

画像情報特論 (2)

- TCP/IP

- インターネットプロトコル (IP)
- Network-Assisted QoS
- TCP (Transport Control Protocol)
- UDP (User Datagram Protocol)

情報理工学専攻 甲藤二郎

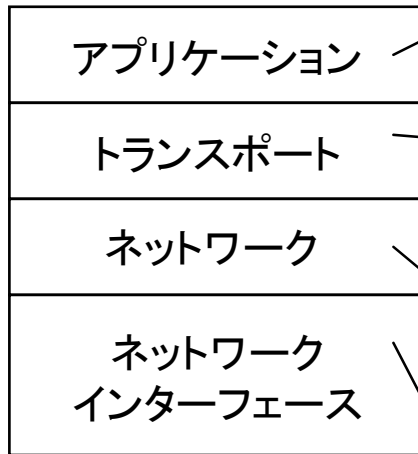
E-Mail: katto@waseda.jp

IP

(Internet Protocol)

プロトコルスタック

プロトコルスタック



アプリケーション

HTTP, RTSP, FTP, Telnet, ...

← RTP: 実時間メディア用途

端末・端末間

TCP: 誤り訂正、順序制御、フロー制御 ... 信頼性重視

UDP: オーバーヘッド少 ... 低遅延、高速性重視

端末・ルータ間、ルータ・ルータ間

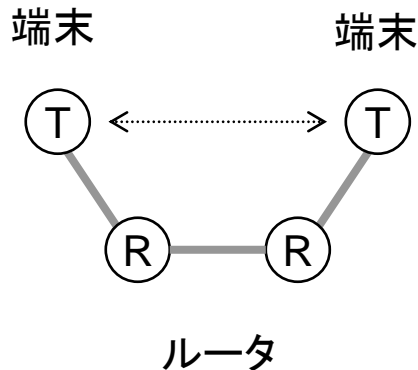
IP: 経路制御、フラグメンテーション

ICMP: エラー通知

IGMP: マルチキャスト (mbone)

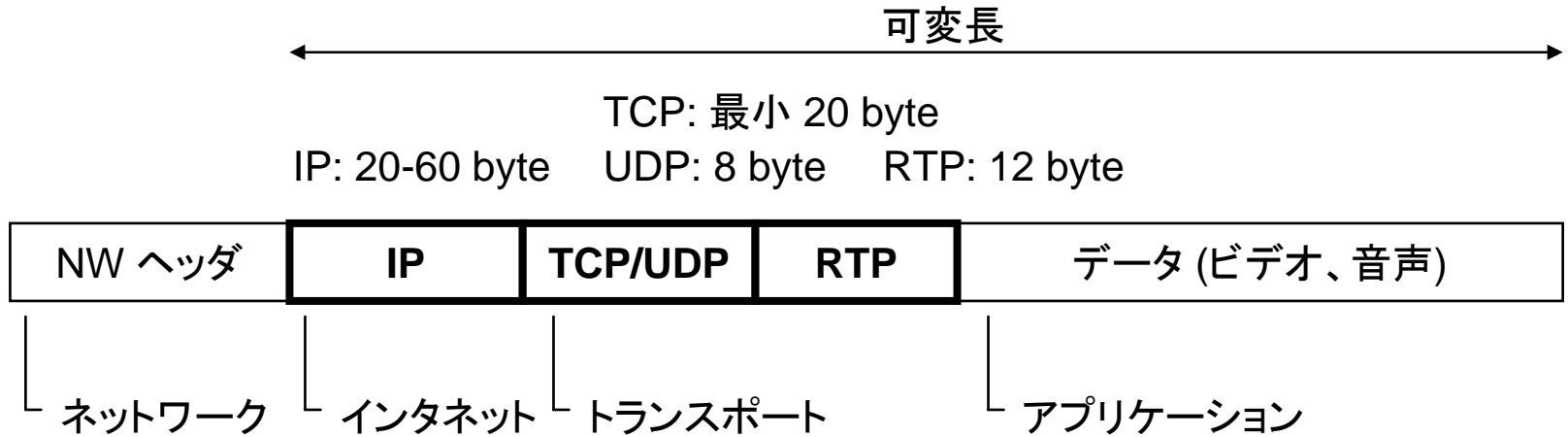
個別リンク

イーサネット, PPP, X.25, ATM, ...

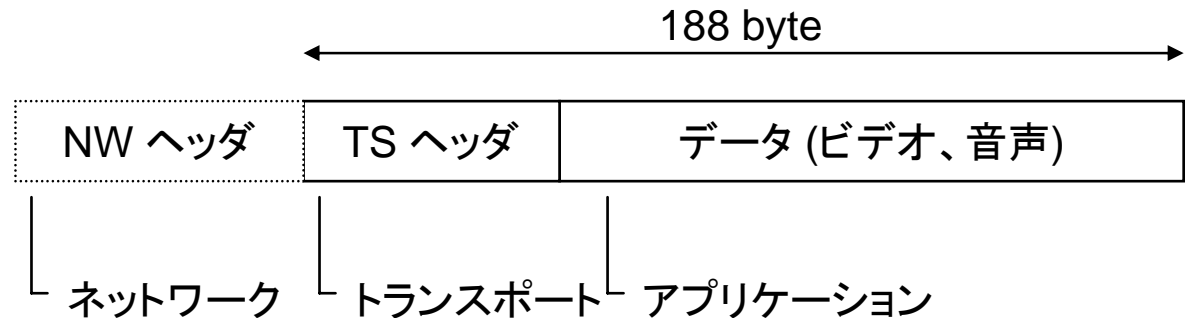


IP データグラム

IP データグラム



cf. MPEG-2 トランスポートストリーム (ITU-T H.222)



IPv4 ヘッダ

IPv4ヘッダ

4 byte

Version	ヘッダ長	サービスタイプ	パケット全長	
フラグメント識別値			フラグ	フラグメントオフセット
TTL (生存時間)	上位プロトコル		ヘッダチェックサム	
送信元 IPアドレス				
受信先 IPアドレス				
(オプション)			(パディング)	
データ				

パケット長:

データのフレーミング (可変長)

TTL:

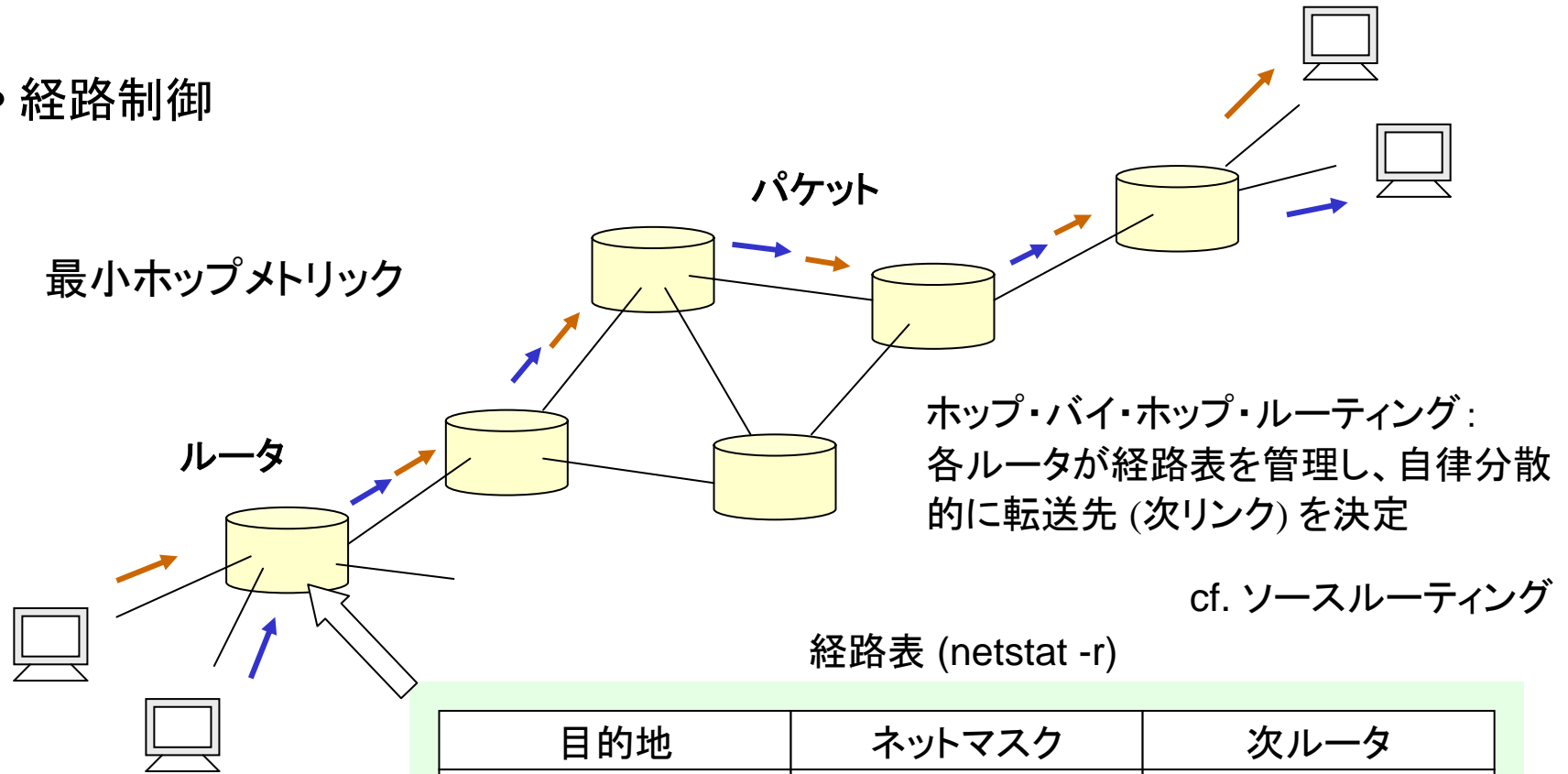
パケット生存時間 (ルータのホップ数)

IPアドレス:

インターネット全体で固有のアドレス。ARP によって
MACアドレスに変換される (Ethernet の場合)

IP の機能

- 経路制御

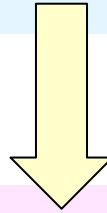


経路表 (netstat -r)

目的地	ネットマスク	次ルータ
133.9.2.x	255.255.255.192	133.9.1.a
133.9.3.x	255.255.255.192	133.9.1.b
133.9.4.x	255.255.255.192	133.9.1.c
.....		
default	0.0.0.0	133.9.1.d

IPの欠点

- 蓄積交換 (store and forward) 故に、パケット転送時間の増大 (**delay**)、転送時間の揺らぎ (**jitter**)、パケット廃棄の発生 (**packet loss**) 等の問題は避けられない。
- パケットの到着順序が逆転することもある (順序制御)。

