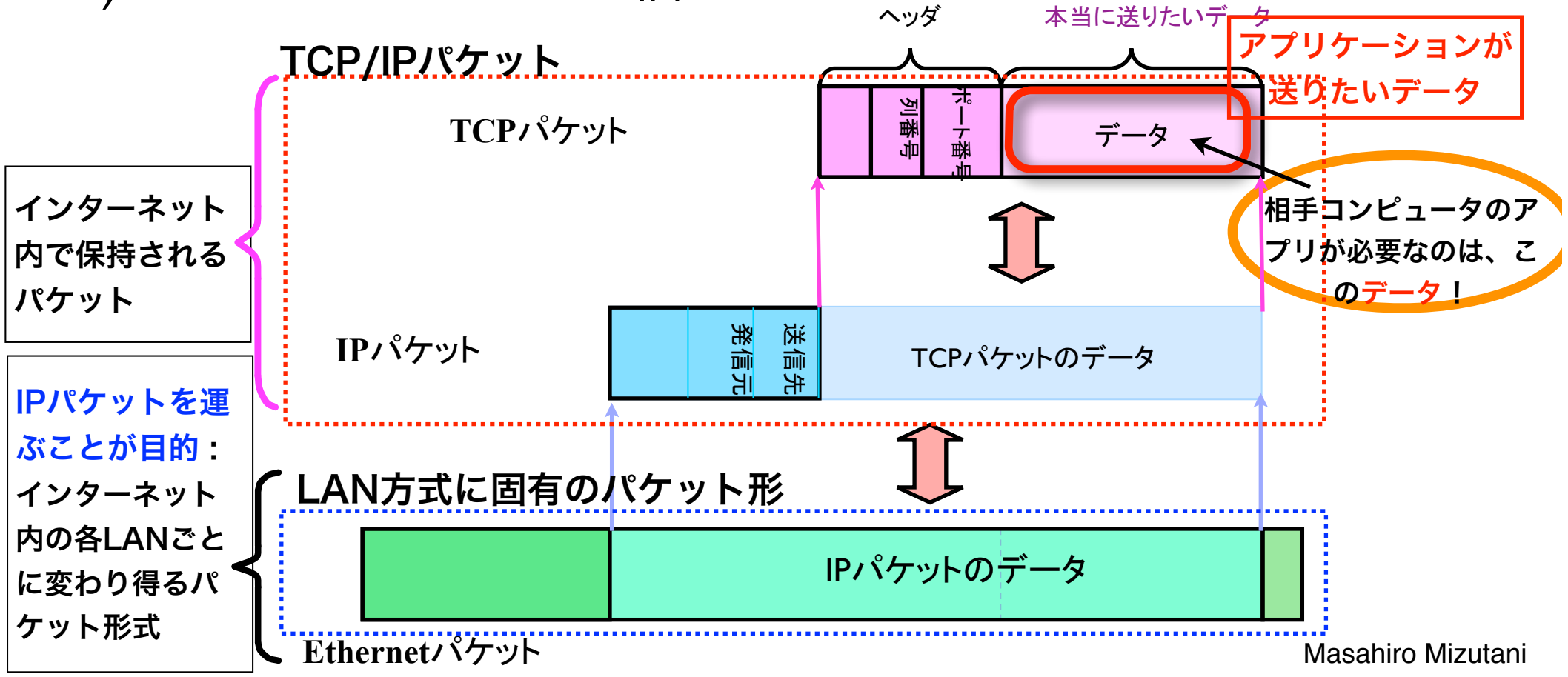
IPアドレスと経路制御
TCPパケット

水谷正大

TCP/IP体系のパケットの扱い

1) 送りたいデータをTCPパケットに載せる
2) TCPパケットをIPパケットに載せる
3) IPパケットを各LAN固有のパケットに載せる
4) コンピュータから送信

