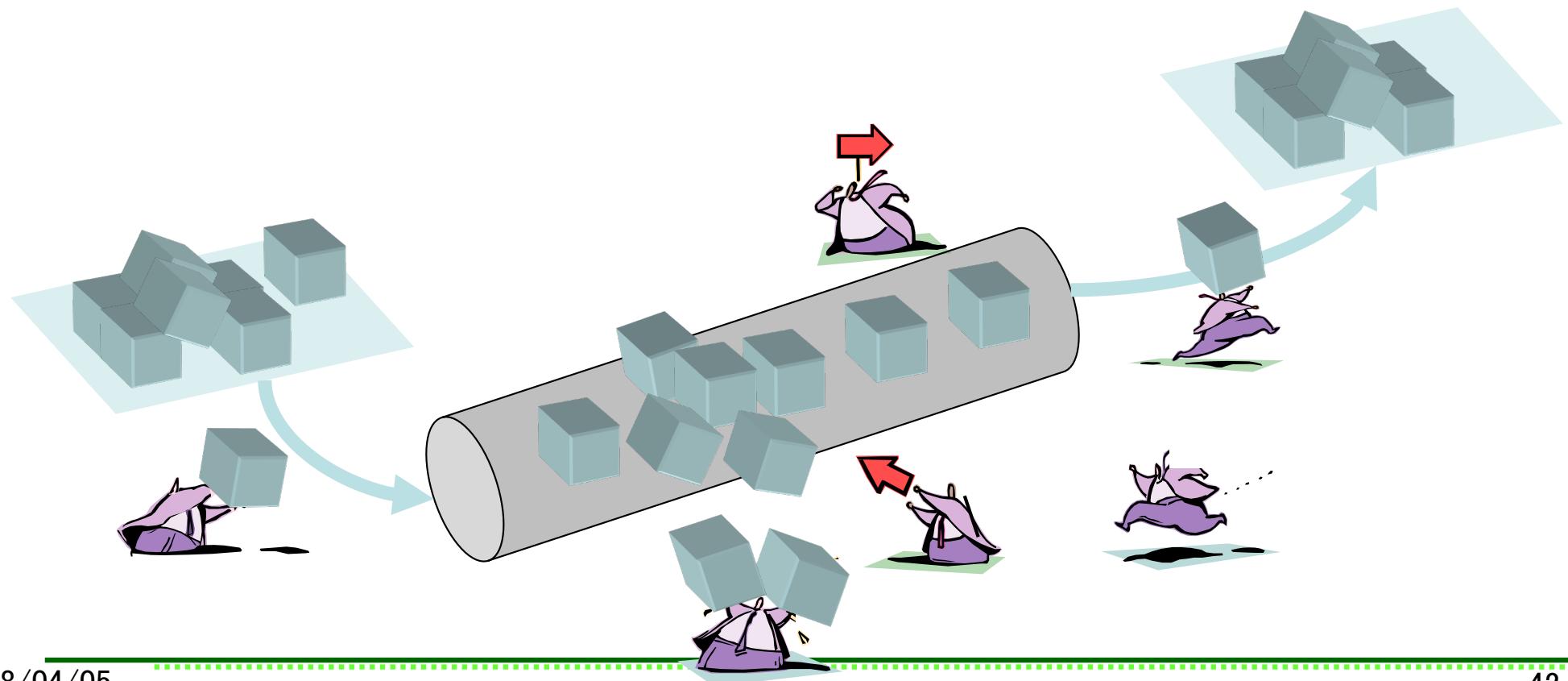
フローントロール

- 相手があふれたら、送信者が加減する
 - Window Size による調節
 - 受信者が主体



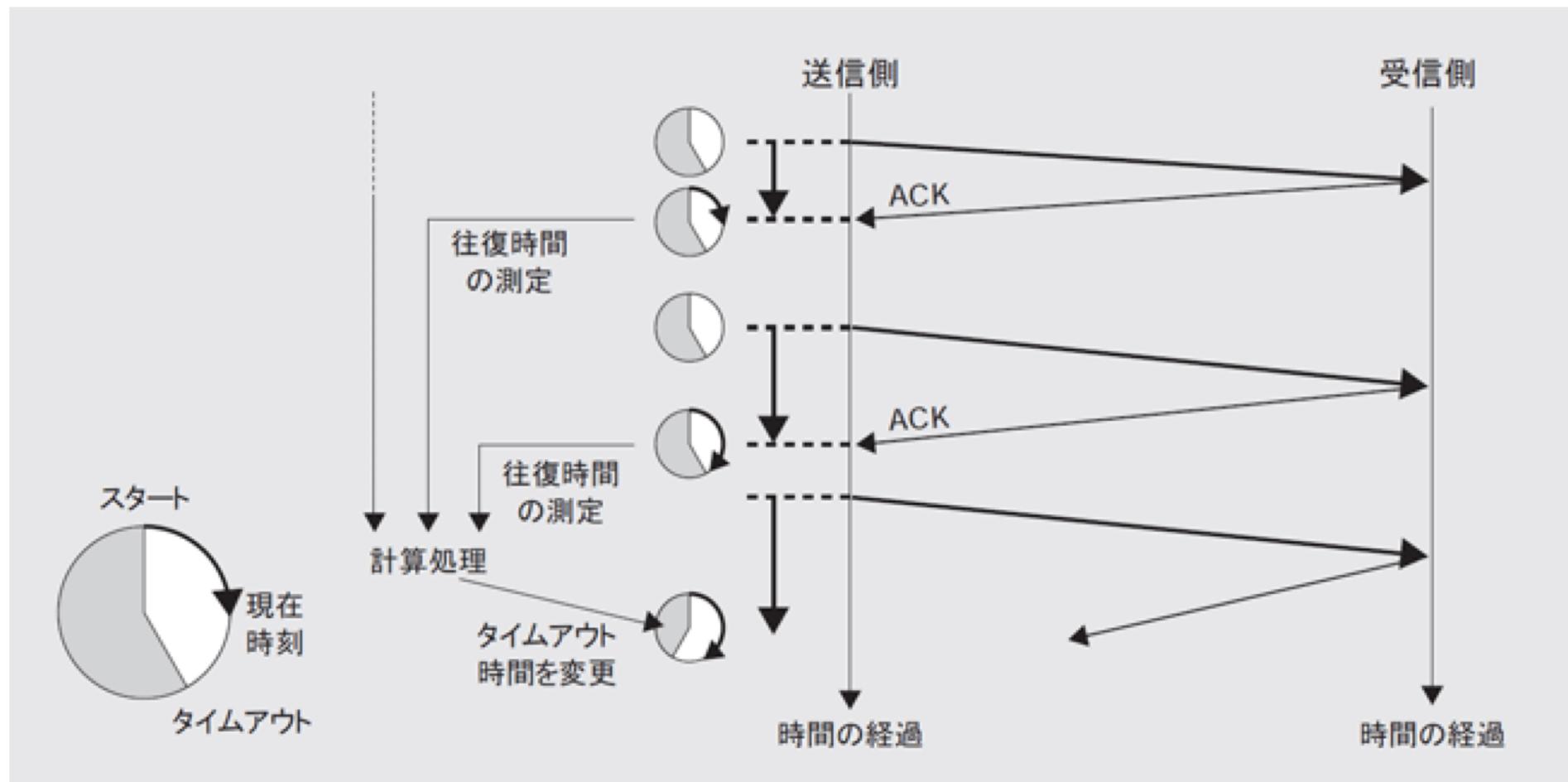
コンジェスチョンコントロール

- congestion = 輻輳
- ネットワークが混んでたら送信者が加減する
 - 相手が受け取れているか自信がない
 - 送信者が主体