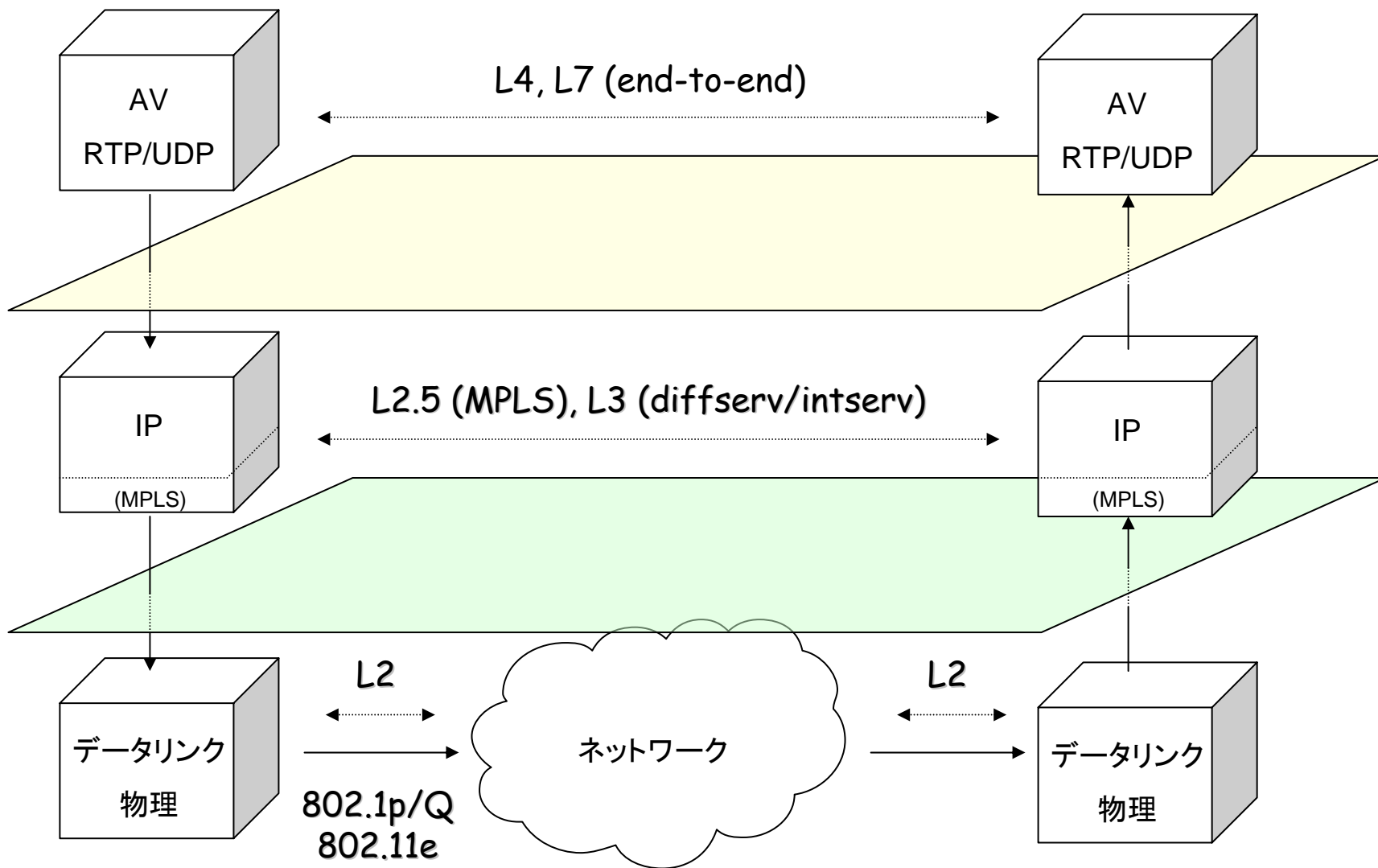


インターネットでもある程度の品質保証 (QoS 保証) を実現したい。
→ Network-Assisted QoS、および End-to-End QoS

Network Assisted QoS

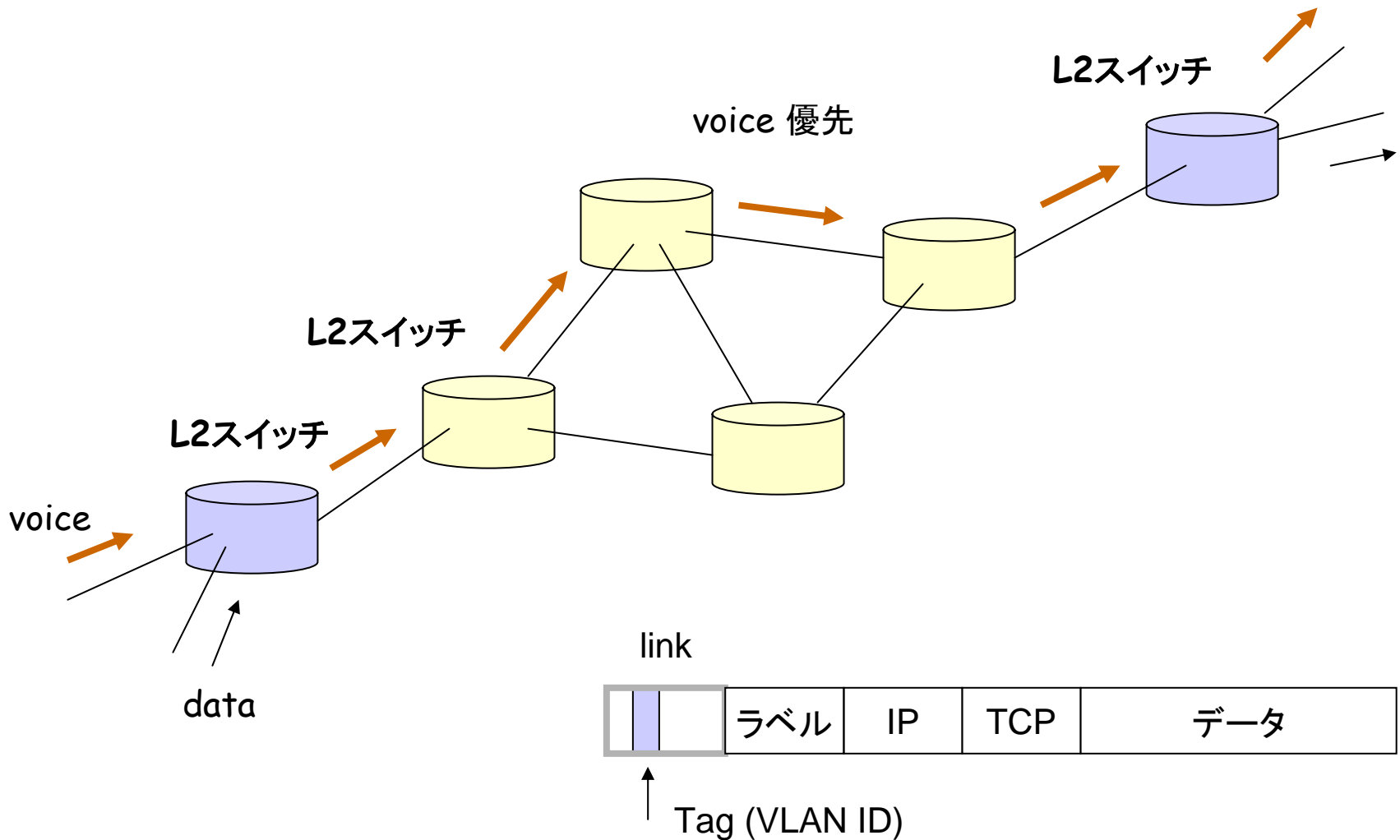
MPLS、Diffserv、トラフィックシェイピング、RSVP

インターネットQoS



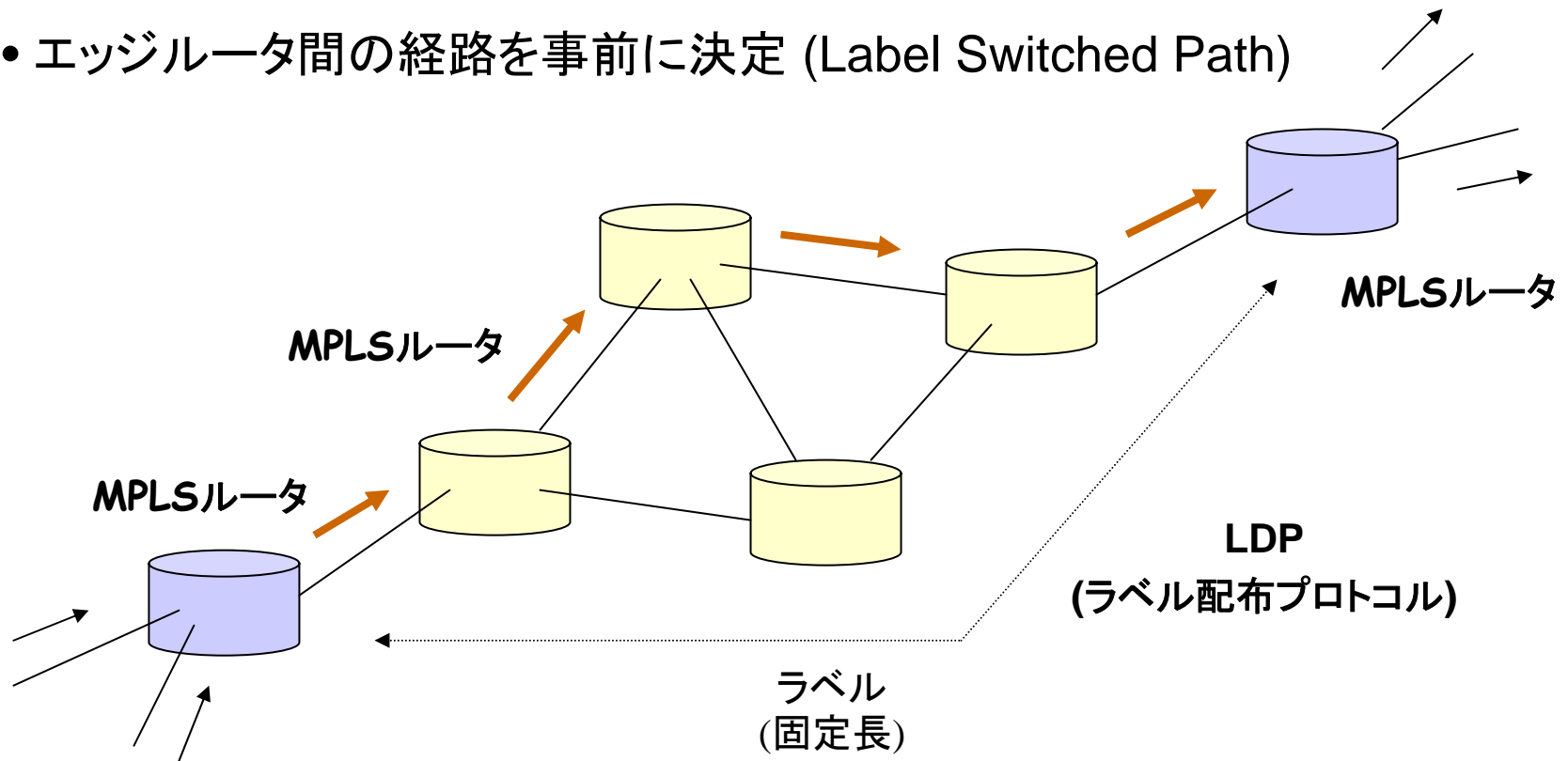
IEEE 802.1p/Q

- Tag を用いたリンク毎の優先制御



MPLS (ラベル・スイッチング)

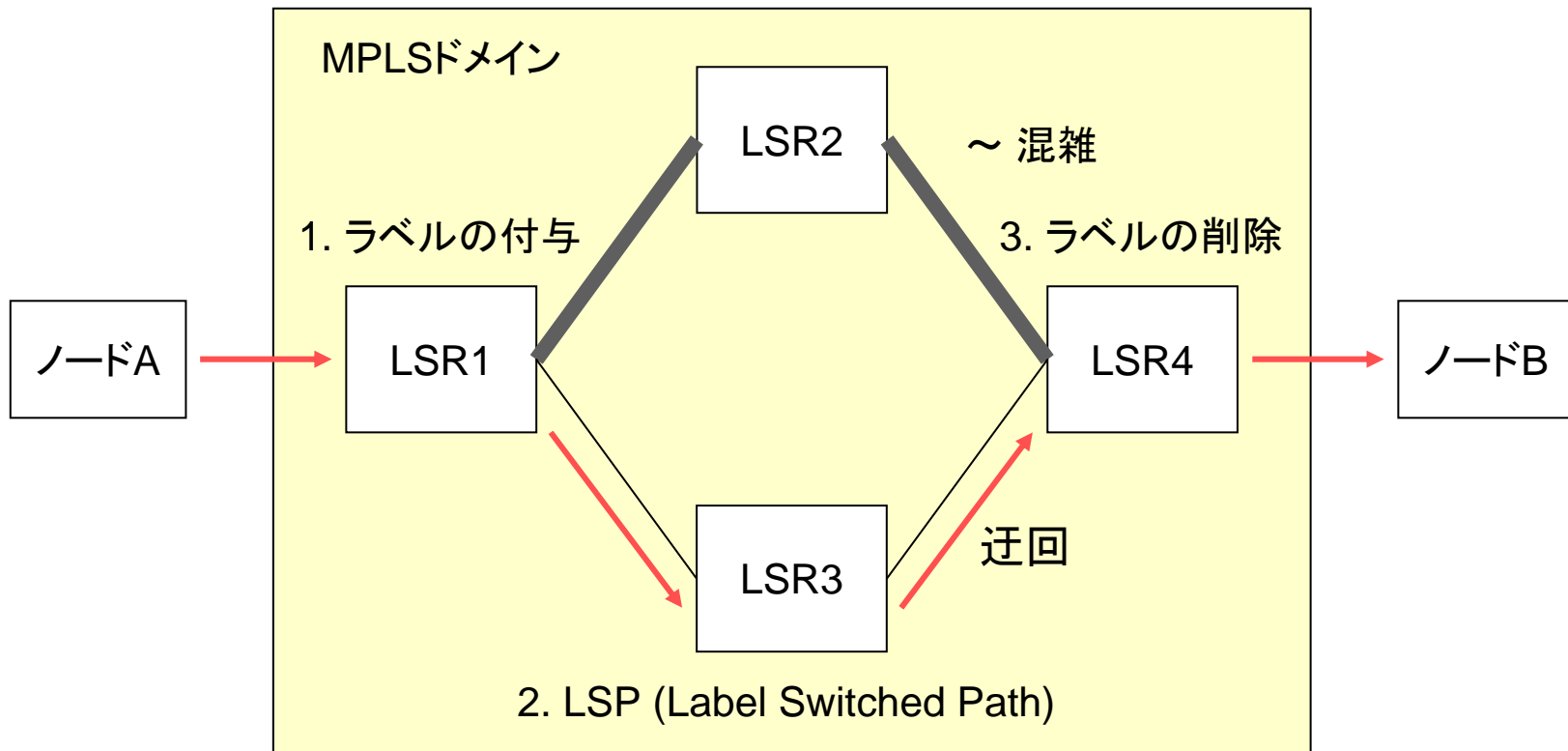
- 固定長ラベルによるハードウェアスイッチング
- エッジルータ間の経路を事前に決定 (Label Switched Path)