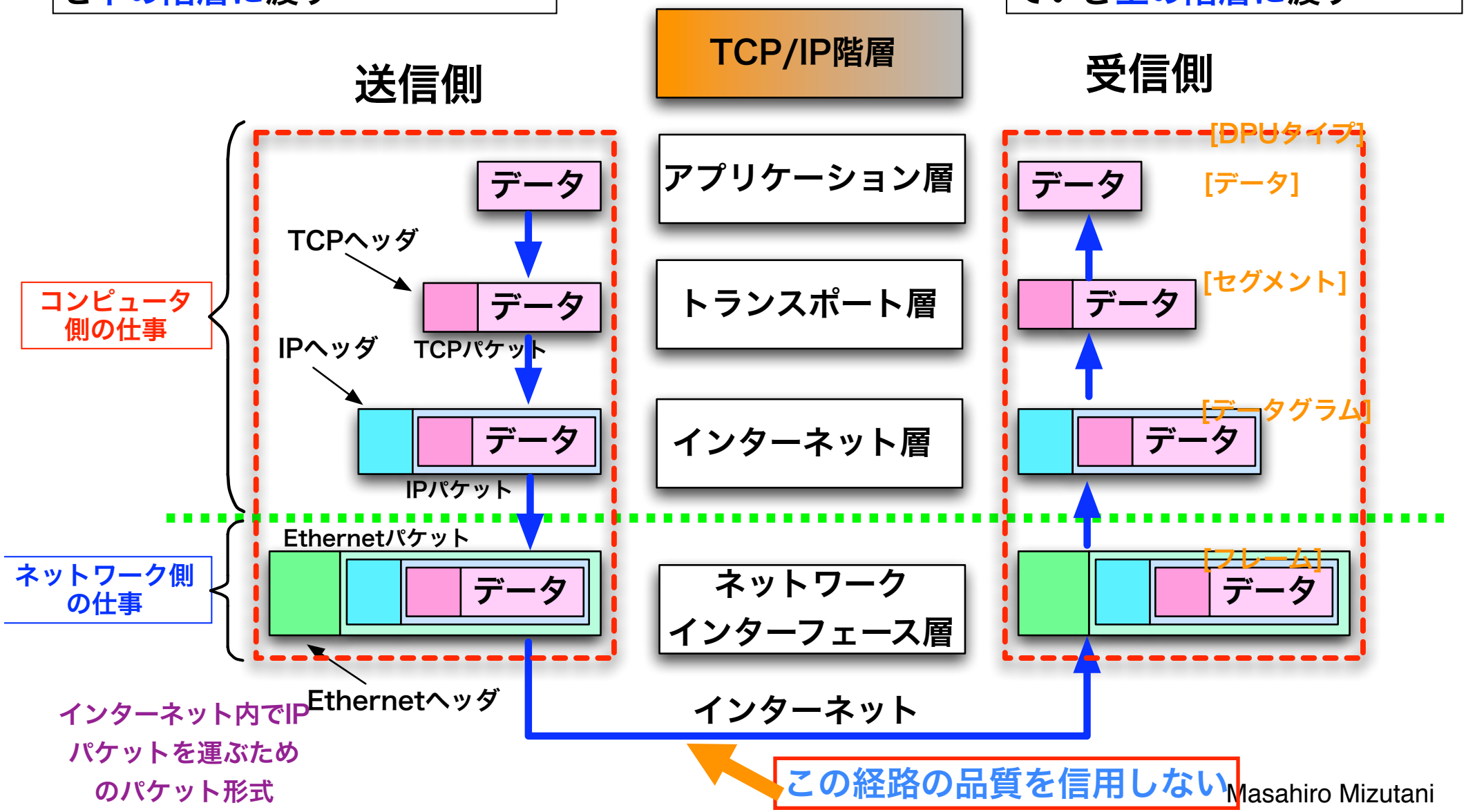


データが相手に届くまで

送信側の責任：データに正しいヘッダを付与しながらパケットを**下の階層に**渡す

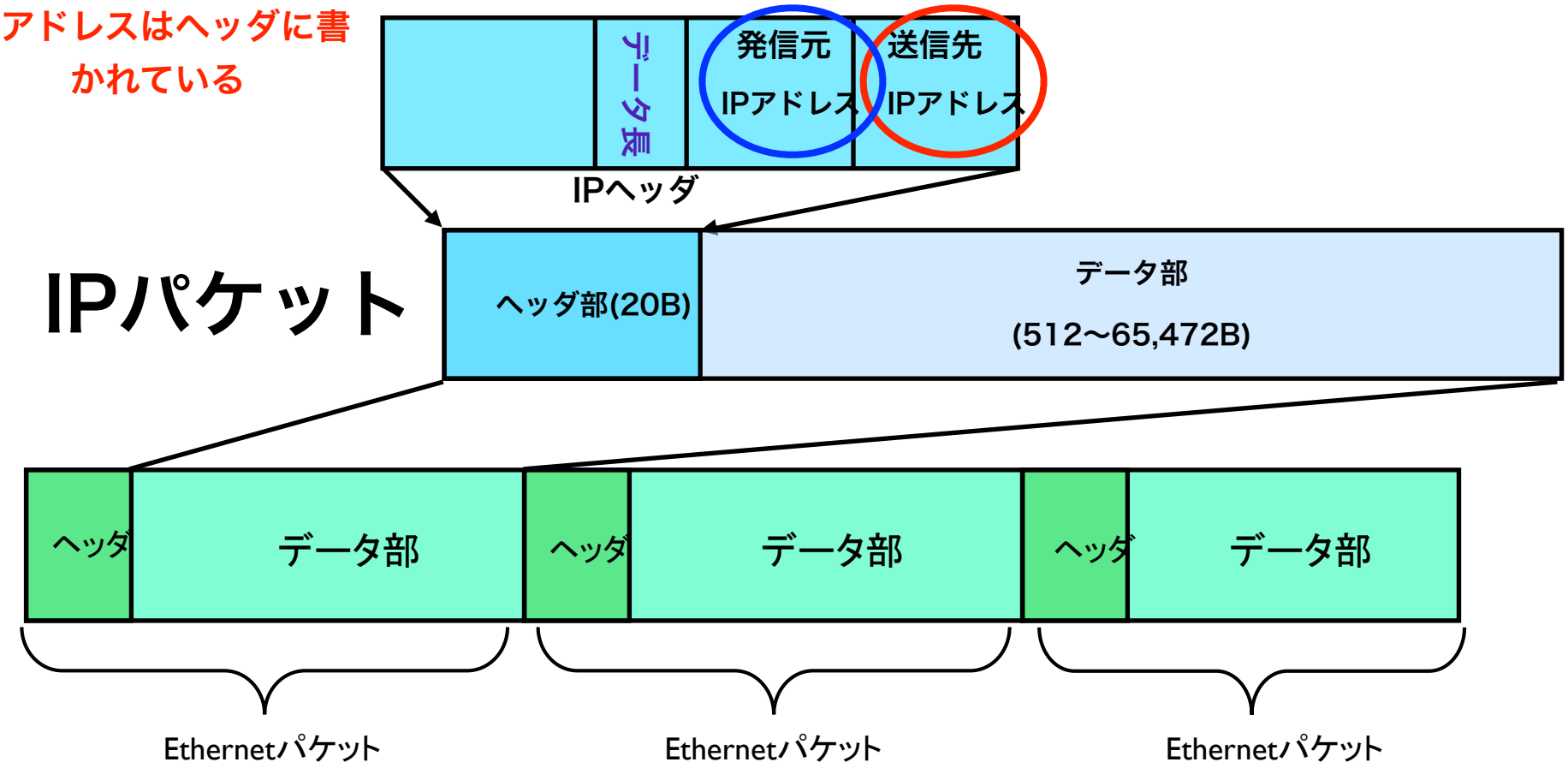
各層同士で、層に対応するデータ通信をやりとりしている

受信側の責任：パケットを受信し、ヘッダを取り除いていき**上の階層に**渡す



IPパケットの構造

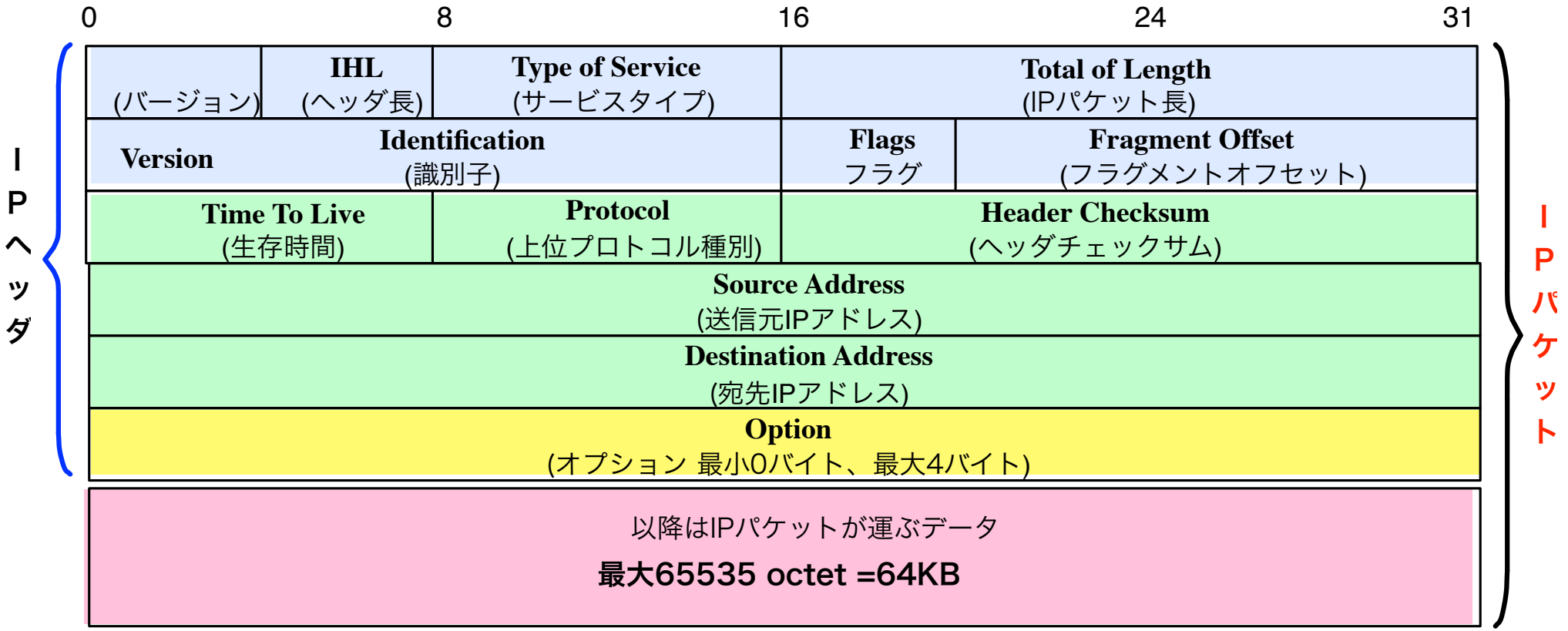
IPアドレスはヘッダに書かれている



IPパケットはネットワークインターフェイス層で伝送されるパケット形式（Ethernetなど）が運ぶ

IPヘッダの構造(I)

IP v4の場合



IPヘッダ　20〜24 octet　　固定長 4octet x 5 = 20octet

オプション 0〜4octet x 1 = 0〜4octet

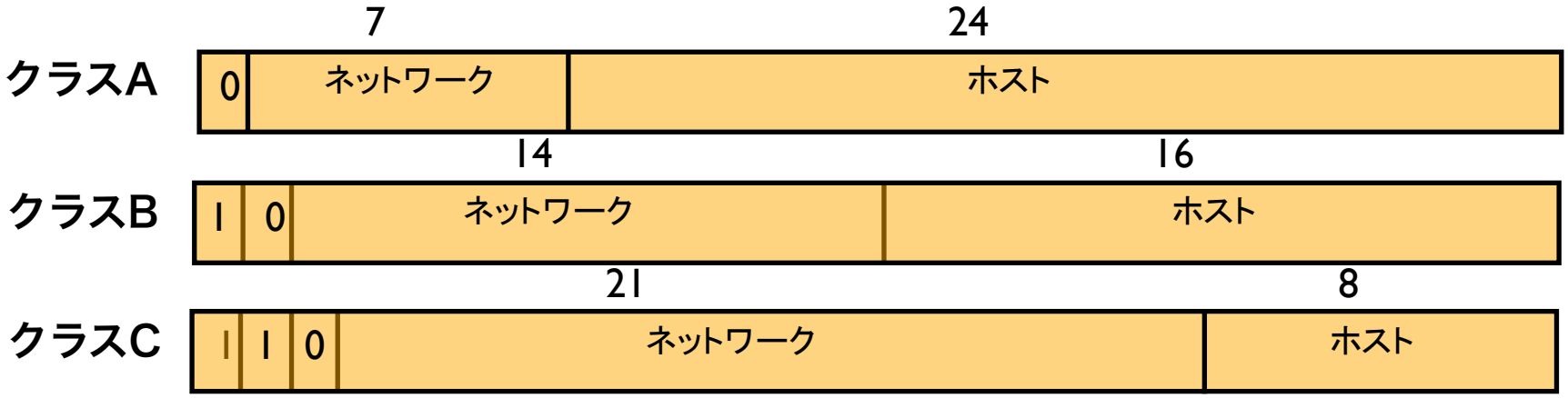
1octet = 8bit

IPパケットデータ　最大65535 octet =64KB

1つのパケット(フレーム)で送信できる最大サイズ(MTU Maximum Transmission Units)は、Ethernetで最大1500B程度。経路途中・送信先のMUTに合わせて自動的にIPパケットのサイズを調整(フラグメント化)

IPアドレス

- ホストを識別するためのビットパターン
  - [ネットワーク]+[そこでのホスト] から成る
- グローバルアドレスとプライベートアドレスに大別
- 原則的にはクラスA,B,Cに分かれている
  - クラスD,Eもある（一般には割り当てられない）



IPアドレスの表記

IPv4　32ビットのアドレス空間

$2^{32} = 4294967296$ (≒43億) 台を識別可能

表記：各8bitずつ[.]ピリオドで区切って10進表記

例：　www.asahi.com

125.56.200.185

IPv6　128ビットのアドレス空間

$2^{128} \sim 3.40 \times 10^{38}$台を識別可能

表記：各16bitずつ [:]コロン で区切って16進表記

例：　google.com

2404:6800:4004:804:0000:0000:0000:1003

（2404:6800:4004:804::1003と略記）

0000が次ぐ部分は一箇所だけ :: と省略できる

IPv4プライベートアドレス

Class A　10.0.0.0～10.255.255.255

Class B　172.16.0.0～172.31.255.255

Class C　192.168.0.0～192.168.255.255

- **プライベートアドレス**
  - LAN内でのみ使える。このアドレスをもつパケットはインターネットに流しては**いけない**
  - 別のLAN内なら、同じプライベートアドレスを自由に使ってもよい
- **グローバルアドレス**（上記ブロック以外）
  - インターネットのどこからでも認識可能

IPv4各クラスの割り当て数

プライベートも含んだ計算上の最大値として

組織数の算出時に2を引くのは、最初と最後のアドレスは使用できないため

Aクラス　1.x.x.x ～126.x.x.x (xは0から255)

ネットワークaddressは8ビットだが、上位1ビットは0に固定で残り7ビット。したがって $2^7 - 2 = 126$ 組織に割当可能で、各組織で $2^{24} - 2 = 16,777,214$ 台のホストが可能

Bクラス　128.1.x.x ～191.254.x.x (xは0から255)

ネットワークaddressは16ビットだが、上位2ビットは10に固定で残り14ビット。よって $2^{14} - 2 = 16,382$ 組織に割当可能で、各組織内では $2^{16} - 2 = 65,534$ 台のホストが接続可

Cクラス　192.0.1.x ～223.255.254.x (xは0から255)

ネットワークaddressは24ビットだが、上位3ビットは110に固定で残り21ビット。よって $2^{21} - 2 = 2,097,150$ 組織に割当可能で、各組織内では $2^8 - 2 = 254$ 台のホストが接続可

Masahiro Mizutani

IPv4アドレス空間は狭い

- IPアドレスの枯渇が深刻化
- 対策
  - 内部のprivateとglobal addrとのaddress変換
  - CIDRなどの工夫も
- でも、もう限界！ (x_x);
  - IPv6(128bit)への移行が現在進行中

もうありません。(--;)

JPNIC 社団法人 日本ネットワークインフォメーションセンター
Japan Network Information Center

TOP | ENGLISH | SITEMAP

サイト内検索:

JPNICはインターネットの円滑な運営を支えるための組織で

トップページ ＞ IPアドレス

JPNICとは | インターネットガバナンス | インターネットの技術 | ドメイン名 | インターネットの歴史・統
IPアドレス | インターネットの基礎 | JPNIC認証局 | JPNICライブラリ | トピックス