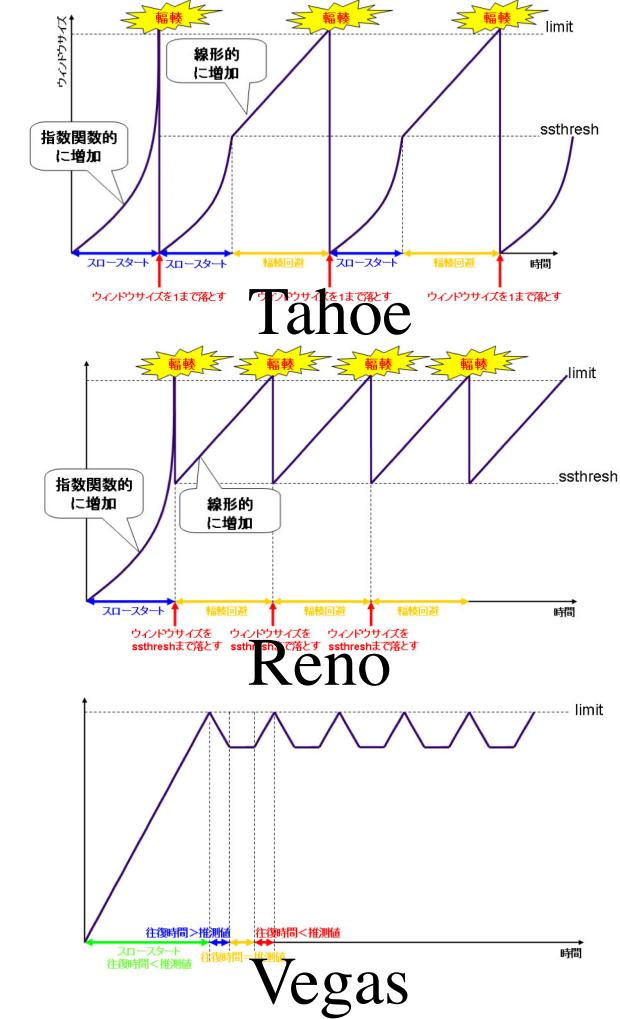
タイムアウト

- Ack 待ち時間を決めるために RTT (Round Trip Time) を利用



TCP アルゴリズムの種類

- Tahoe
 - スロースタート
- Reno
 - Fast Recovery
- NewReno
 - Fast Recovery アルゴリズムの修正
- Vegas
 - 輻輳の発生予想に RTT を利用
- CTCP
 - Reno と Vegas の複合
 - Windows Vista 以降



NAT とは ?

- NAT
 - Network Address Translation
- NAPT
 - Network Address and Port Translation
 - 別名 IP Masquerade
- Global IP address と Private IP address を変換する仕組み
 - 家庭用のブロードバンドルータ
 - 研究室内ネットワーク