CoS (class of service)

MPLSのトラヒック・エンジニアリング

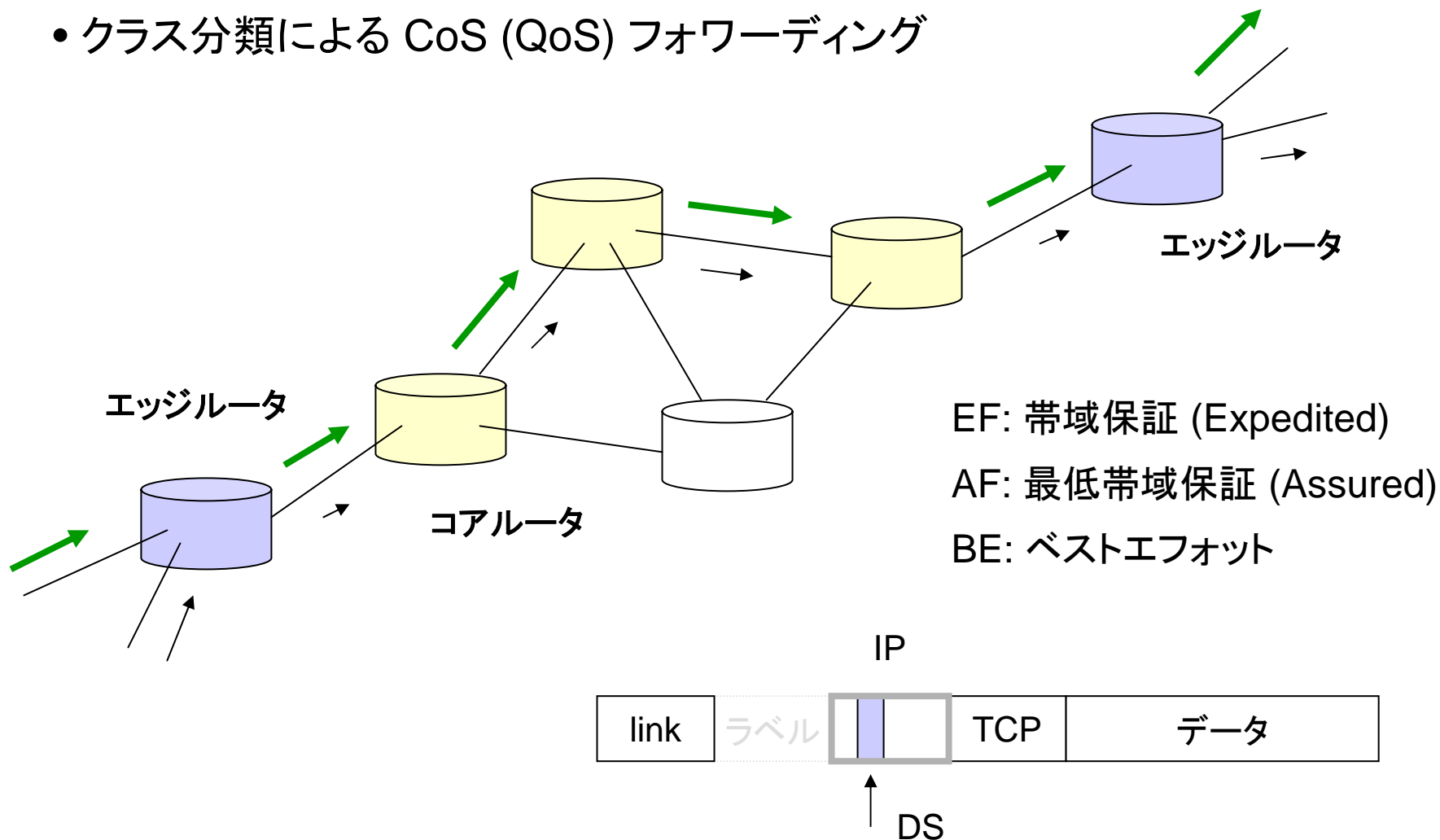
- 最小ホップ数以外のメトリックの活用



トラヒックエンジニアリング：負荷分散、高速迂回、...

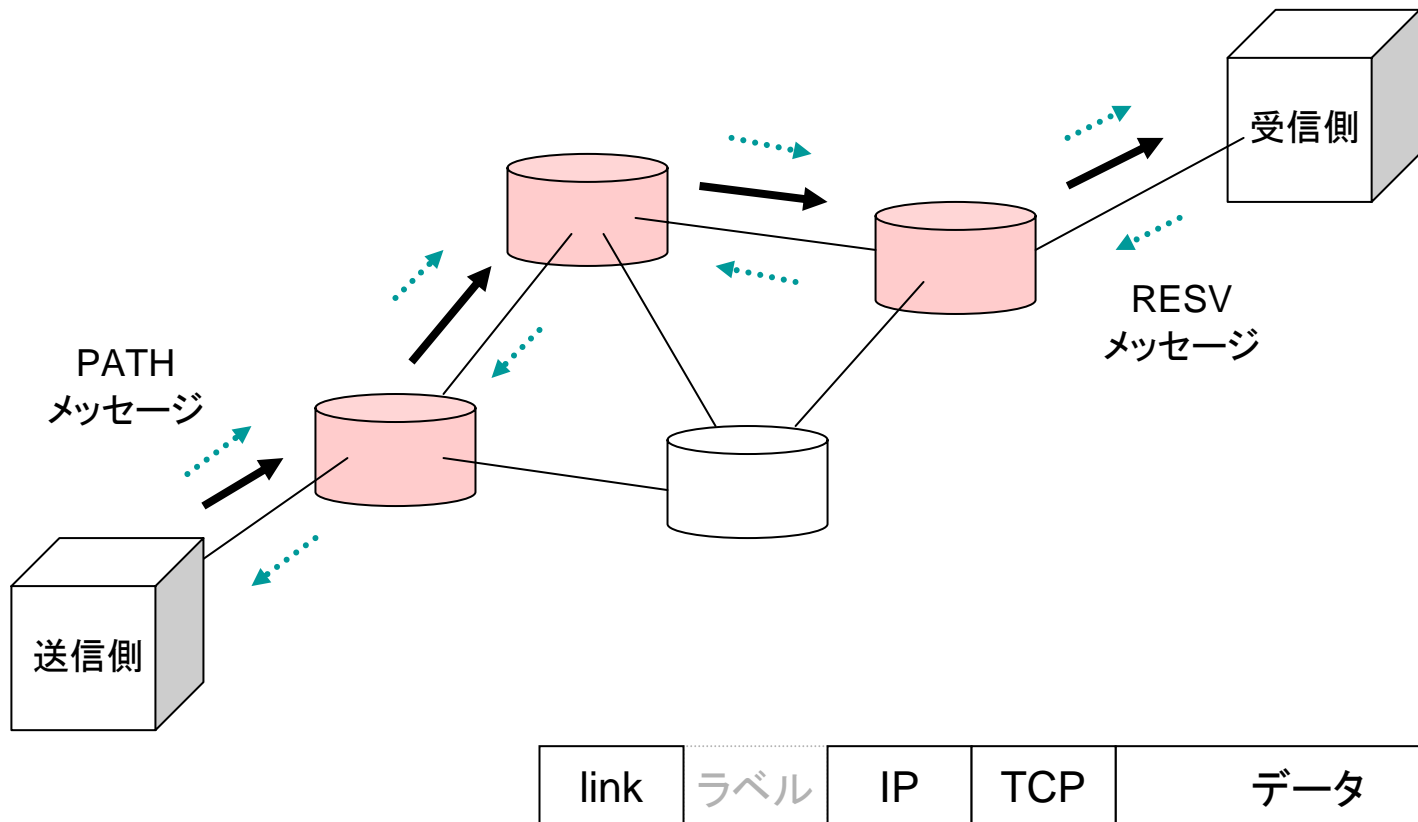
Diffserv (differentiated services)

- IP ヘッダの TOS フィールドの再定義 → DS フィールド
- クラス分類による CoS (QoS) フォワーディング



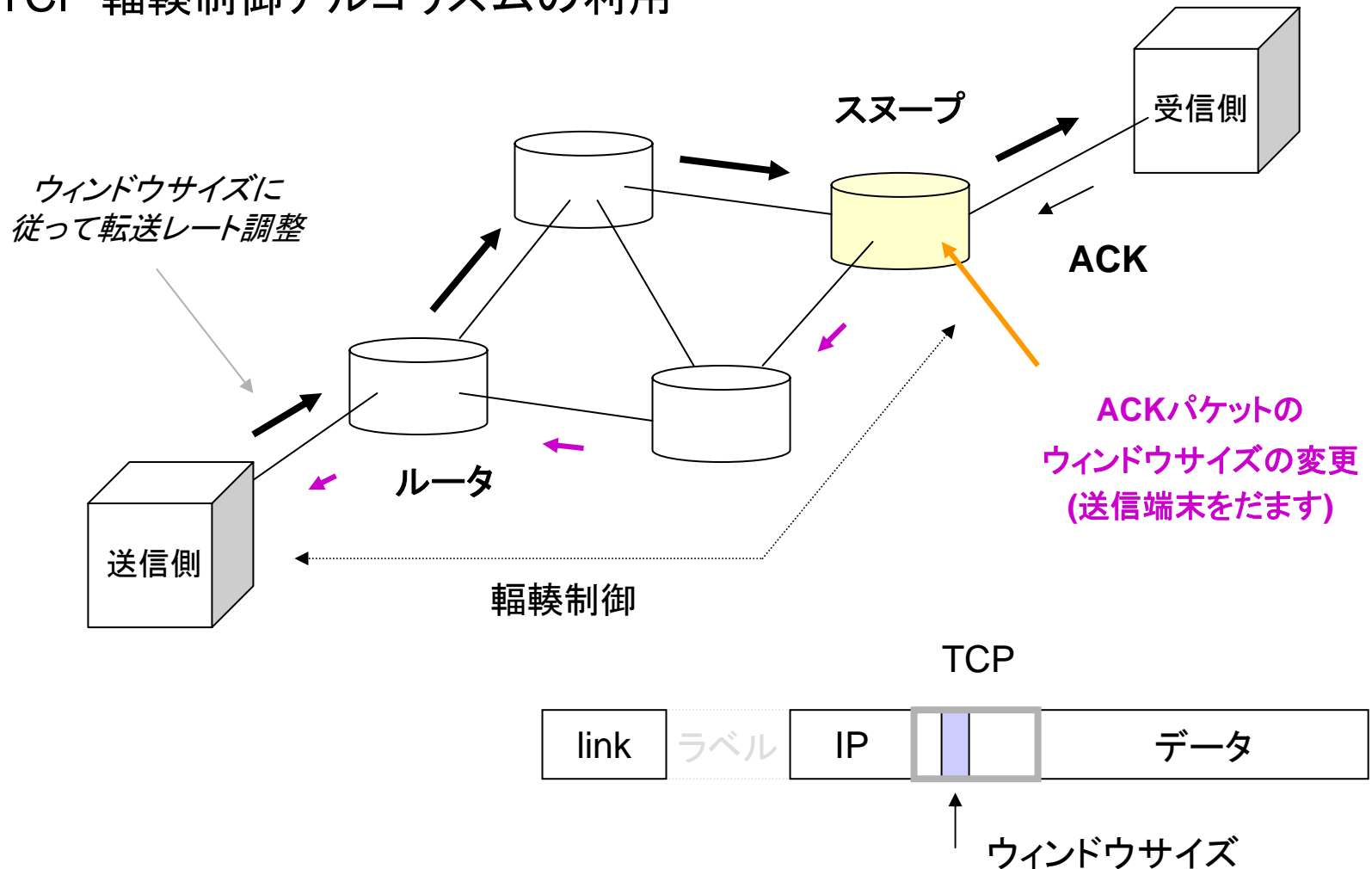
RSVP (intserv)