IPアドレス管理の基礎知識 | IPアドレス・AS番号が欲しい時は | IPアドレス登録管理業務について | ドキュメント一覧
IPアドレス関連のミーティング | IPアドレストピックス | 統計・各種リスト | JPIRR | IPv6対応テストベッド | よくある質問

IPv4アドレス在庫は JPNIC 枯渇しました

ツイート | いいね！ 0 | プリント用ページ

IPv4アドレスの在庫枯渇に関して

アジア太平洋地域におけるIPv4アドレスの在庫が無くなりました
JPNICも通常の割り振りを終了します

2011年4月15日

2011年2月3日、インターネット上で利用されるアドレス資源をグローバルに管理するIANA (Internet Assigned Numbers Authority) において新規に割り振りできるIPv4アドレスが無くなりました。続いて2011年4月15日には、アジア太平洋地域のRIR (地域インターネットレジストリ)であるAPNICにおいても、通常の申請により割り振り可能であるIPv4アドレスの在庫がなくなり、アジア太平洋地域は、いわゆる「IPv4アドレス在庫枯渇」の状態となりました。JPNICでは独自のアドレス在庫を保有せず、APNICと共有しているため、APNICでの通常割り振り終了に伴い、JPNICでの通常の割り振りも終了しました。

Internet Society主催でWolrd IPv6 Launchが開催

http://www.worldipv6launch.org/ 2012年6月6日

ホストのIPアドレスの割り当て

- IPアドレスやDNSサーバを手動で設定
- DHCPサーバに対する自動設定プロトコル
  - DHCP(Dynamic Host Configuration)
    - DNSサーバの在りかも自動設定可能
- IPアドレスが割り当てられたか否かを確認
  - Windows：`ipconfig`
  - UNIX系： `ifconfig`

Loacal loopback address

- 自ホスト(自分自身)を表すためのIPアドレス
  - 原則は127.0.0.1　名前は `localhost`
- 自ホストで動作しているサービスにアクセスする場合に必要

Windowsでは ipconfig (1)

C:\> ipconfig
Windows IP Configuration
Ethernet adapter ローカル エリア接続:

Connection-specific DNS Suffix . : localdomain
IP Address. . . . . : 192.168.254.128
Subnet Mask . . . . . : 255.255.255.0
Default Gateway . . . . . : 192.168.254.2

IP Address AND Subnet Mask = Host Address

直接送信できないIP パケットはこのIP addressに投げる

Windowsでは ipconfig (2)

C:> ipconfig /all ← optionとして /all をつけた

```
Windows IP Configuration
        Host Name . . . . . . . . . . . . : umeko
        Primary Dns Suffix  . . . . . . . :
        Node Type . . . . . . . . . . . . : Unknown
        IP Routing Enabled. . . . . . . . : No
        WINS Proxy Enabled. . . . . . . . : No
        DNS Suffix Search List. . . . . . : localdomain

Ethernet adapter ローカル エリア接続:

        Connection-specific DNS Suffix  . : localdomain
        Description . . . . . . . . . . . : AMD PCNet Adapter
        Physical Address. . . . . . . . . : 00-0C-29-32-32-F3

        Dhcp Enabled. . . . . . . . . . . : Yes
        Autoconfiguration Enabled . . . . : Yes
        IP Address. . . . . . . . . . . . : 192.168.254.128
        Subnet Mask . . . . . . . . . . . : 255.255.255.0
        Default Gateway . . . . . . . . . : 192.168.254.2
        DHCP Server . . . . . . . . . . . : 192.168.254.254
        DNS Servers . . . . . . . . . . . : 192.168.254.2
        Lease Obtained. . . . . . . . . . : 2008年4月30日 5:21:26
        Lease Expires . . . . . . . . . . : 2008年4月30日 5:51:26
```

→ Host Address = IP Address AND Subnet Mask

← DHCPでIP address が設定された

← DHCPでDNSのIP address も設定された

MacOSではifconfig

自宅無線LANの例

```
$ ifconfig
lo0: flags=8049<UP,LOOPBACK,RUNNING,MULTICAST> mtu 16384
        inet 127.0.0.1 netmask 0xff000000
        inet6 ::1 prefixlen 128
        inet6 fe80::1%lo0 prefixlen 64 scopeid 0x1
        inet6 fde8:c2b7:5f0b:8cc:129a:ddff:feab:1238 prefixlen 128
gif0: flags=8010<POINTOPOINT,MULTICAST> mtu 1280
stf0: flags=0<> mtu 1280
en0: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
        ether 10:9a:dd:ab:12:38
        inet6 fe80::129a:ddff:feab:1238%en0 prefixlen 64 scopeid 0x4
        inet6 2001:c90:941:20f0:129a:ddff:feab:1238 prefixlen 64 autoconf
        inet 192.168.1.18 netmask 0xffffff00 broadcast 192.168.1.255
        media: autoselect
        status: active
utun0: flags=8051<UP,POINTOPOINT,RUNNING,MULTICAST> mtu 1500
        inet6 fe80::129a:ddff:feab:1238%utun0 prefixlen 64 scopeid 0x5
        inet6 fd00:6587:52d7:33:129a:ddff:feab:1238 prefixlen 64
```

loopback

network

メディア種別

無線ethernet

自hostのIPv4 address

リモートホストのIPアドレス

DNSサーバに名前解決の問い合わせを要求

Windows/MacOS

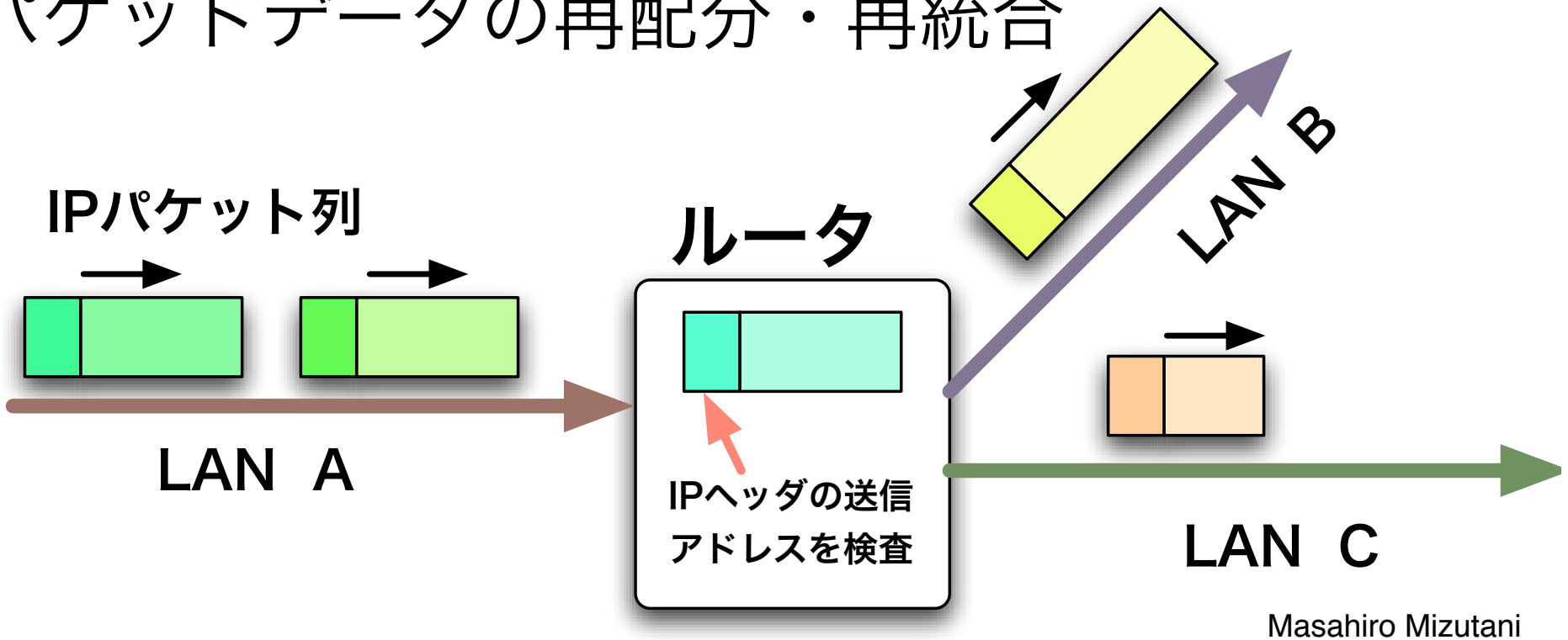
```
% nslookup www.nhk.or.jp
Server:      8.8.8.8
Address:    8.8.8.8#53

Non-authoritative answer:
Name:  www.nhk.or.jp
Address: 202.214.202.101
```

MacOS

```
% host www.nhk.or.jp
www.nhk.or.jp has address 202.214.202.101
www.nhk.or.jp mail is handled by 10 iron.nhk.or.jp.
```

ルータ（router）

- LAN間接続に必要な役割をするコンピュータ
- 宛先アドレスによる経路選択機能（経路制御）
  - LANごとに異なるパケットフレーム(IPパケットを載せている)の差異を調整
  - パケットデータの再配分・再統合



1秒間のIPパケット数

- IPパケット長が200byeとする。
- 8byteヘッダのUDPに乗せてReal-Time通信プロトコルRTP(ヘッダ12byte)を使うとする。
- ヘッダを覗いた実際の音声データ160byte
- 64kb/secのPCMデジタル符号化で160byteのデータの発生に20msecかかる。
- 1秒間にIPパケットを50個。双方向通信として100個が流れている。
- 10分間のYouTube動画のストリーミングを発生させるために必要なパケット数は通常の品質で10万パケットといわれている。

経路制御(Routing Control)

- IPの機能で最重要の役割
- ルータが各IPパケットの経路をその都度選択
  - 各ルータは経路制御表を持っている
  - 取り込んだIPパケットのヘッダを読んで
  - 経路制御表を参照しながら
  - どの経路にパケットを中継するかを決定
- 各ルータが経路制御を繰り返す
  - 最終的に目的のネットワークに到達
  - そのネットワーク内の目的のホストに届く