IPアドレスとポート番号

	IP address	Port
送信元	203.178.1.10	25
宛先	130.69.1.1	50492



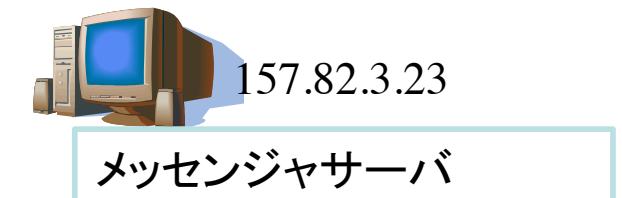
Mail サーバ
203.178.1.10



Mail
Web
Messenger

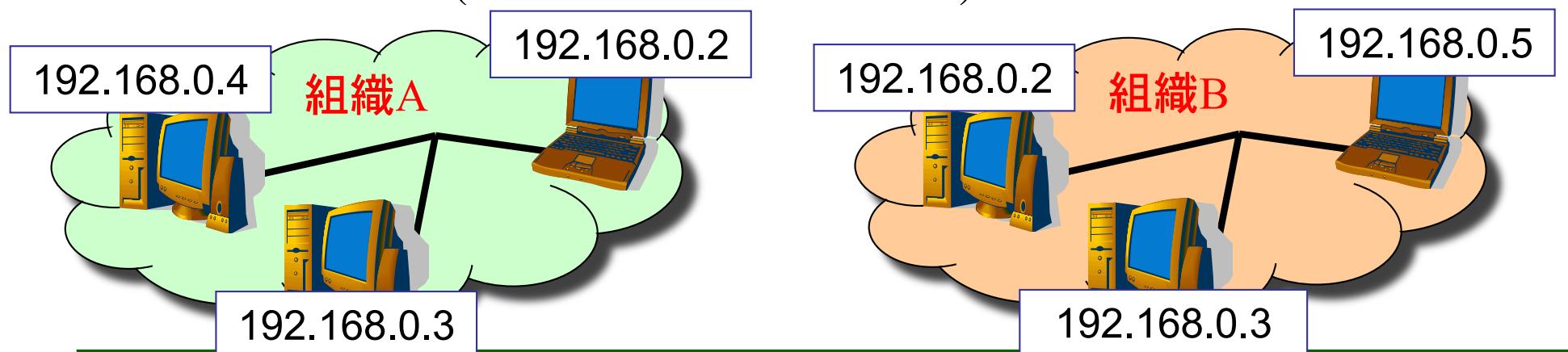


	IP address	Port
送信元	133.11.5.30	80
宛先	130.69.1.1	50388



プライベートIPアドレス

- 限られた範囲内で一意になるアドレス
 - そのままでは外と通信できない。
 - 自宅やオフィスのLAN、ファイアウォールの内側などで利用される
 - RFC1918
 - 192.168.0.0/16 (192.168.0.0 – 192.168.255.255)
 - 172.16.0.0/12 (172.16.0.0 – 172.31.255.255)
 - 10.0.0.0/8 (10.0.0.0 – 10.255.255.255)

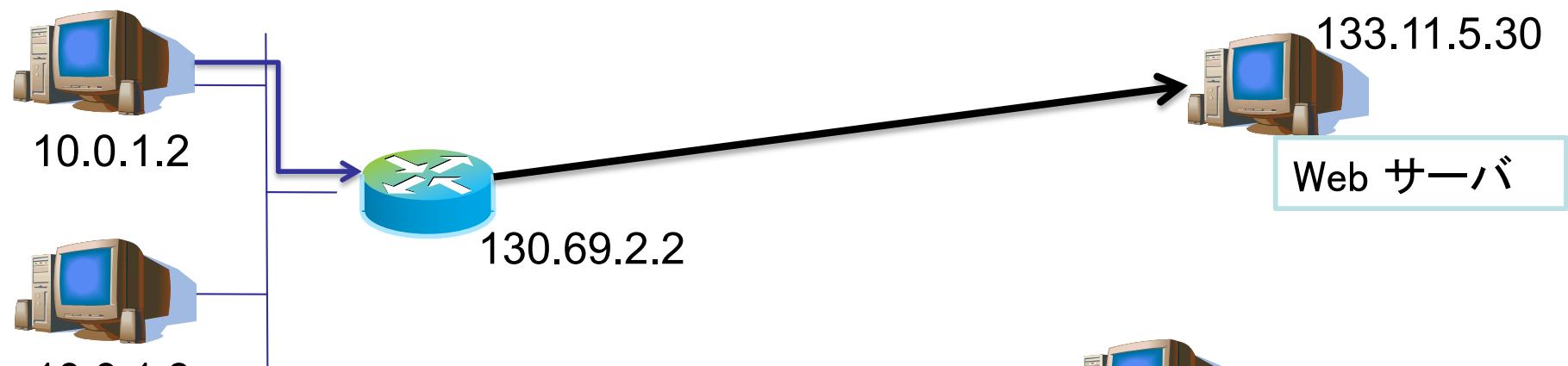


IPアドレスとポート番号(NAT)

	IP address	Port
送信元	10.0.1.2	50872
宛先	133.11.5.30	80



Mail サーバ
203.178.1.10



	IP address	Port
送信元	130.69.2.2	3098
宛先	133.11.5.30	80

133.11.5.30
Web サーバ

157.82.3.23
メッセンジャーサーバ

IPアドレスとポート番号(NAT)