- ルータ間のメッセージ交換による帯域確保
- スケーラビリティ (大規模化) に問題



TCP スヌーピング

- ルータによる TCP ヘッダのスヌーピング (L4-Switch)
- TCP 輻輳制御アルゴリズムの利用

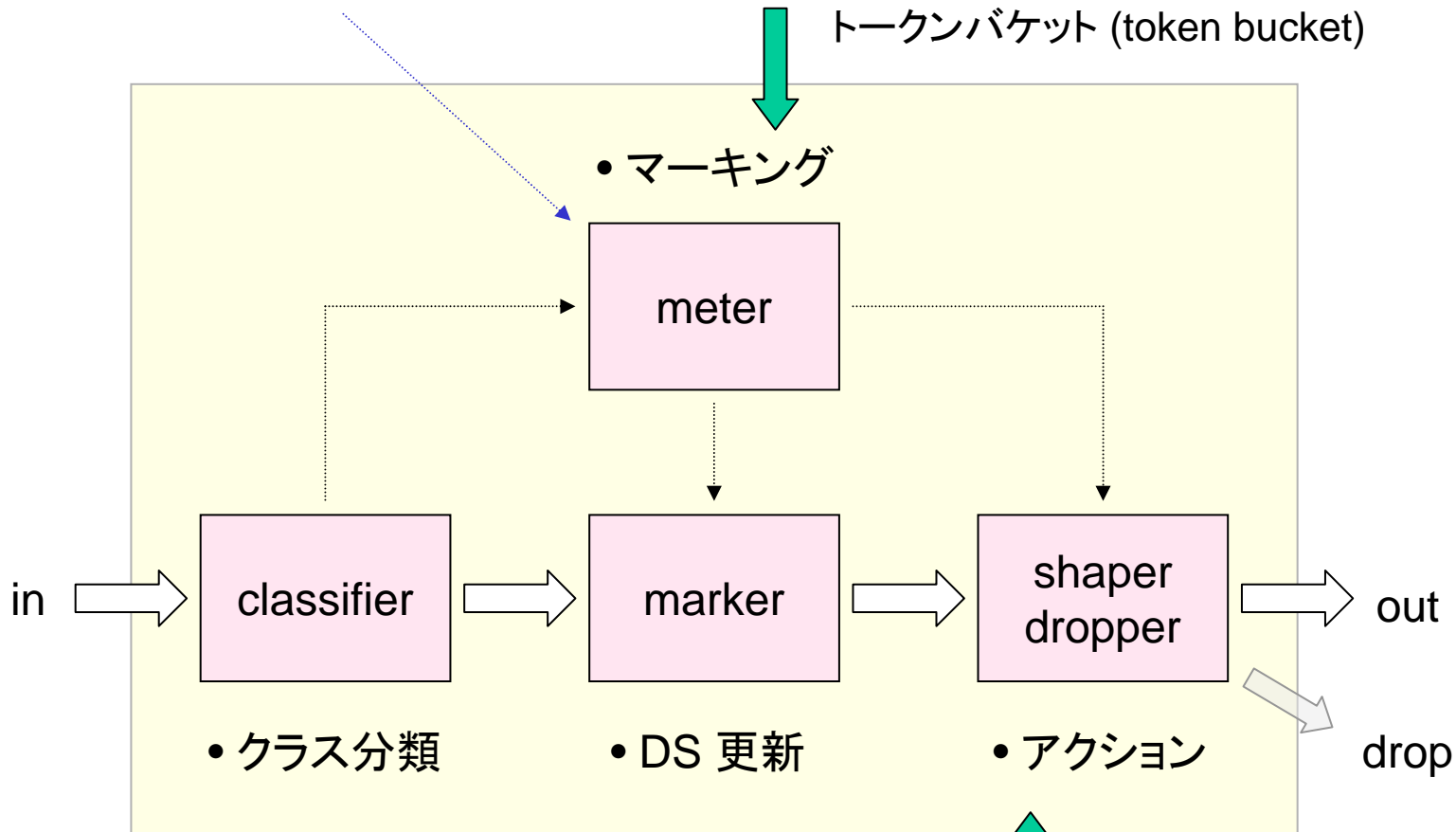


フローの差別化 (1)

SLA 設定 (帯域ブローカ)

TCM (Three Color Marker)

トークンバケット (token bucket)

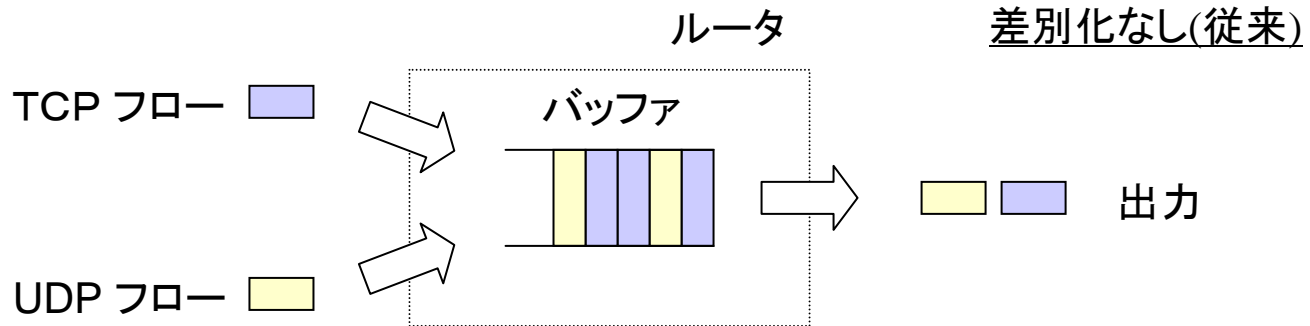


Diffservルータの構成例

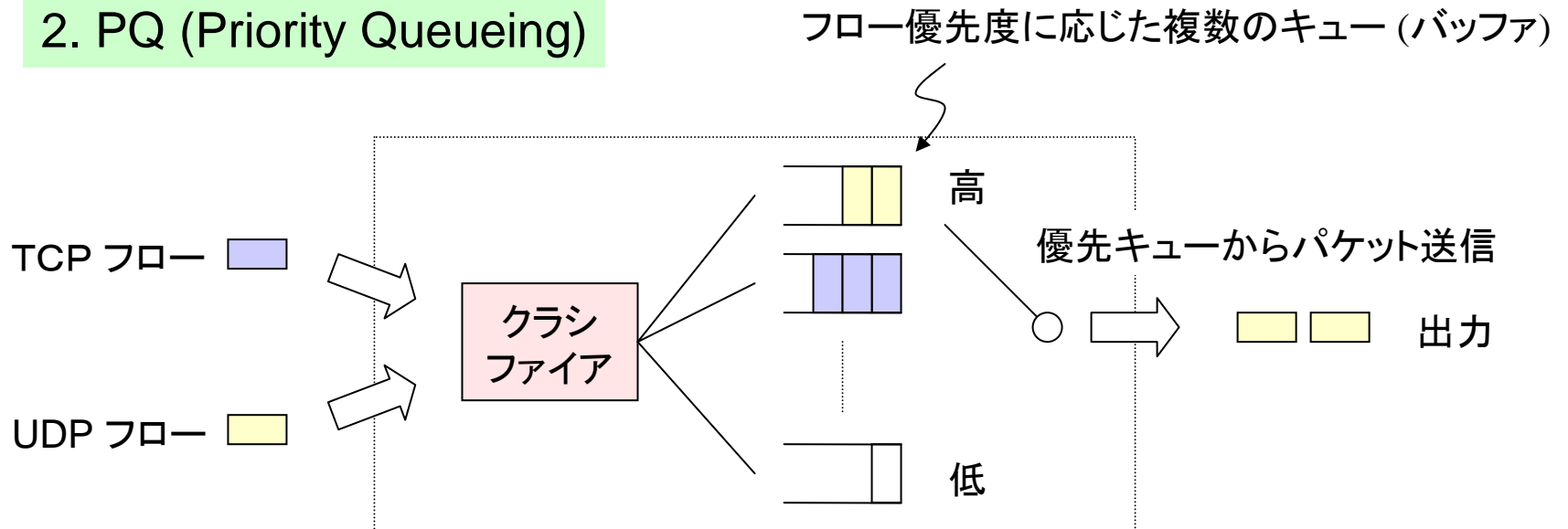
各種の制御アルゴリズム
PQ, WRR, WFQ, CFQ, ...

フローの差別化 (2)

1. FIFO (First-In First-Out)

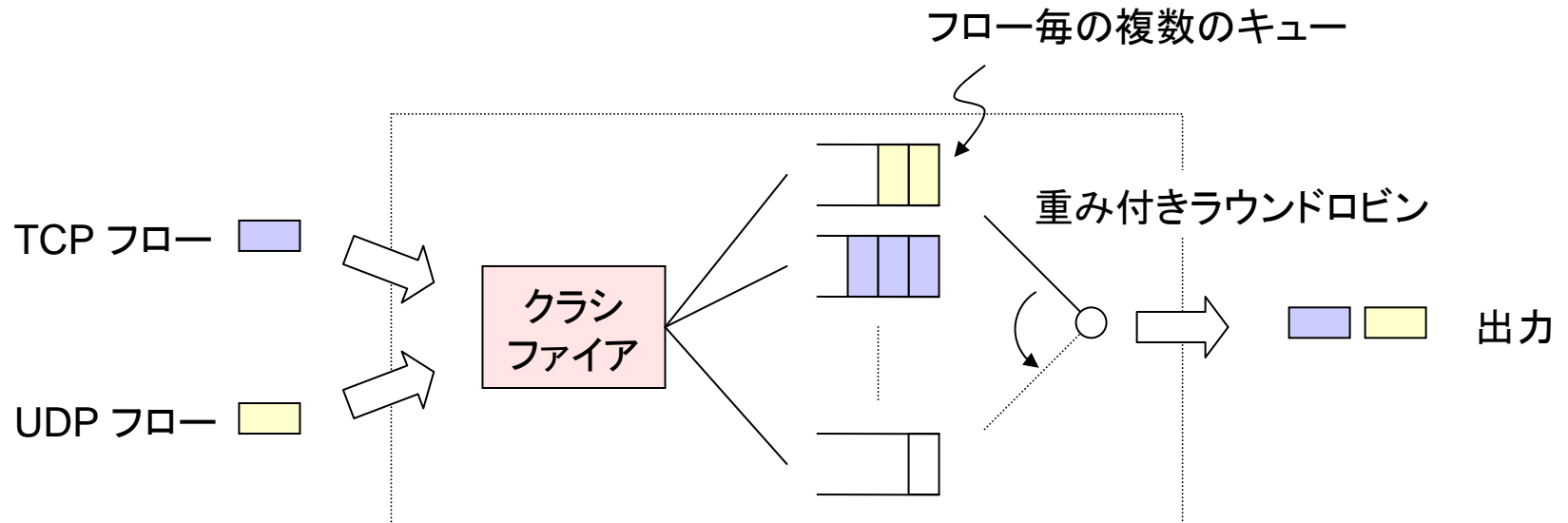


2. PQ (Priority Queueing)



フローの差別化 (3)

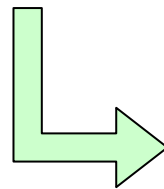
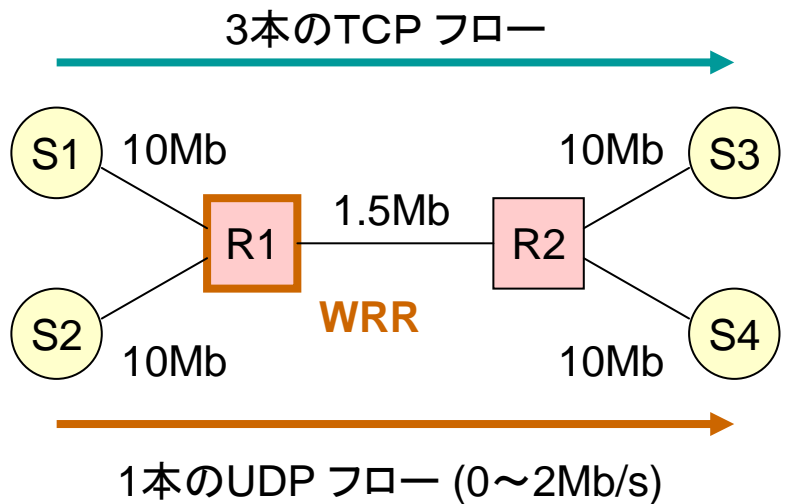
3. WRR (Weighted Round Robin)



- PQ: 優先キューが空になるまで非優先パケットは送出されない (欠点)
- WRR: 既定の個数のパケットを送出すると非優先パケットを送出する (改善)