階層的ネットワークの構造

自律システム(AS) 16bitの**AS番号**(0～65535の番号空間)が割り当てられている

単一の管理者によって管理されるルータとネットワークの集合体

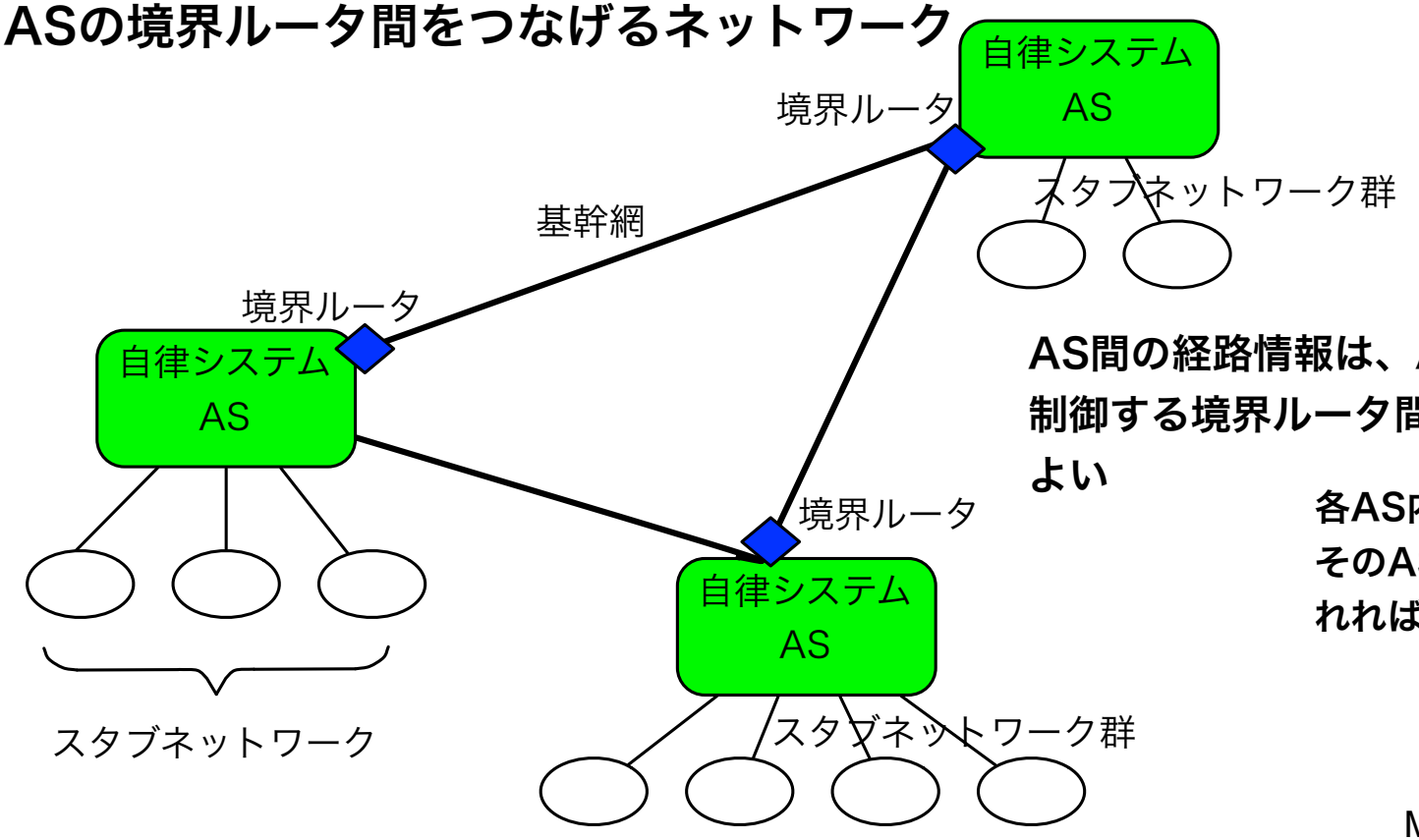
スタブネットワーク

ASの中にあって、それにつながっているサブネットワーク

同じAS番号を持つ

インターネットの基幹網（backbone）

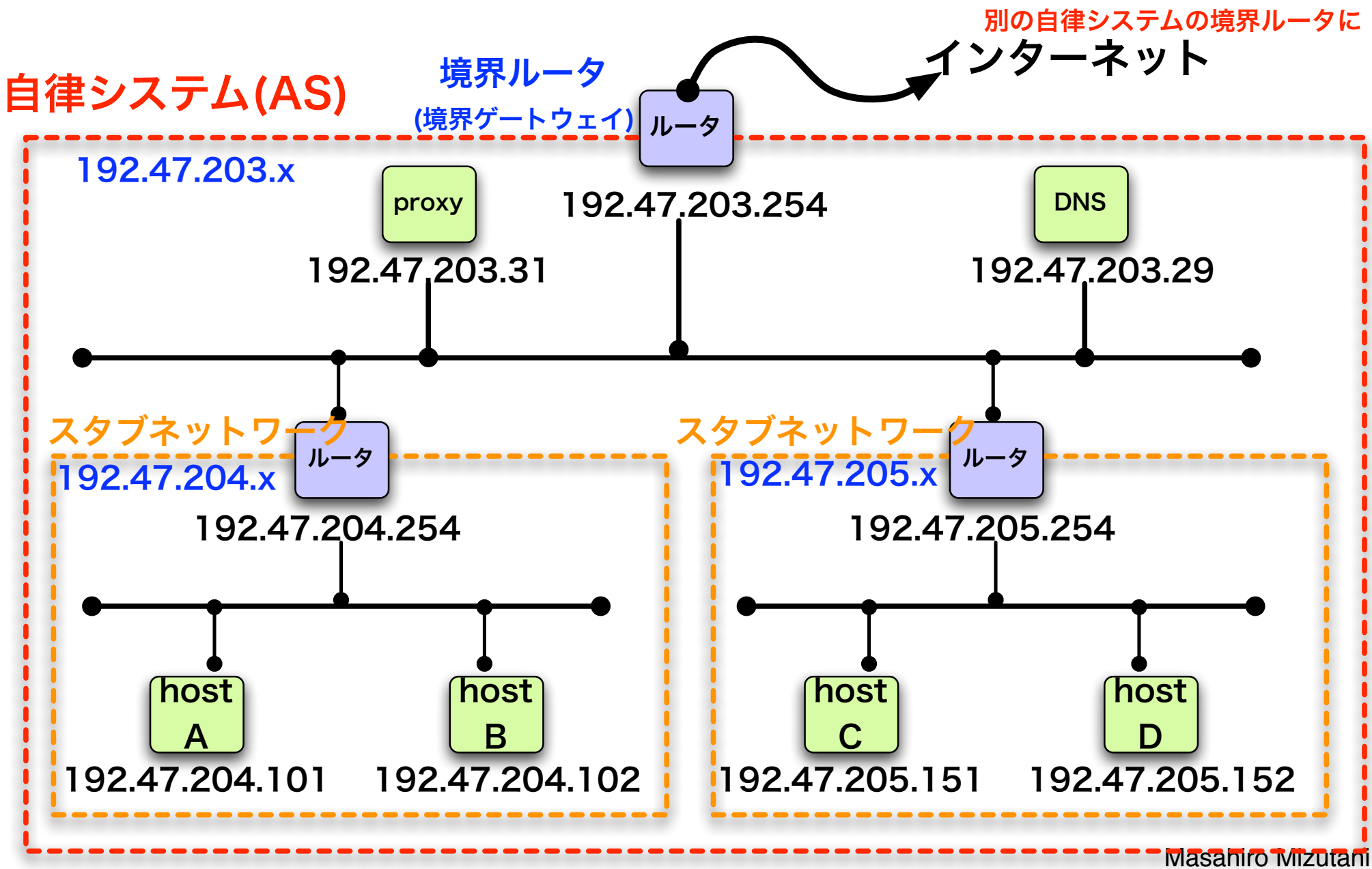
ASの境界ルータ間をつなげるネットワーク



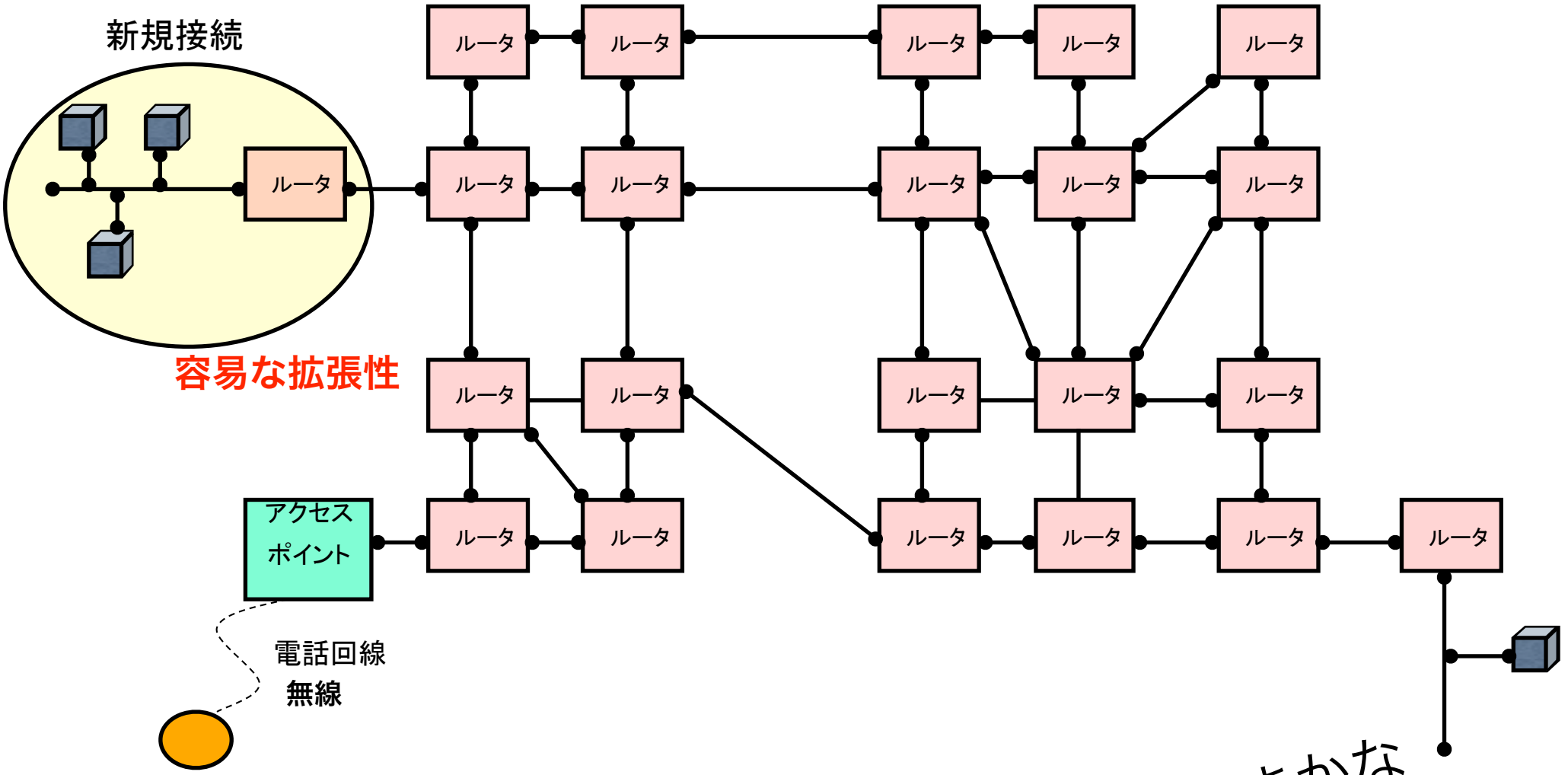
AS間の経路情報は、AS間の経路を制御する境界ルータ間で交換すればよい

各AS内の経路情報は、そのAS内だけで交換されればよい

AS内のルータとネットワーク構成



インターネット基幹と経路制御



ASルータ同士のネットワークがインターネットの骨組み

各ASルータは内部にあるスタブネットワーク群の境界に置かれている

経路制御時に解決すべきこと

動的な経路制御表の更新

ネットワーク状況に応じたルータの最小広域木(minimum spanning tree)の維持

各種の経路制御プロトコル（アルゴリズム）

障害対策

パケットの寿命と廃棄

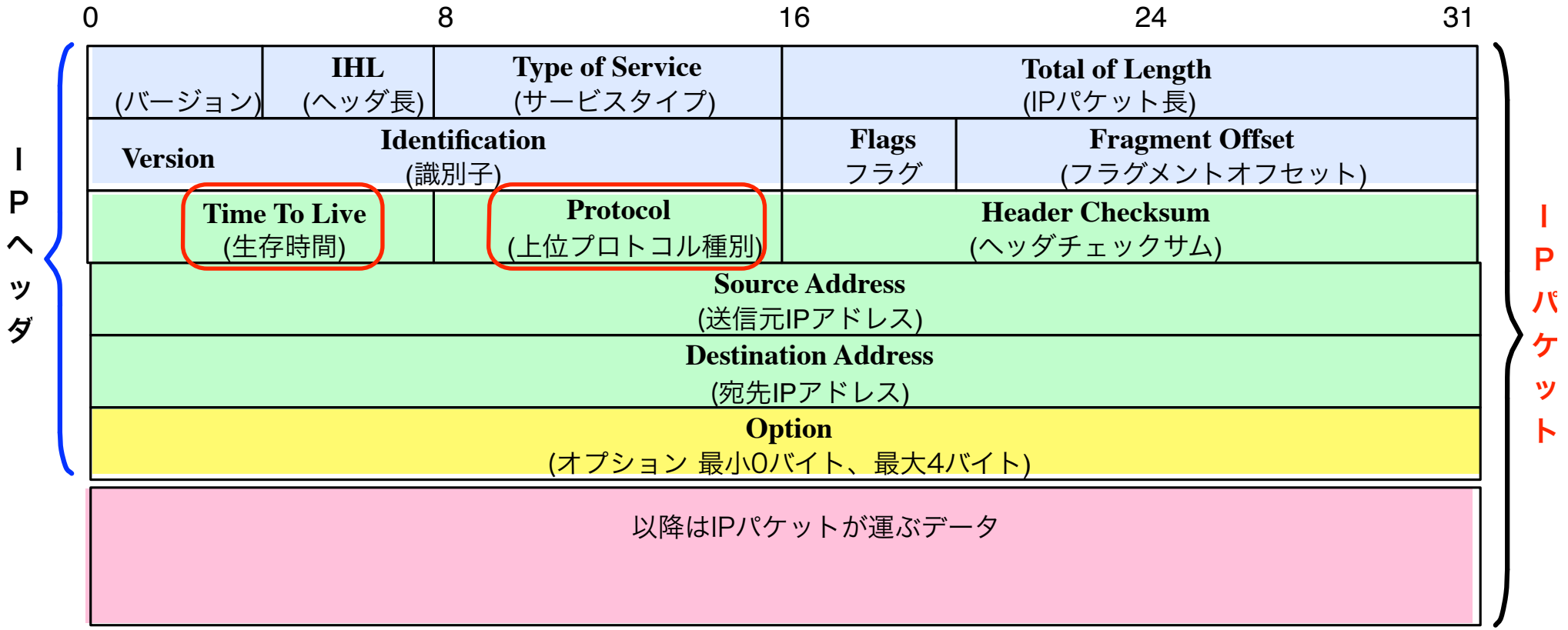
ICMPを使う

通信量制御

送信IPパケットを制御する（渋滞の回避）

IPの上位プロトコル(TCPなど）を使う

IPヘッダの構造(2)