	IP address	Port
送信元	133.11.5.30	80
宛先	10.0.1.2	50872



Mail サーバ
203.178.1.10



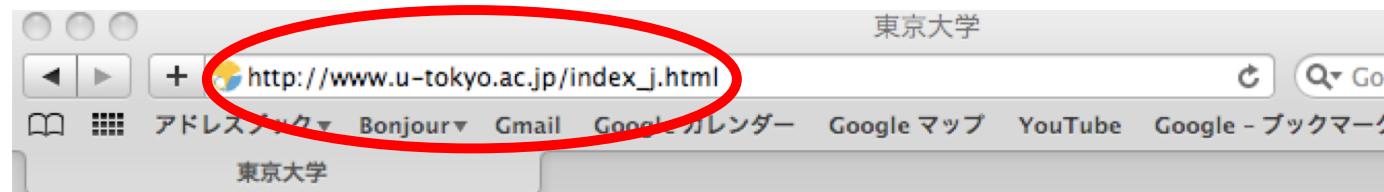
	IP address	Port
送信元	133.11.5.30	80
宛先	130.69.2.2	3098

157.82.3.23
メッセンジャーサーバ

DNS とは

- Domain Name System の略
- ユーザが何か「通信をしよう！」と行動を取った場合
 - 後ろでがんばって動いています
 - 「名前と IP アドレス」の変換を行います
- いわばインターネットの電話帳

ドメイン名はここに

A screenshot of a web browser displaying an email message. The subject line 'Subject: テストメール' is circled in red. The message body contains various header fields and a message body. The message body starts with 'テストメール送信。' and ends with '-- Yuji Sekiya'.

東京大学案内 | 学部・大学院・研究所・センター | 東京大学
| 受験生の方へ | 在学生の方へ | 留学生の方へ | 卒業生の方へ

明日の日本を支えるために
—教育研究の危機を越えて—

【重要】新型インフルエンザについて

Information

■ 柏キャンパス

■ 医学部附属病院(本郷)

Topics

総長講演会

to: sekiya@nc.u-tokyo.ac.jp
Subject: テストメール
From: Yuji Sekiya <sekiya@nc.u-tokyo.ac.jp>
X-Spam-CheckResult: ham, SpamAssassin 3.1.8 (2007-02-13) on anzu
X-Spam-Level:
X-Spam-Status: No, score=-3.8 required=5.5 tests=AWL,BAYES_00,FORGED_RCVD_HELO, IS0202
Version=3.1.8
X-Original-To: sekiya@anzu.nezu.wide.ad.jp
Delivered-To: sekiya@anzu.nezu.wide.ad.jp
Date: Wed, 16 Dec 2009 05:56:34 +0900
Message-ID: <cm27hsox8fx.wl%sekiya@wide.ad.jp>
User-Agent: Wanderlust/2.15.5 (Almost Unreal) SEMI/1.14.6 (Maruoka)
FLIM/1.14.9 (Gojo) APEL/10.7 Emacs/22.2.50 (i386-apple-darwin9.4.0)
MULE/5.0 (SAKAKI)
Organization: The University of Tokyo
MIME-Version: 1.0 (generated by SEMI 1.14.6 - "Maruoka")