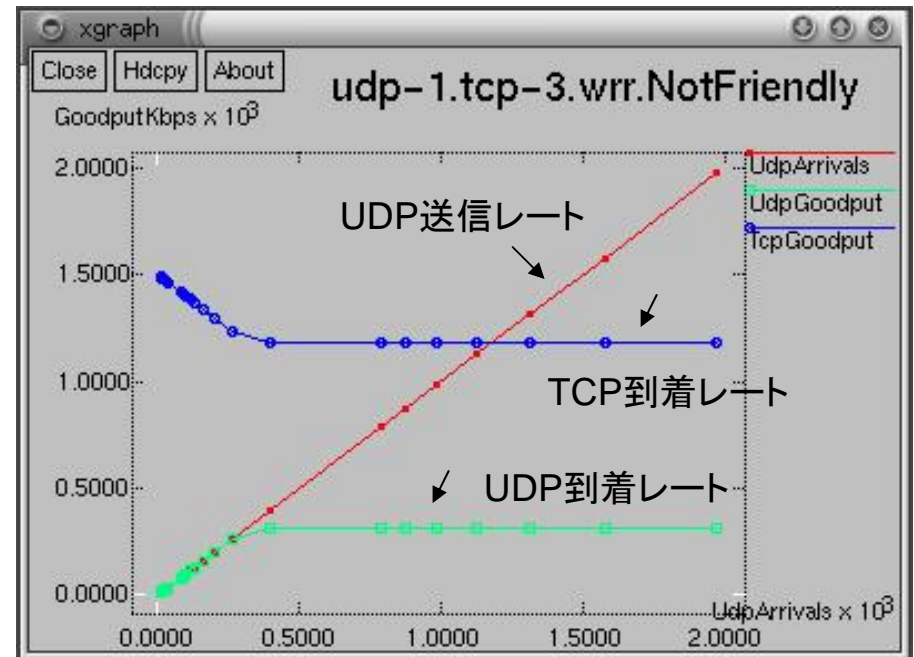
フローの差別化 (4)

• WRR の効果



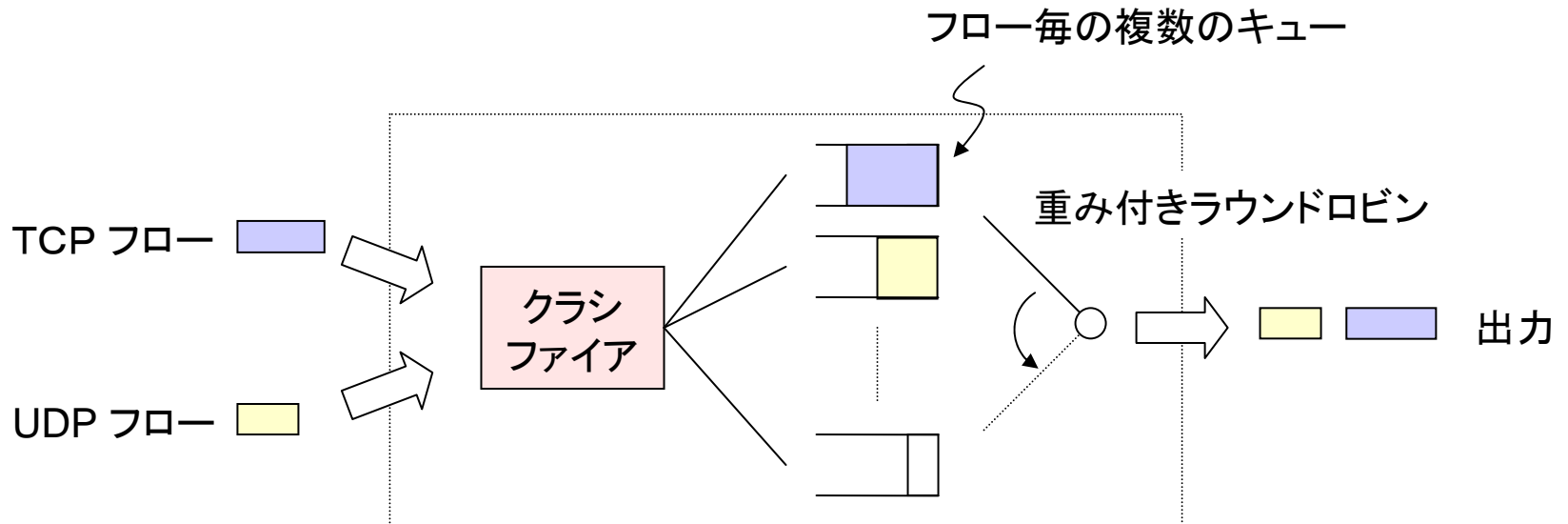
ネットワークシミュレータ

3本のTCPフローと1本のUDPフローがほぼ均等に帯域をシェアしている。クラス許容量を超えたUDPは廃棄。



フローの差別化 (5)

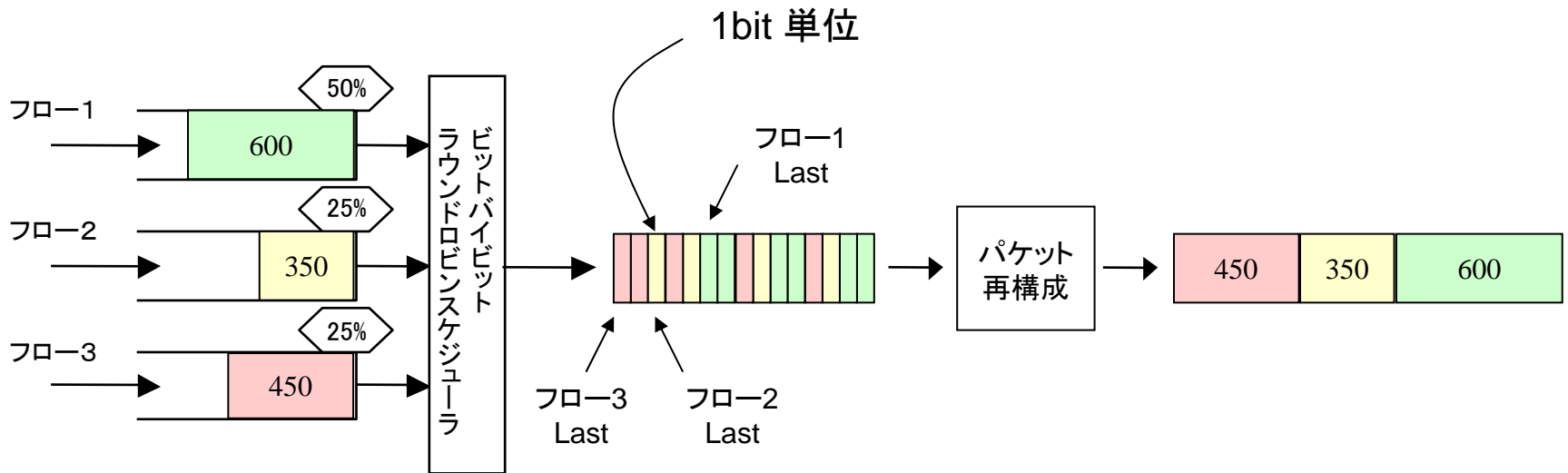
4. DRR (Deficit Round Robin)



- WRR: 可変長パケットが来ると、長いパケットが優先 (欠点)
- DRR: パケットの「個数」ではなく、「バイト数」で重み付け (改善)

フローの差別化 (6)

5. WFQ (Weighted Fair Queuing)



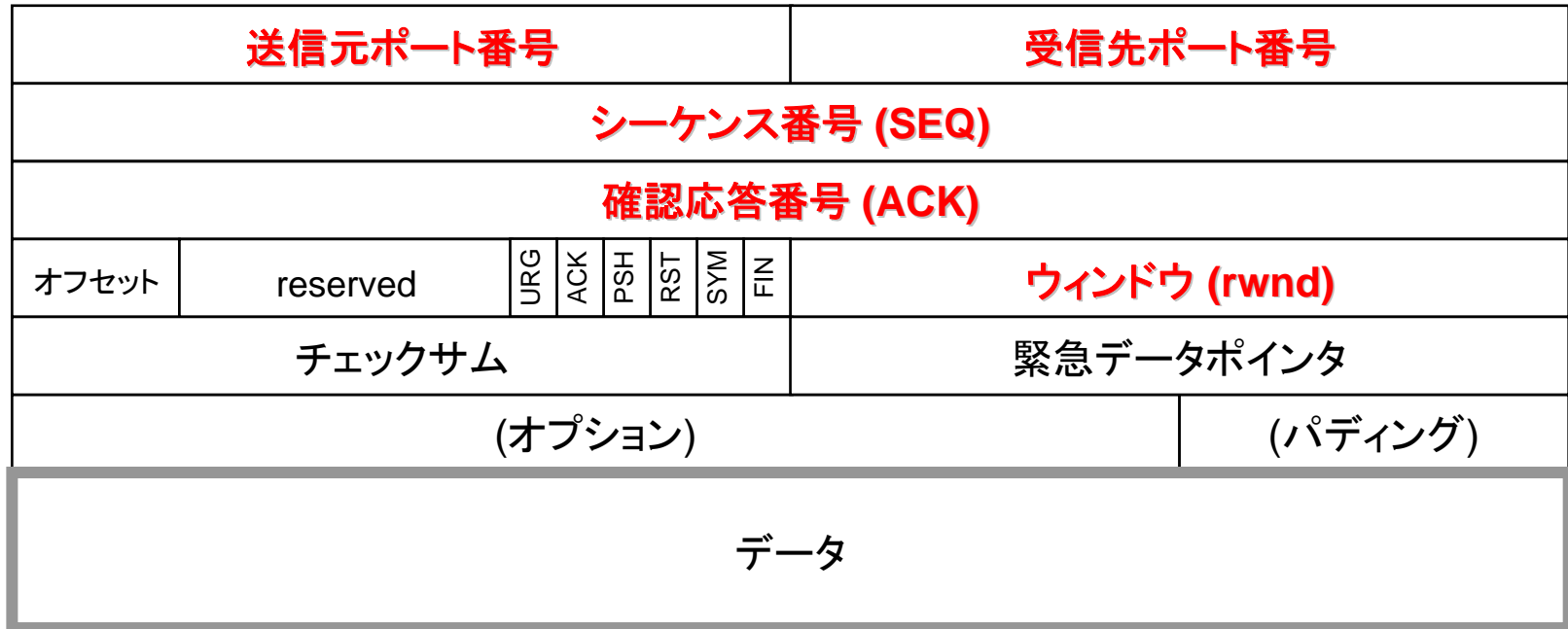
- WRR/DRR: 「個数/帯域幅」は weighted fair だが、「到着時間」は not fair
- WFQ: 「帯域幅」だけではなく、「到着時間」も weighted fair

TCP

Transport Control Protocol

TCP ヘッダ

4 byte



ポート番号: アプリケーションの識別

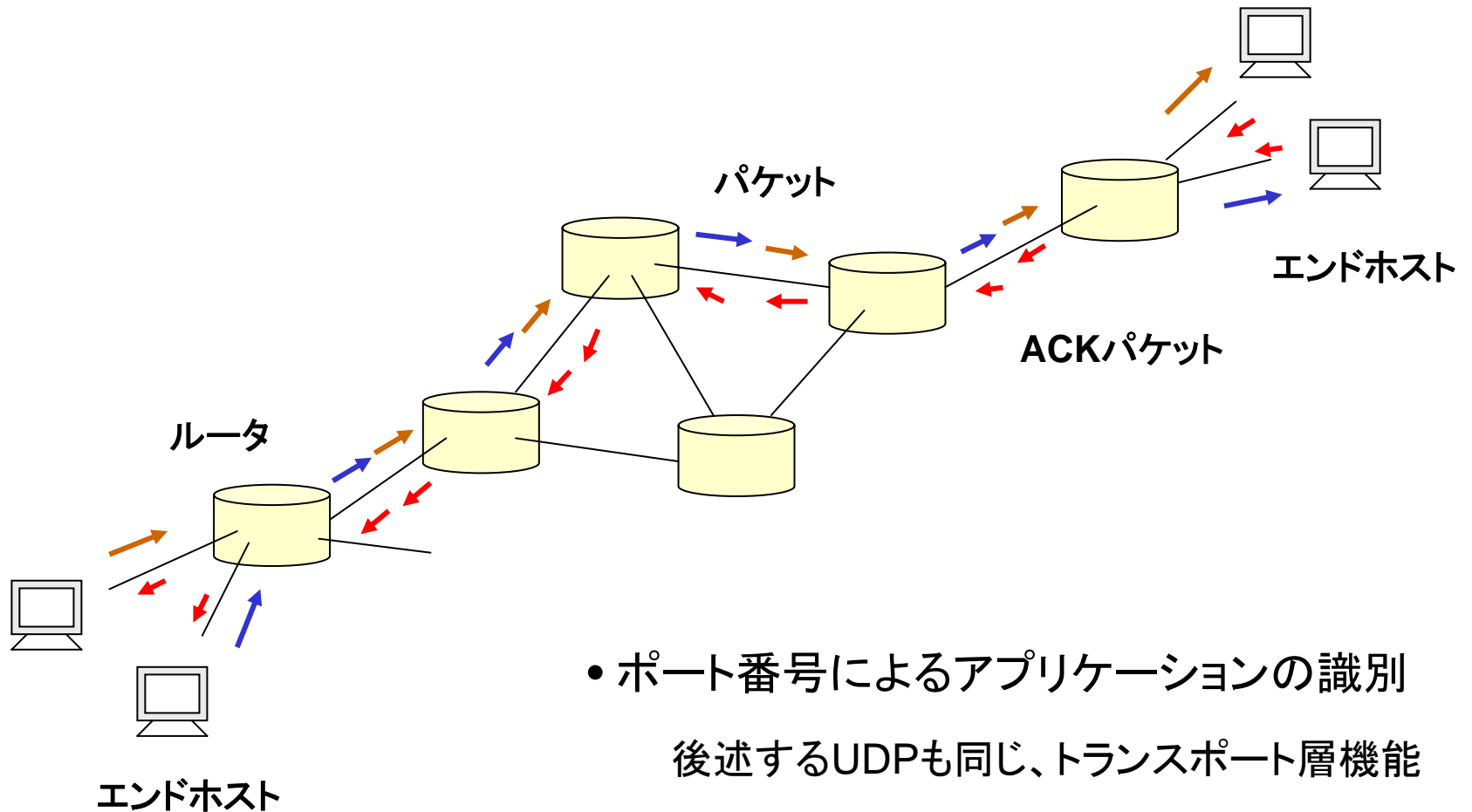
シーケンス番号: パケット廃棄、順序逆転を検出 (バイト単位でカウント)