TTL: IPパケットがインターネット上で生存できる事が許されるルータを経由できる数。ルータを経由するごとにTTL値を1減らし、この値が0になるとそのパケットは破棄される

Protocol: IPパケットで使用される次（上位）のレベルのプロトコル。 ICMP, IGMP, TCP UDP, IPv6 などがある。

代表的なネットワークコマンド

同じ名称のコマンドでオプションや表示形式は異なる

MacOS	Windows	用　　途
ifconfig	ipconfig	自ホストのTCP/IP構成情報を取得する。
ping	ping	ホスト（またはIPアドレス）からの応答を確認。自ホストの設定の正常さの確認にも。
nslookup	nslookup	DNSサーバーに名前解決に関する問合せをおこなう。オプションで各種のDNS情報も。
host		簡易版nslookup
traceroute	tracert	宛先ホストまでのネットワーク経路（中継ルータ）をリストする。
arp	arp	IPアドレスとMACアドレスの対応表を得る。
netstat	netstat	各種ネットワーク接続とその統計情報（経路表を含む）を返す。
route	route	経路表の状態や変更、追加、削除を行う。

オプションやヘルプ

MacOS　コマンド `--help`　または　`man` コマンド

Windows　コマンド `-help`

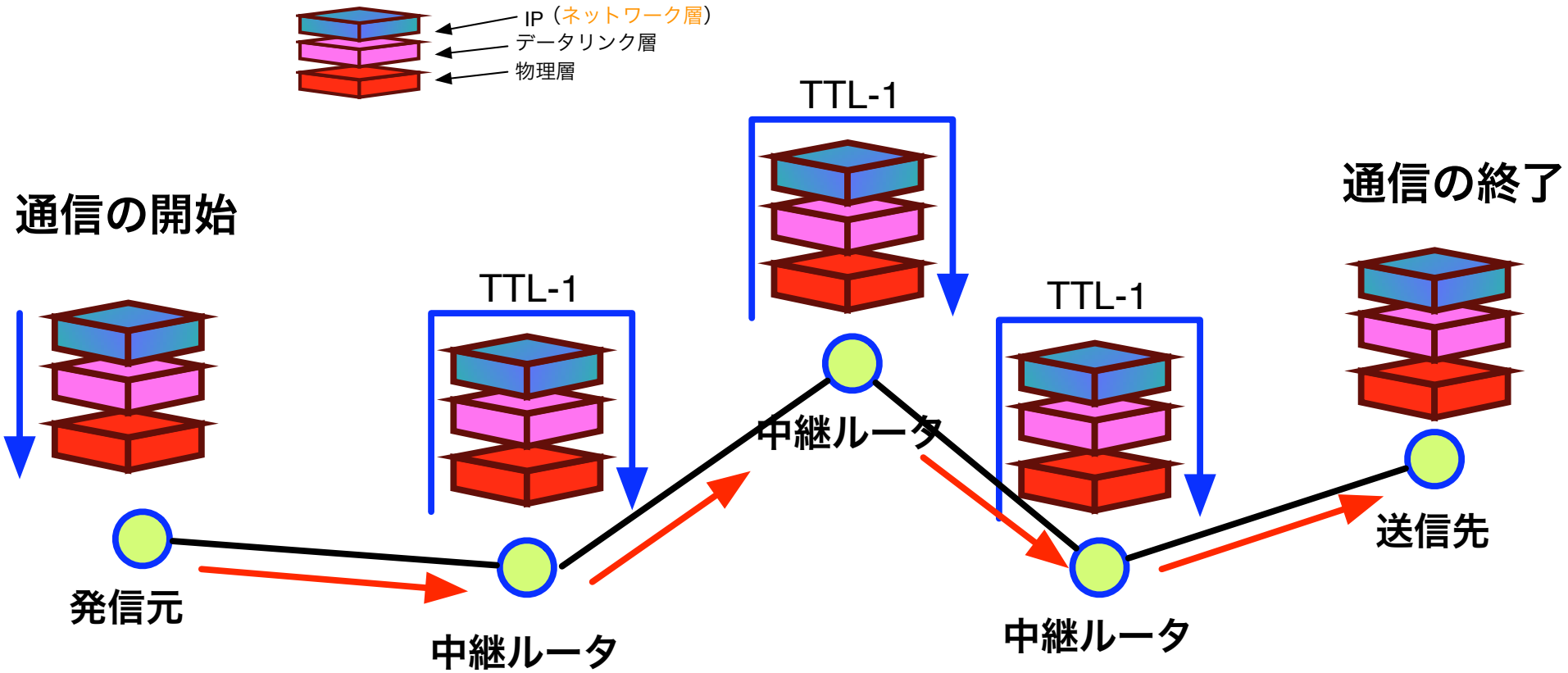
More about IPパケット転送

バケツリレー方式

中継ルータは自分宛パケットでなければ、TTLを1減らしてして経路制御表にしたがって次に送る

転送判断はネットワーク層で

転送はデータリンク層で(ARPを使う)