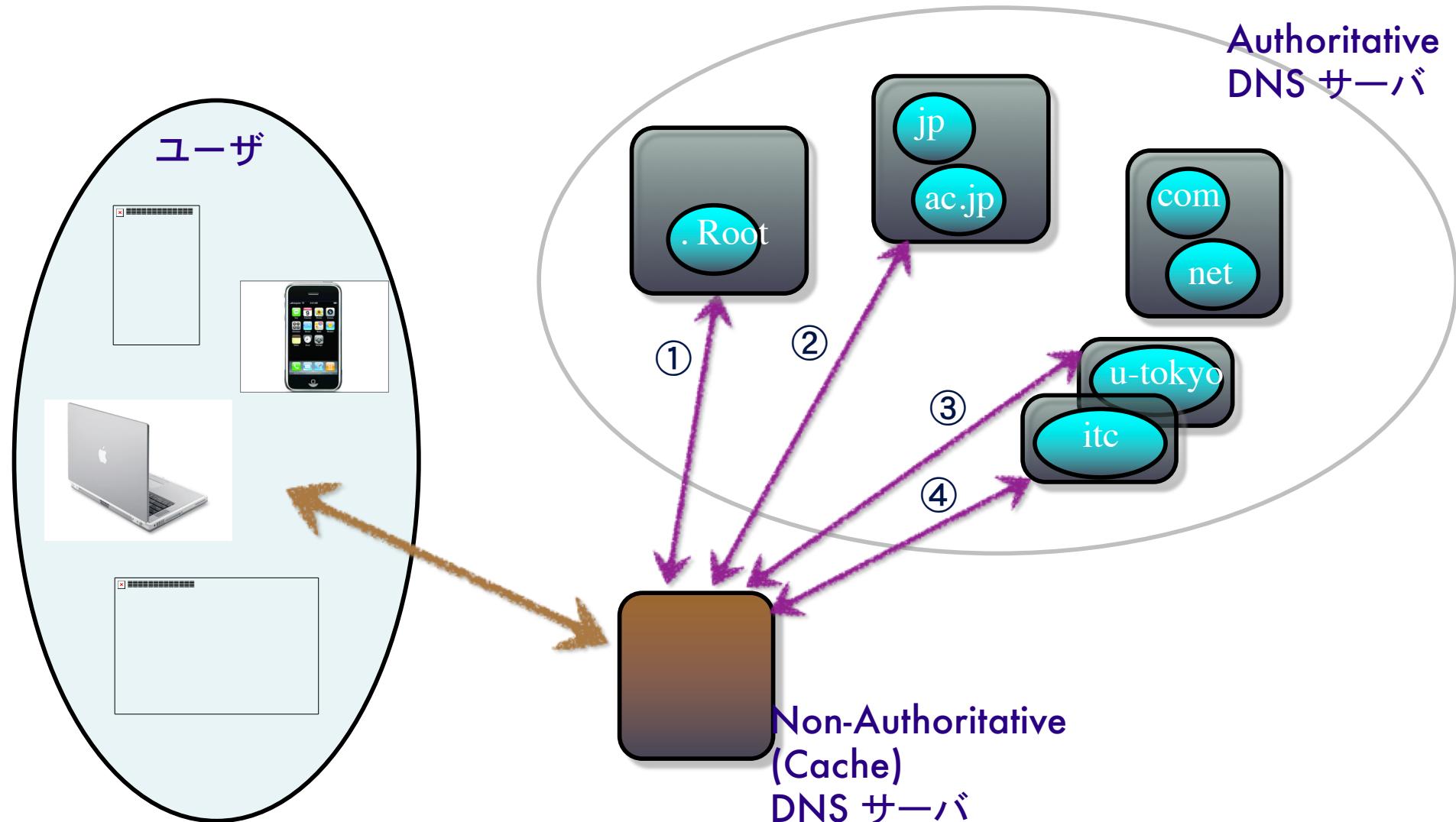
テストメール送信。

-- Yuji Sekiya

名前(ドメイン名) の仕組み

- 階層構造になっている
 - www.itc.u-tokyo.ac.jp
 - jp 日本
 - ac 大学組織
 - u-tokyo 東京大学
 - itc 情報基盤センター
 - www ホストの名前

DNSの動作概要



<http://www.itc.u-tokyo.ac.jp/>

インターネットにとって DNS とは

- 欠かすことのできない基盤サービス
 - 唯一の世界規模分散データベース
- 正しい情報を教えてくれることが前提となっている
 - DNSが嘘をつくと、すべてのアプリケーションがだまされる
 - 電話帳や番号案内みたいなもの