確認応答番号: 次パケットで受信予定のシーケンス番号、あるいは重複 ACK の通知

ウィンドウ: 受信者が求める最大セグメントサイズ

TCP の機能

- End-to-End の確認応答による誤り制御とフロー制御

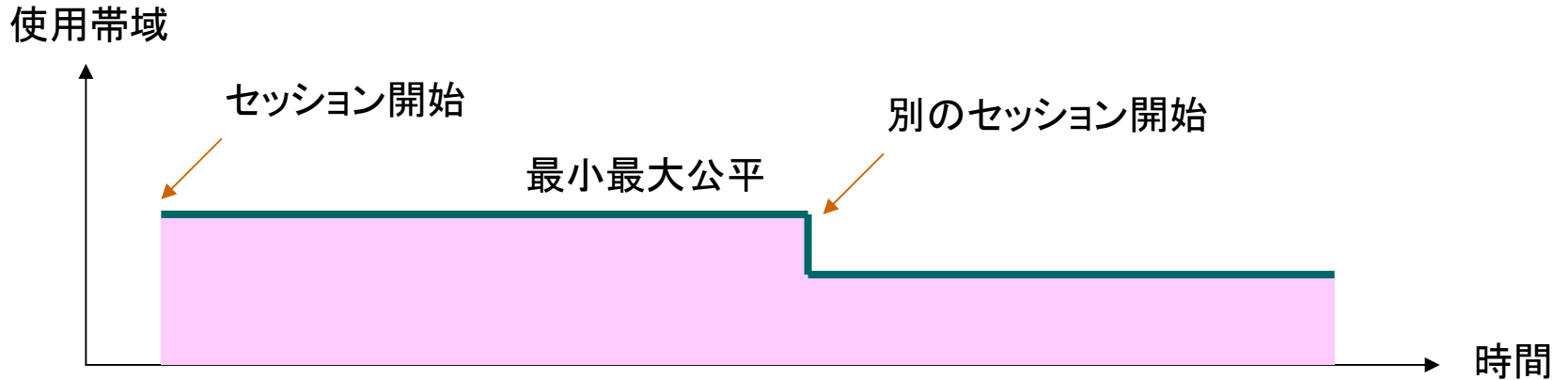


- ポート番号によるアプリケーションの識別
後述するUDPも同じ、トランスポート層機能
いわゆる well-known port

TCPにおけるフロー制御

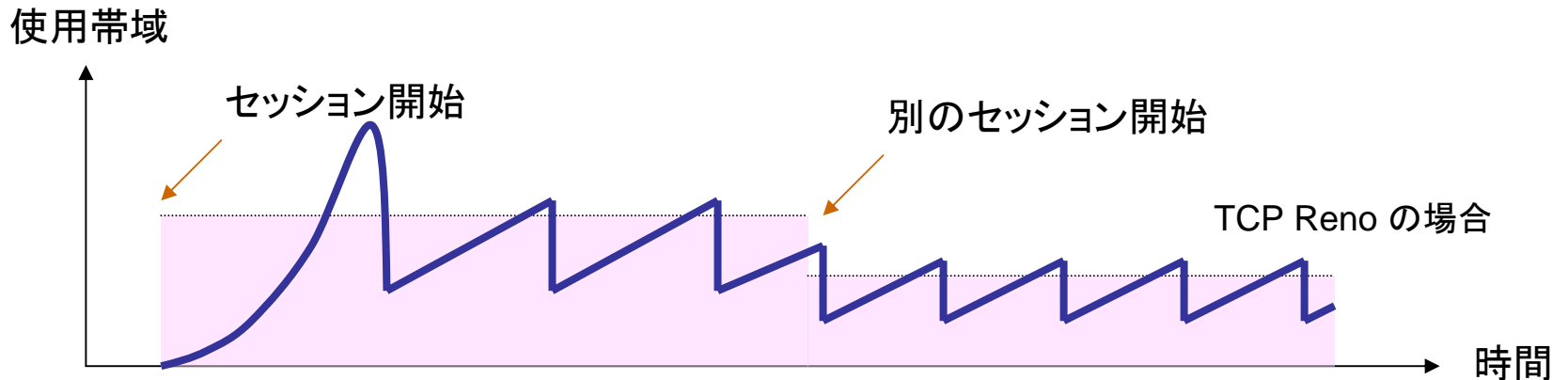
理想:

* 集中型の帯域管理装置 (電話に近い)



TCP: スロースタート + ふくそう回避

* 端末毎の分散制御



いろいろなTCP

	要点
TCP Tahoe	スロースタート + ふくそう回避 + 高速再送
TCP Reno	Tahoe + 高速回復
TCP Vegas	RTT (round trip delay) ベースのふくそう制御
TCP SACK	Reno + 選択的再送 (selective repeat)

- スロースタート: slow start
- ふくそう回避: congestion avoidance
- 高速再送: fast retransmission
- 高速回復: fast recovery

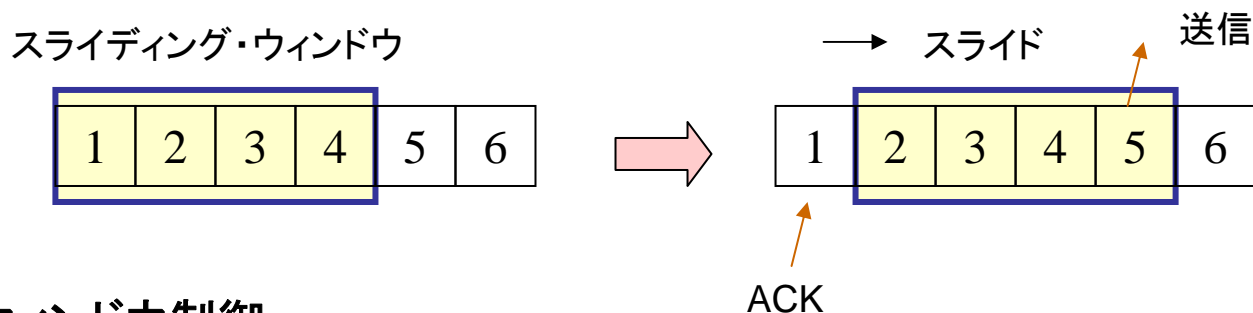
* 広く用いられているのは TCP Reno 系 (SACKを含む)

古典的なTCP

- **Go-Back-N ARQ (スライディング・ウィンドウ):**

送信者は ACK を待たずに N 個の packets を送信する

受信者が ACK を返すとウィンドウがスライドして次 packets が送出される
しばしば n 個の packets 毎に1つの ACK を返す (累積応答)



- **ウィンドウ制御:**

rwnd: 広告ウィンドウ (advertisement window)

受信者が要求するセグメント (packet) サイズ、あるいは受信可能なセグメントサイズを通知し、スライディングウィンドウ (送信 packet 数) を制御

欠点: ボトルネックリンクに非常に弱い

TCP Tahoe (1)

- 送信側パラメータを三つ追加:

 cwnd: ふくそうウィンドウ (congestion window: 初期値1)

 ssthresh: スロースタートとふくそう回避のモード選択閾値 (初期値大)

 tcprecvthresh: 高速再送を行う重複ACK数 (通常は3)

- スロースタート (指数増加: スループット探索モード):

 if (cwnd < ssthresh)

 --- ACK 毎にパケットを2個送出 ---

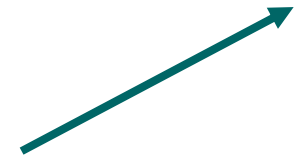
 cwnd += 1;

- ふくそう回避 (加法増加: スループット安定モード):

 else if (cwnd >= ssthresh)

 --- ACK 毎にパケットを1個送出、cwnd 個送出後1個追加 ---

 cwnd += 1/cwnd;



TCP Tahoe (2)

- 二通りのパケット廃棄の検出:

- (1) 重複 ACK の受信 (TCP ヘッダの ACK ナンバが更新されない場合)
- (2) タイムアウト (ACK が返って来ない場合)

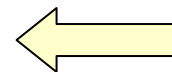
- 高速再送 (軽いふくそう):

ACK が返って来るということは深刻なふくそうではない (仮定)

```
if ( 重複 ACK 数 == tcprecvthresh )
```

```
    --- パケットを再送 ---
```

```
    ssthresh = cwnd/2; cwnd = 1;
```



スロースタートから再開
(ssthresh > cwnd)

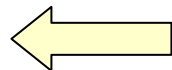
- タイムアウト値の更新 (重いふくそう):

タイムアウトが起こるということは深刻なふくそう (仮定)

```
if ( タイムアウト )
```

```
    --- パケットを再送 ---
```

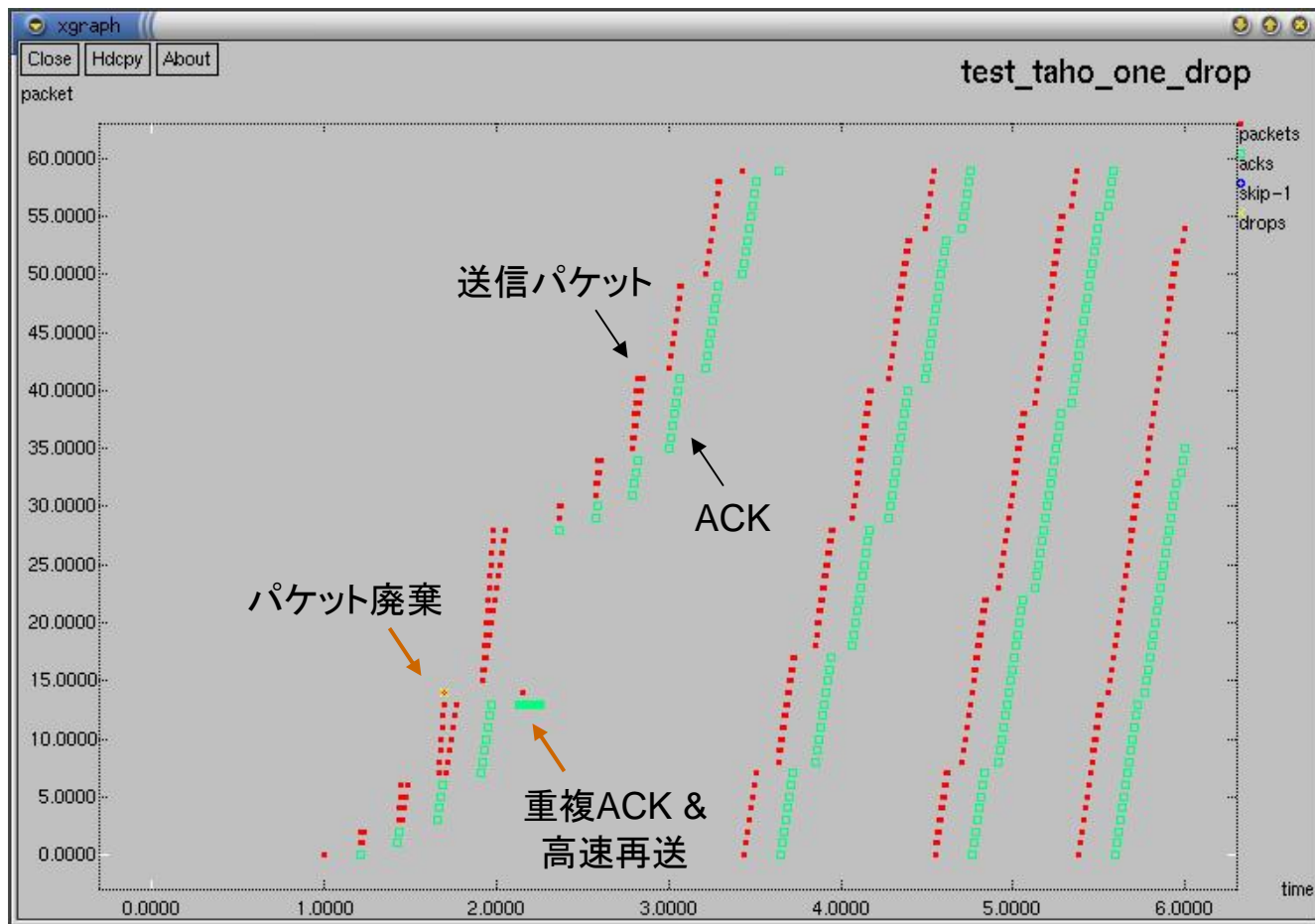
```
    timeout *= 2;
```



指数的バックオフ

TCP Tahoe (3)