IPを支える制御プロトコル ICMP

中継ルータにおいて正常にIPパケット配送されない事態がしばしば発生

ICMP(Internet Control Message Protocol)
この不具合を始点ホストに報告する機構

パケットを廃棄する

エコー要求, エコー応答
終点までの到達性を確認，RTT(Round Trip Time)の計測

終点到達不能
ノードがIPパケットを配送できない場合。経路情報がなくて到達不能

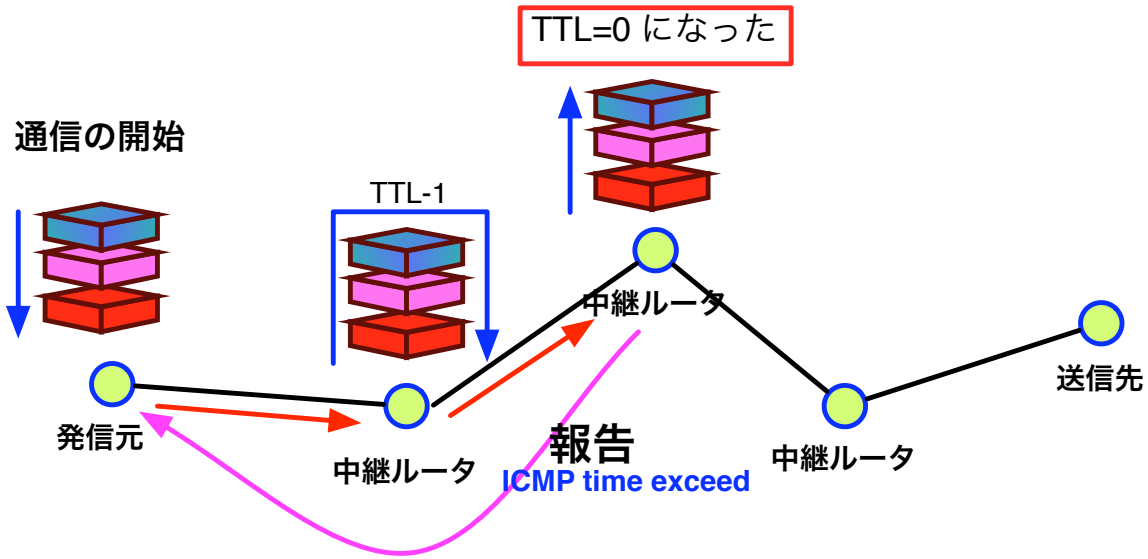
始点抑制
輻輳状態に陥っている場合

時間超過
TTL=0になった場合

IPパケットの寿命(終焉)

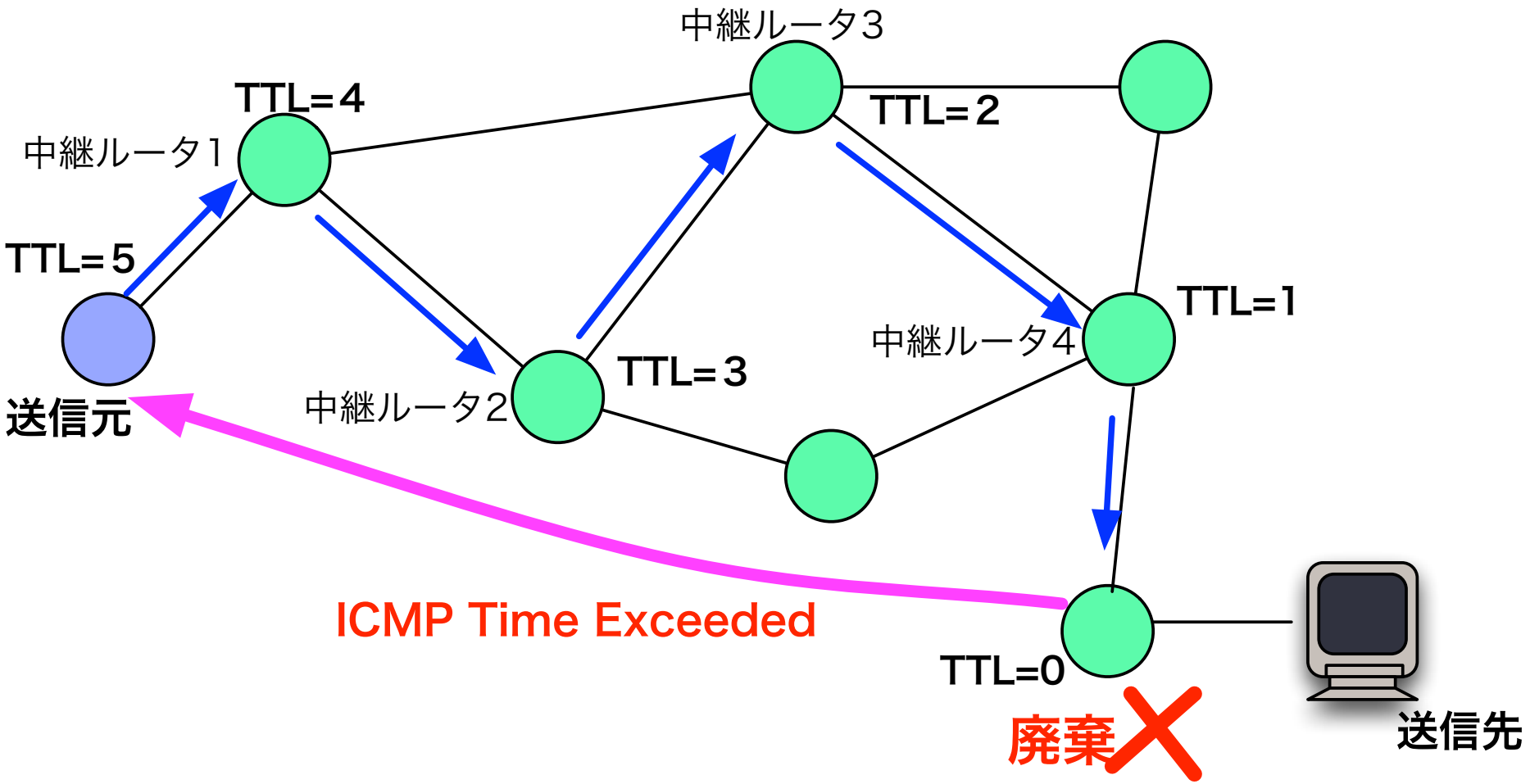
- 宛先IP addrに対する経路情報を中継ルータが持っていない（宛先にたどり着けない）
- 宛先に到着しても目的ポートが開いてない（受信者が受け取れない）
- TTL値が0になってしまった

⇒ ICMP エラー通知＋トラブルシュートの仕組み



ICMPエラーの発生

何らかの不具合でIPパケットが廃棄されるとき始点にエラーメッセージを送り、その状況を報告する



ネットワークの到達可能性の検証

プログラム　**ping**　　**MacOS/Windows**

ICMP(Internet Control Message Protocol)のエコー要求パケット
を目的ホストに送る
相手先ホストはエコー応答パケットを返す

```
$ ping www.yahoo.co.jp
PING www.ya.gl.yahoo.co.jp (203.216.235.222): 56 data bytes
64 bytes from 203.216.235.222: icmp_seq=0 ttl=56 time=8.815 ms
64 bytes from 203.216.235.222: icmp_seq=1 ttl=56 time=10.106 ms
^C [Ctl-cで強制終了]
```

```
$ ping info.tsuda.ac.jp    学外からはinfo.tsuda.ac.jpにはunreachable
PING info.tsuda.ac.jp (133.99.161.9): 56 data bytes
Request timeout for icmp_seq 0
Request timeout for icmp_seq 1
^C [Ctl-cで強制終了]
```