パケット数



→

スロースタート(1)

スロースタート(2)

ふくそう回避

NS (Network Simulator) によるシミュレーション例

TCP Reno (1)

- Tahoe の問題点:

高速再送後、スロースタートに戻る必要は無い

パケット廃棄前の cwnd の値は安全 (仮定: 現在の cwnd の半分)

- 高速回復:

```
if ( 重複 ACK 数 == tcprecvthresh )
```

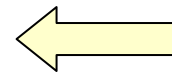
```
--- パケットを再送 (高速再送) ---
```

```
ssthresh = cwnd/2;
```

```
cwnd = cwnd/2 + tcprecvthresh;
```

↑
安全な値

↑
重複 ACK 分 (ACK が正しく返っている)



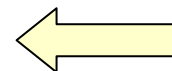
ふくそう回避モードから再開
(ssthresh < cwnd)

```
if ( 重複 ACK 数 > cwnd/2 )
```

```
--- 重複 ACK 毎に新しいパケットを一つ送信 ---
```

```
if ( 再送パケットの確認応答 )
```

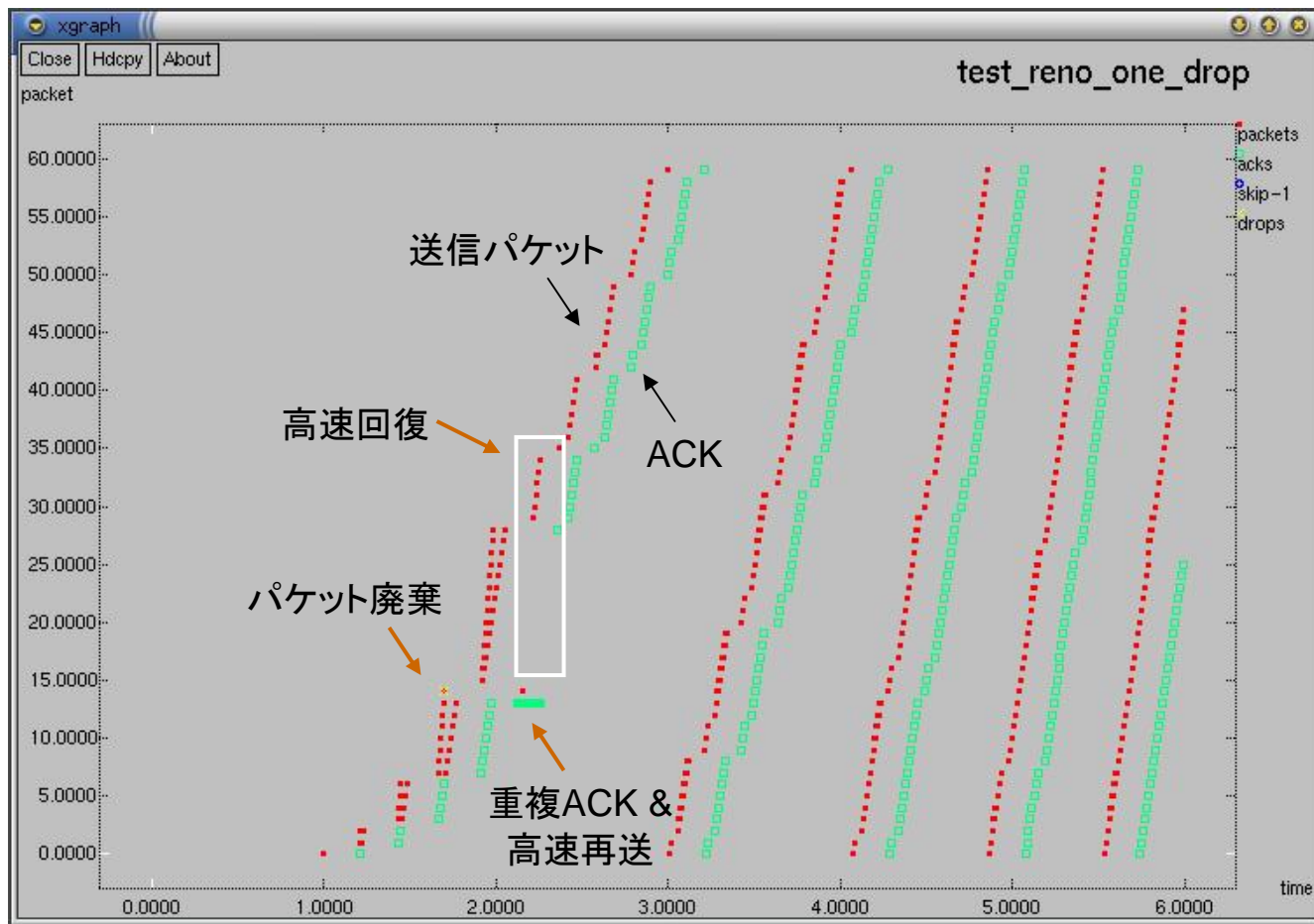
```
cwnd = ssthresh;
```



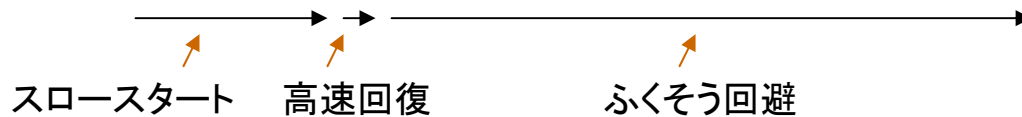
通常のふくそう回避へ

TCP Reno (2)

パケット数



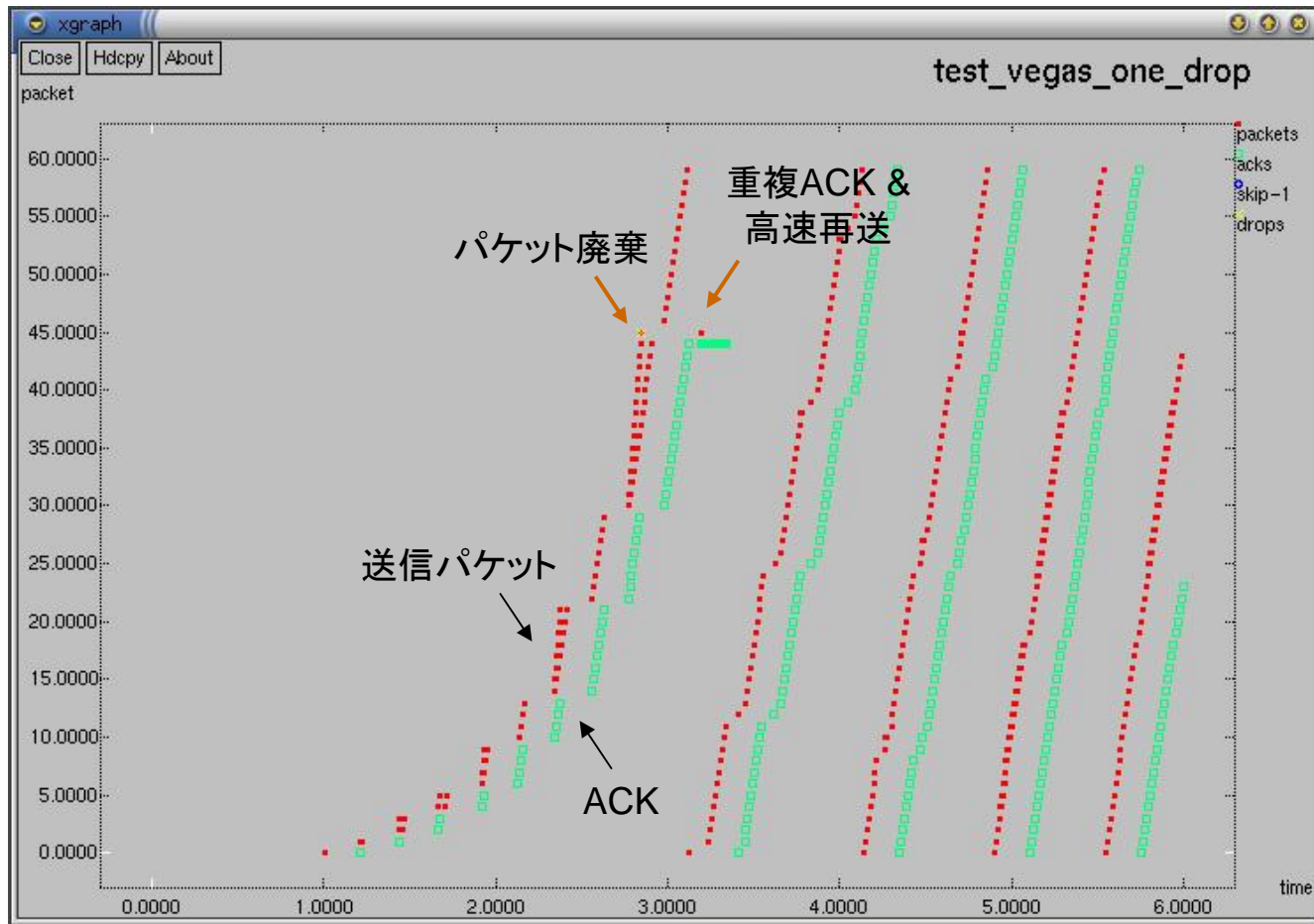
時間



NS (Network Simulator) によるシミュレーション例

TCP Vegas (2)