IP経路の確認

プログラム　traceroute　MacOS

tracert　Windows

・IPヘッダにあるTTL(Time To Live)フィールドを利用(最大255)。

・ルータをホップ（通過）する毎に1減らされる。TTL値=0になると、

そのIPデータグラムは廃棄され、ICMPはTTL超過を認識

・ICMP Time Exceededエラーを送信ホストに知らせる。

traceroute/tracertの仕組み

1) 1ホップ目のルータからエラーが返る（3回繰り返す）

2) 次にTTL=2とすれば2ホップ目のルータからエラーが返る

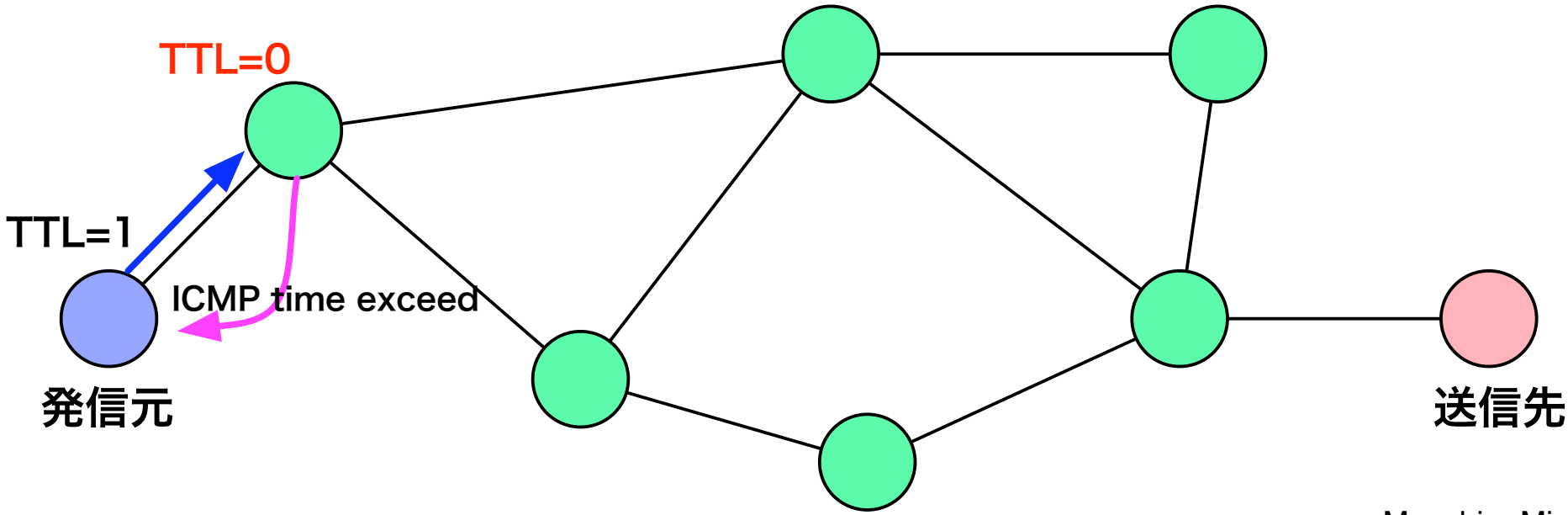
……

n) TTL=nとしてnホップ目のルータからエラーが返る

ICMPの利用

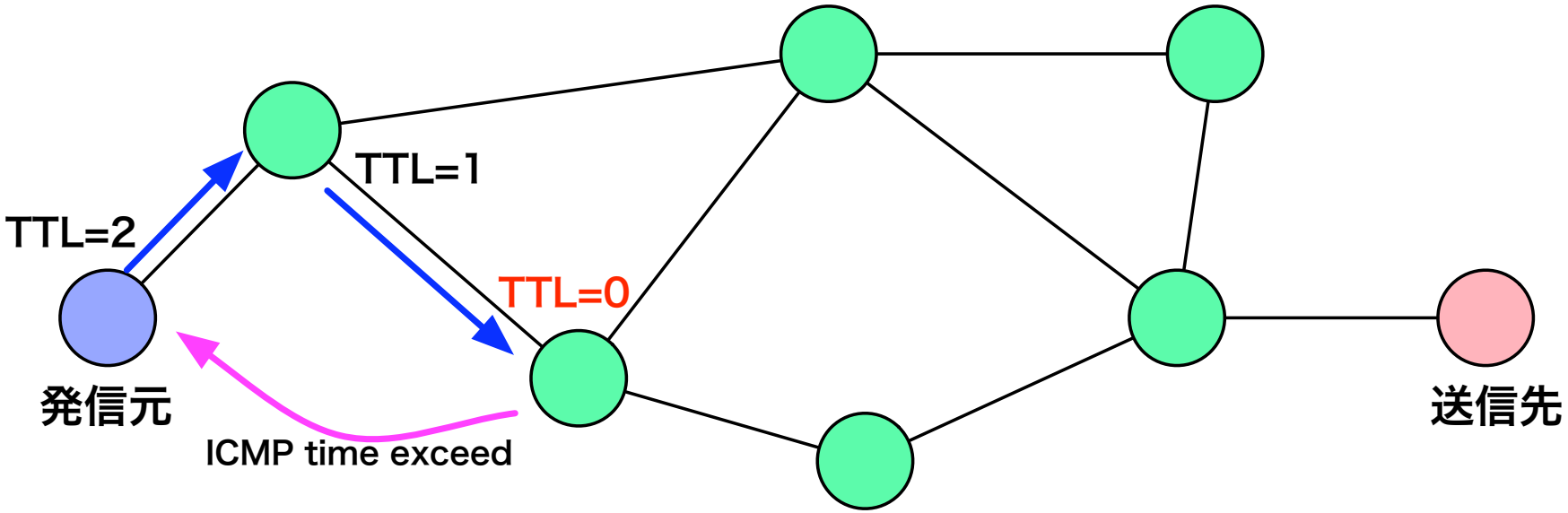
traceroute(1)

- 発信元から送信先までUDPデータグラムを送る
  1. TTLを1にして送信
  2. 最初の中継ルータはTTL=0のパケットを受信してICMP time exceedを返す
  3. 1ホップ先が判明



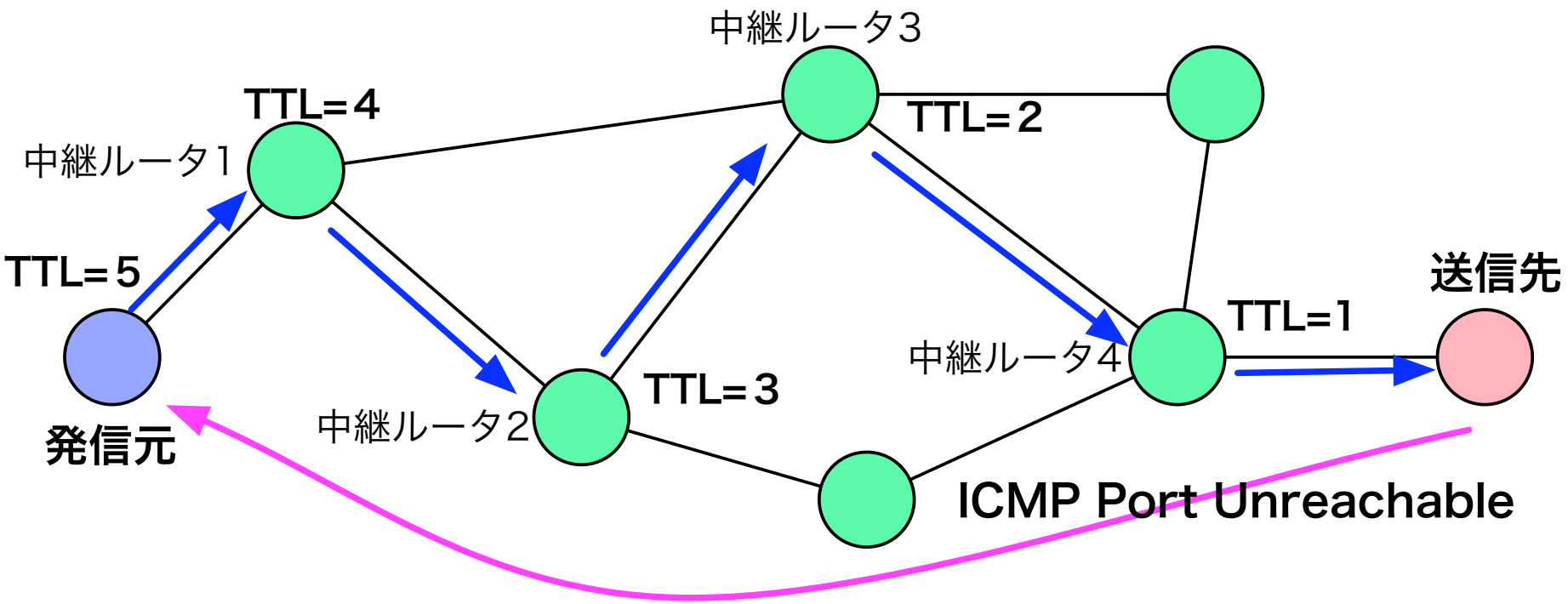
traceroute(2)

- 発信元はTTLを1増やし2にして再送信
  1. 最初の中継ルータはTTLを1つ減らして次の中継地へ
  2. 2番目でTTL=0となり、ICMP time exceedを送信して2つめの中継ルータが判明

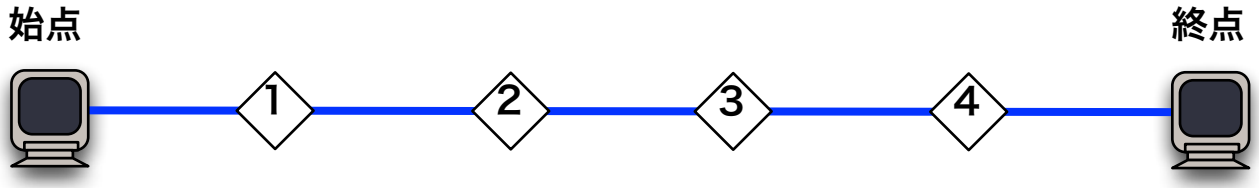


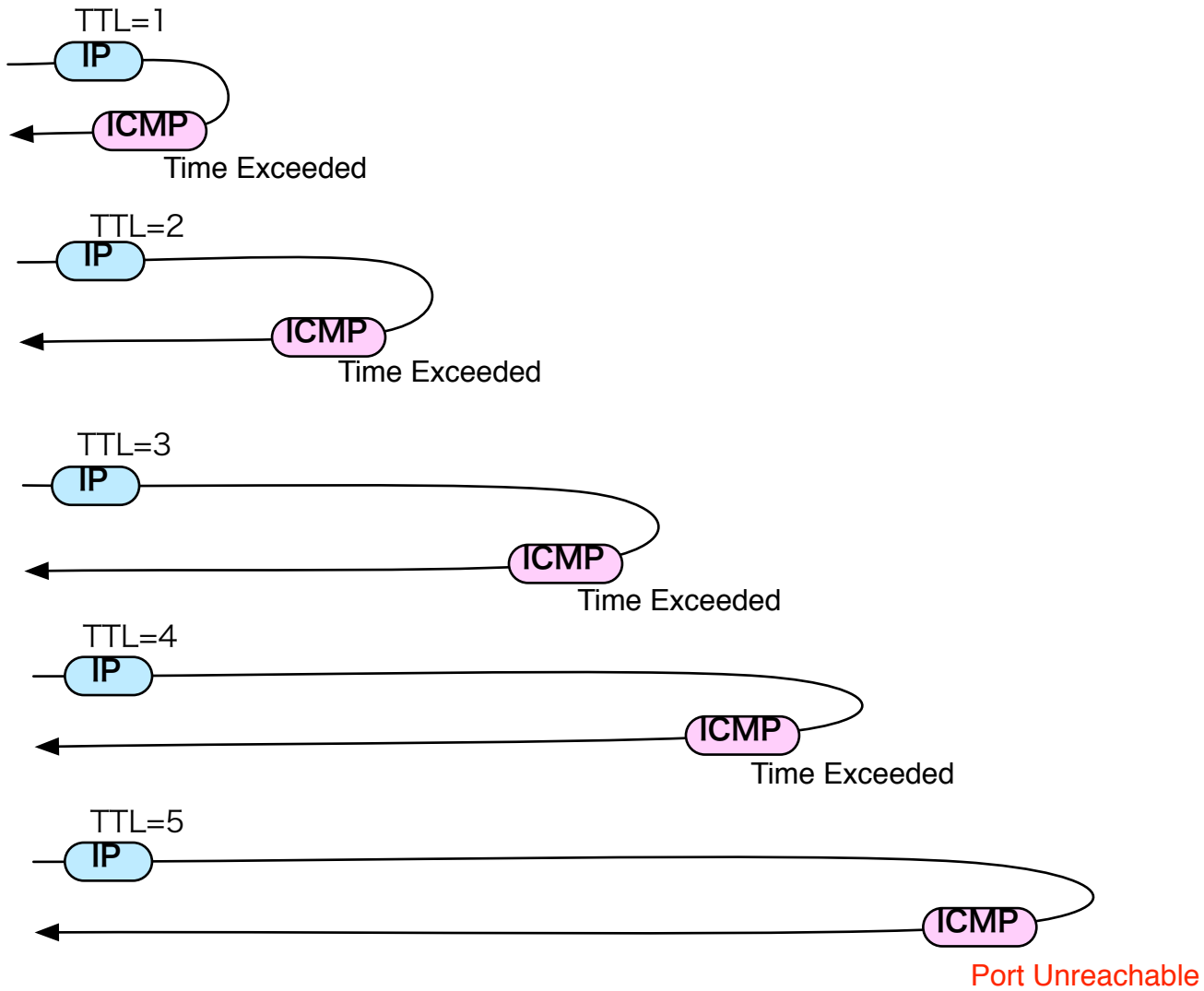
traceroute(3)

- これを繰り返す
  1. 宛先に届いても、通常は受け取れないポート番号に設定されているため、ICMP Port Unreachableを発信元へ
  2. これで送信先に届いたことが分かる



tracerouteの仕組み





ところで、経路制御は
どうやって行うの？

経路制御のための課題

- End-to-Endの**到達性**を保証する
- **最短経路**の自動発見
- **動的な対応**の必要性
  - 障害が生じたときの迂回路の発見
  - トラフィックの分散
  - 管理ポリシーとの調和

経路制御プロトコル

- 経路情報
  - 隣接ルータを認識
  - トポロジー情報を交換
  - リンク情報を交換
- 目的に応じて複数の経路制御プロトコル
  - 距離ベクトル型（RIP:Routing Information Protocol）
  - リンク状態型（OSPF:Open Shortest Path First）
  - パスベクトル型（BGP:Border Gateway Protocol）

IP経路制御(routing)

各routerは制御表に基づいて単純に判断

宛先が同一ネットワーク→直接転送
宛先が別ネットワーク→defaultルータに転送

制御表(routing table)の構造

宛先addr	次hop先	方向(InterFace)
宛先addr	次hop先	方向(InterFace)
宛先addr	次hop先	方向(InterFace)
宛先addr	次hop先	方向(InterFace)
…	…	…

経路制御表の利用

RouterはIPパケットを受け取ると経路制御表を検索

Routerは次の順番でパケット処理

1. **宛先IP addrに適合するエントリ発見→データ転送**
   宛先とPtoP接続してる場合
2. **宛先Network addrに適合するエントリ発見→データ転送**
   宛先がlocalなnetworkに接続してる場合
3. **defaultエントリ発見→データ転送**
   他のnetworkのrouterへ転送
4. **全てに該当しない場合→エラーを返す**
   host到達不可、network到達不可

経路制御表の取得

プログラム　**netstat**　　**MacOS/Windows**

MacOS/Windowsともに **route** があるが、誤って利用すると経路情報を書き換えてしまう可能性があるのでここでは netstat を使う

% **netstat** -r　← -r オプションを付けて経路制御表情報を取得

Routing tables

パケットの宛先

自ホストが接続しているネットワークのルータアドレス

宛先に向いている通信デバイス

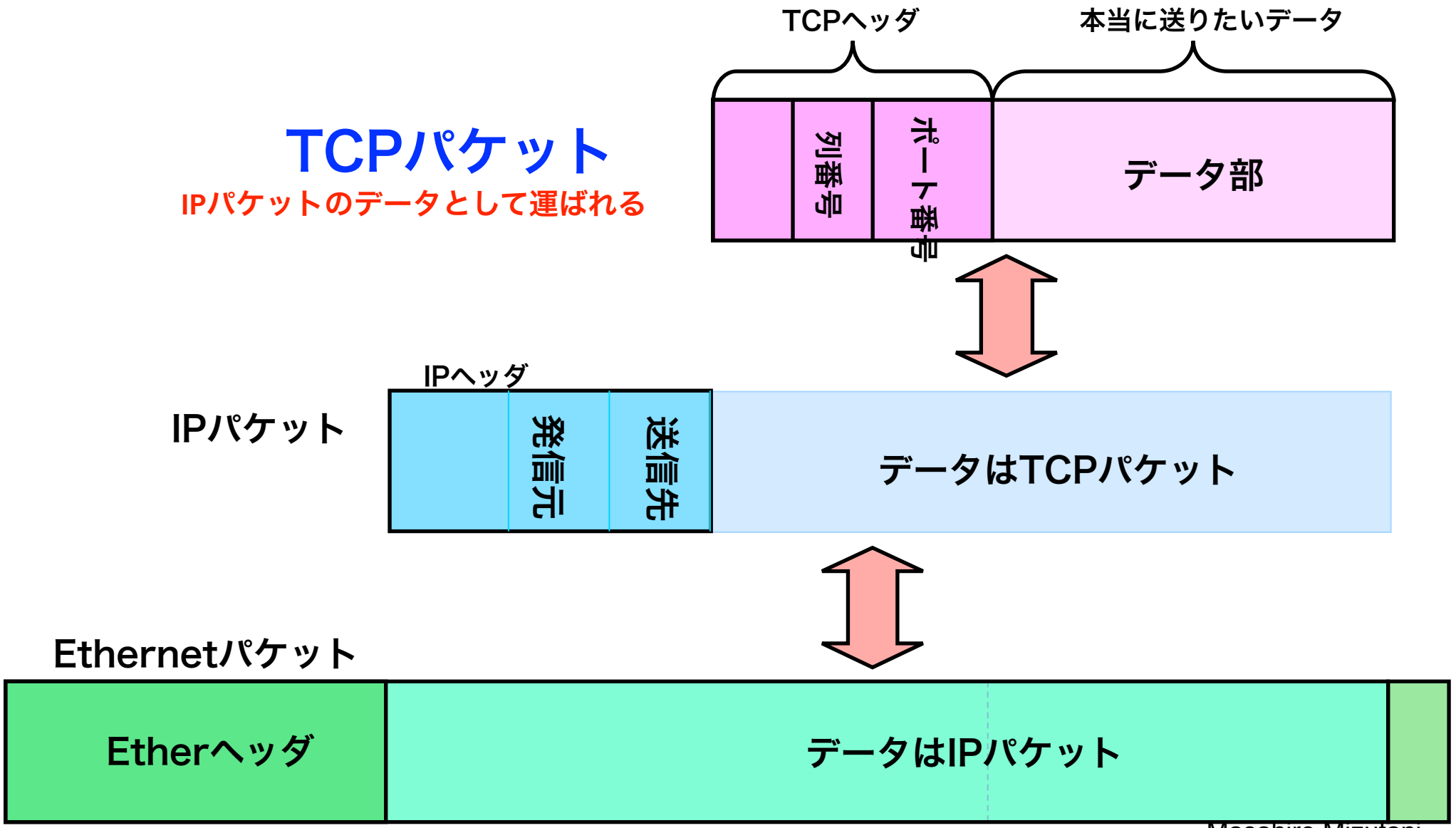
現経路制御表にある経路が失効するまでの時間(秒)

Internet:

Destination	Gateway	Flags	Refs	Use	Netif	Expire
default	133.99.135.254	UGSc	4	6	en0	
10.37.129/24	link#8	UCS	0	0	en2	
10.37.129.2	127.0.0.1	UHS	0	3	lo0	
10.211.55/24	link#9	UCS	0	0	en3	
10.211.55.4	127.0.0.1	UHS	0	3	lo0	
127	127.0.0.1	UCS	0	0	lo0	
127.0.0.1	127.0.0.1	UH	18	7029	lo0	
133.99.135/24	link#2	UCS	2	0	en0	
133.99.135.42	127.0.0.1	UHS	0	3	lo0	
133.99.135.251	0:17:f2:93:f2:6c	UHLW	1	109148	en0	663
133.99.135.254	0:19:30:0:50:0	UHLW	5	0	en0	1186
169.254	link#2	UCS	0	0	en0	

じゃぁ、通信量制御
はどうやって行うの？

TCPパケットの構造


TCPヘッダ
本当に送りたいデータ
TCPパケット
IPパケットのデータとして運ばれる
列番号
ポート番号
データ部
IPヘッダ
IPパケット
発信元
送信先
データはTCPパケット
Ethernetパケット
Etherヘッダ
データはIPパケット

TCP（伝送制御プロトコル）

- データがどのアプリで使われるかを知らせる
  - ポート番号
- 双方向の信頼性のある接続を実現
  - パケットをシークエンス番号順に並べ替える
  - データの破壊・損失重複・順序誤りがないことを保証
    - パケット損失・誤り時は送信元に再送要求
- 通信状況に応じて通信量を制御
  - 輻輳の防止

TCPの状態構造と動作は複雑

設計思想

通信品質に関する大きな転換

ネットワーク経路でなく**端点に責任を持たせる**

端点の高い能力が必要（高速で確実な処理）

経路途中での失敗を両端点がリカバーする

昔の電話網は「ボロい電話機」と「立派な電話局施設」だった

役割

セッションの形で**1対1(多)通信**を実現

コネクション指向（接続確認）

信頼性ある**双方向**通信を実現

欠損パケット再送などのエラー検出機能

輻輳回避

代償として速度は遅くなる

輻輳制御は難しい

- 輻輳は悪化していく傾向にある
- IPネットワークには輻輳制御機能がない
  - 端点は遠くのネットワークの状態が分からない
  - 自己責任で推測して送出せねばならない
- IP＋TCPネットワークの登場
  - 端点に高度な機能を要求 ←公衆電話網と違う
    - 窓サイズを増減させて転送制御
    - 送信者の輻輳窓と受信者の窓を調節しあう
      - 段階的通信状態
      - slowスタート、パケットロスで輻輳回避

IPとTCPのまとめ

宛先コンピュータにパケットが到達するための経路制御

IP

パケットでやり取りしてネットワーク回線を共有
パケットは途中多くのルータを通過
迂回路を用意して通信の信頼性に寄与

＋

コンピュータ同士の1対1の信頼性ある双方向通信を実現

TCP

通信の輻輳回避、品質・信頼性の確保
伝走経路でのデータ落ちの際に再送を要求
容易なネットワーク拡張性
端点に機能・責任を持たせる

Masahiro Mizutani

通信基盤としてのTCP/IP

- 様々な機種のコンピュータを接続
- IPアドレスで指定した1対1双方向通信を可能
- TCPによる信頼性のある接続指向の通信

⇒ IPとTCPでパケットの到達性は確保した

では、ホスト間で実際にオシャベリをするには？

（互いに聞こえるけど、相手の話している言葉がわからないことがある）

⇒ 各種のアプリケーションプロトコルを利用

プロトコルは誰が決めるの？

- インターネットのユーザ・技術者自身が提案
  - 実践的かつオープンな仕組み
- 話し合いは、電子メールおよび定例の会合で
  IETF(Internet Engineering Task Force)
- RFC(Request for Comments)などの文書
  - インターネット上で利用される各種プロトコルの仕様を記述・公開