パケット数



時間

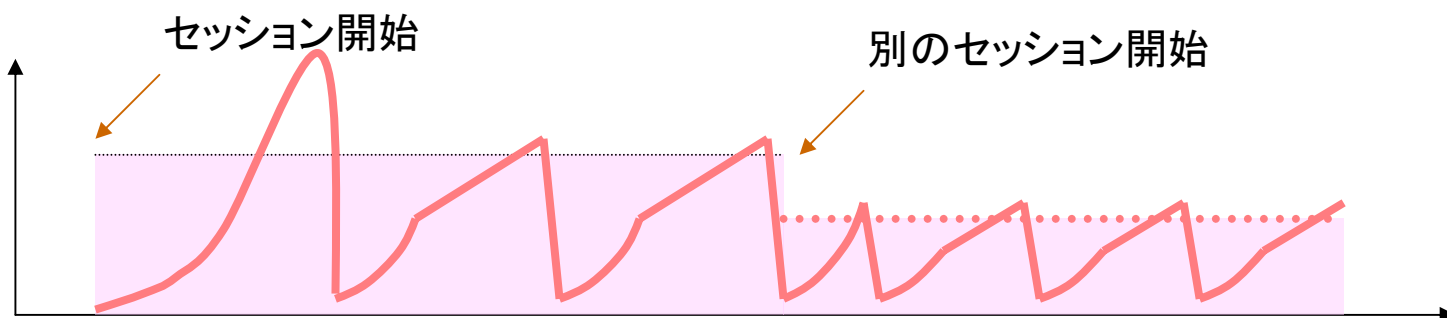
→
スロースタート

→
ふくそう回避

NS (Network Simulator) によるシミュレーション例

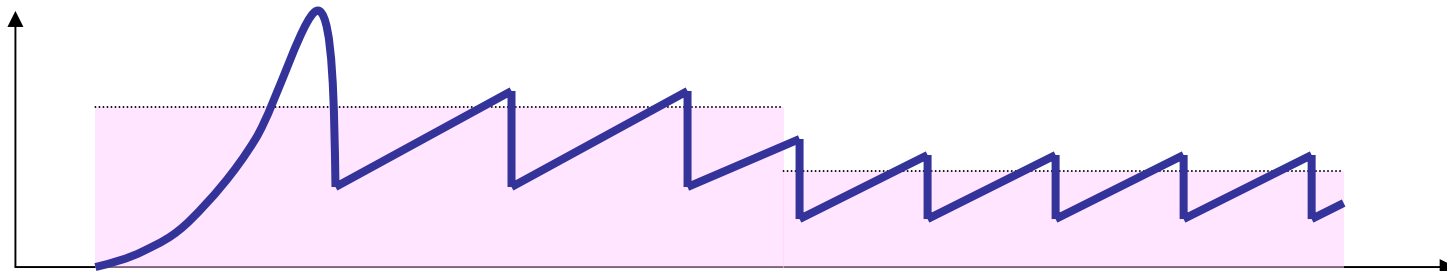
直感的な比較

Tahoe



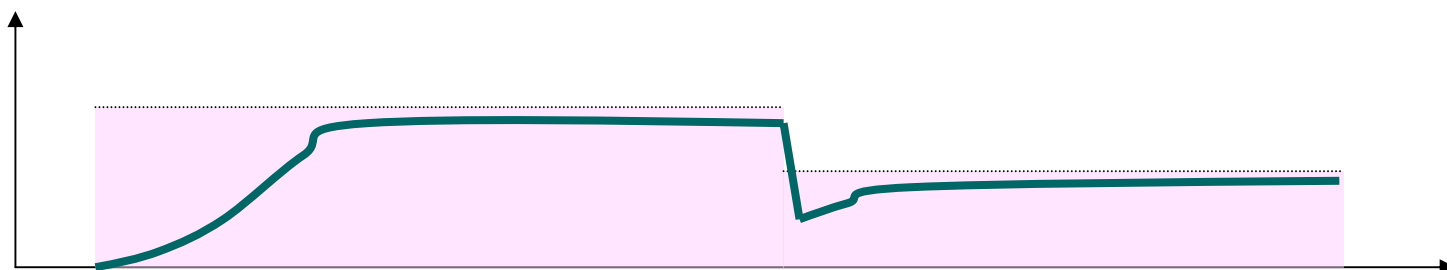
廃棄が起きたら ssthresh を半分にして slow start から再開

Reno



廃棄が起きても重複 ACK を受け取ったら congestion avoidance から再開

Vegas

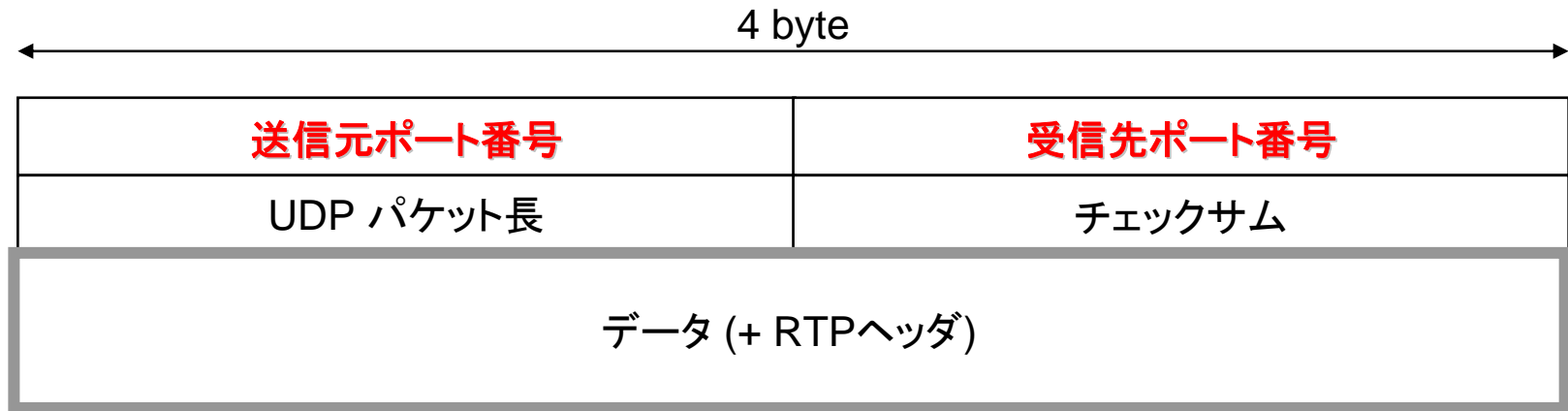


RTT をモニタしながら廃棄が起きないように cwnd を制御

UDP

User Datagram Protocol

UDP ヘッダ



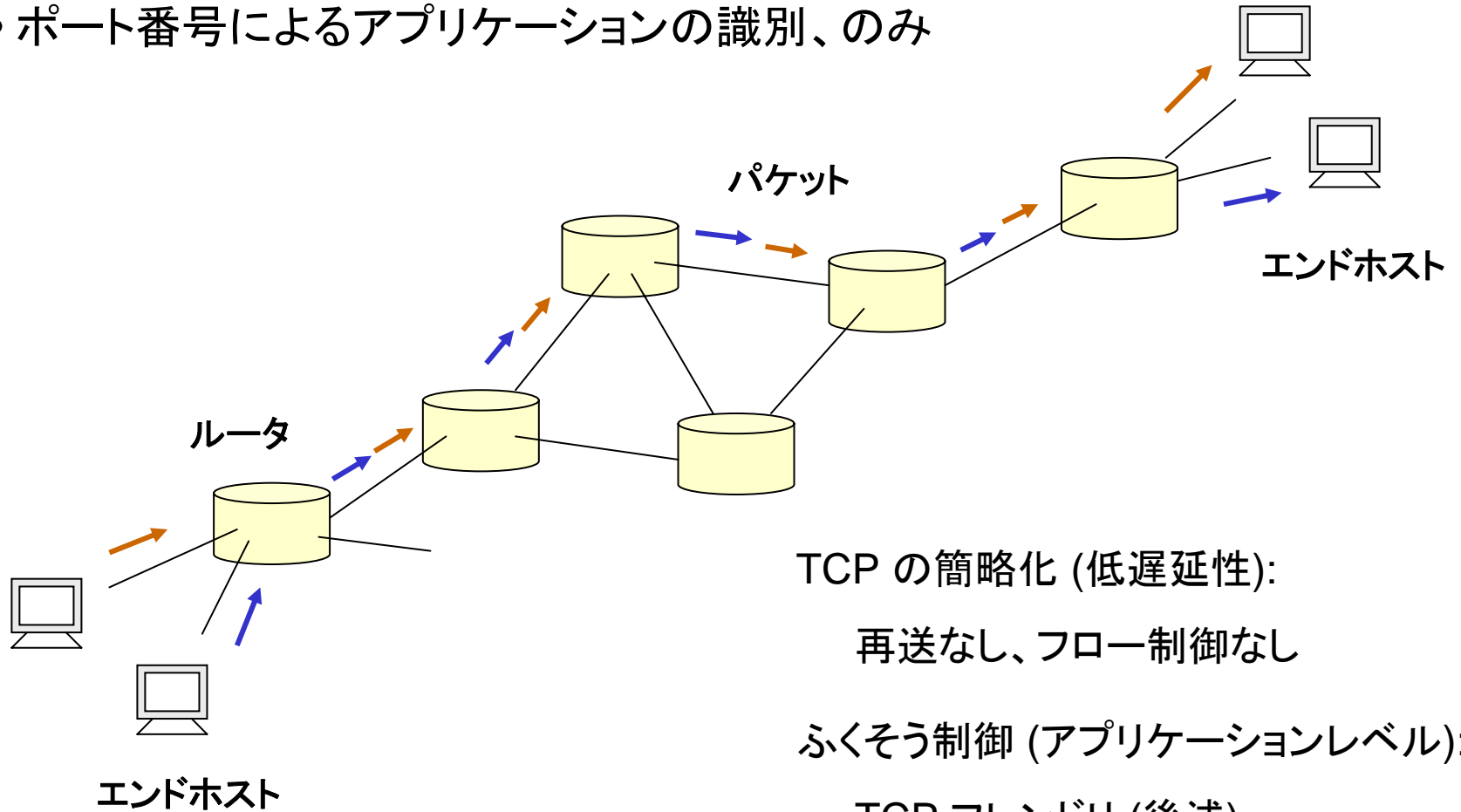
ポート番号: アプリケーションの識別

パケットの紛失、重複、順序逆転などについてまったく関知しない

→ アプリケーションで対処

UDP の機能

- ポート番号によるアプリケーションの識別、のみ



TCP の簡略化 (低遅延性):

再送なし、フロー制御なし

ふくそう制御 (アプリケーションレベル):

TCP フレンドリ (後述)

低遅延 (UDP) ↔ 信頼性 (TCP)

UDPのまとめ

- 再送を行わない信頼性のないデータ転送:

転送遅延は抑えられる。

→ 遅延に敏感な IP 電話にとっては大きな利点。ACK 爆発が発生しないため、マルチキャストにも適している。

- アプリケーションレベルの誤り制御とふくそう制御 (アダプテーション):

パケット廃棄やネットワークの輻輳に対して UDP は何も行わないため、アプリケーションレベルで対処する必要がある。

→ 再同期 (パケット廃棄対策)、TCP フレンドリ (輻輳制御)、信頼性マルチキャスト (NACK あるいは FEC)、等

TCP と UDP: まとめ

インターネット電話	TCP	UDP
メディア情報	△	◎
制御情報	◎	△

インターネット放送	TCP	UDP
オンデマンド放送	○	○
ライブ放送	×	◎
マルチキャスト	×	◎ (クラスD)
制御情報	◎	○ (カルーセル)

Etherereal / Wireshark

パケットキャプチャツール

SIPの実例

Packet List:

No.	Time	Source	Destination	Protocol	Info
18	20.685553	133.9.250.234	133.9.250.229	SIP/SDP	Request: INVITE sip:katto@133.9.250.229, with session des
19	20.692912	133.9.250.229	133.9.250.234	SIP	Status: 100 Trying
20	20.793854	133.9.250.229	133.9.250.234	SIP	Status: 180 Ringing
23	26.929883	133.9.250.229	133.9.250.234	SIP/SDP	Status: 200 OK, with session description
24	27.431929	133.9.250.229	133.9.250.234	SIP/SDP	Status: 200 OK, with session description
25	27.503385	133.9.250.234	133.9.250.229	SIP	Request: ACK sip:katto@133.9.250.229:5060
26	27.503822	133.9.250.234	133.9.250.229	SIP	Request: ACK sip:katto@133.9.250.229:5060
1476	41.906306	133.9.250.234	133.9.250.229	SIP	Request: BYE sip:katto@133.9.250.229:5060
1478	41.919481	133.9.250.229	133.9.250.234	SIP	Status: 100 Trying
1480	42.407312	133.9.250.234	133.9.250.229	SIP	Request: BYE sip:katto@133.9.250.229:5060
1484	42.759780	133.9.250.229	133.9.250.234	SIP	Status: 200 OK
1485	42.760981	133.9.250.229	133.9.250.234	SIP	Status: 200 OK
1486	42.768199	133.9.250.234	133.9.250.229	SIP	Request: ACK sip:katto@133.9.250.229:5060

Packet Details (Packet 23):

- Session Initiation Protocol
Status line: SIP/2.0 200 OK
- Message Header
 - To: sip:katto@133.9.250.229;tag=954937868
 - From: sip:katto@133.9.250.234;tag=1171186158
 - CSeq: 1 INVITE
 - Call-ID: -1136463554146893190@133.9.250.234
 - Via: SIP/2.0/UDP 133.9.250.234:5060;branch=8509FAEA13C4000000EE25AC7265-3*0
 - Content-Type: application/sdp
 - Contact: sip:katto@133.9.250.229:5060
 - Content-Length: 123
- Session Description Protocol
 - Session Description Protocol Version (v): 0
 - Owner/Creator, Session Id (o): - 1022834342125 1022834348304 IN IP4 133.9.250.229
 - Session Name (s): -
 - Connection Information (c): IN IP4 133.9.250.229
 - Time Description, active time (t): 0 0
 - Media Description, name and address (m): audio 5006 RTP/AVP 3 0 8

Packet Bytes:

0000	00 02 b3 90 19 b8 00 d0 b7 44 94 44 08 00 45 00D.D..E.
0010	01 e7 9a 76 00 00 80 11 9e ac 85 09 fa e5 85 09	...V....
0020	fa ea 0b 23 13 c4 01 d3 28 b8 53 49 50 2f 32 2e	...#.... (.SIP/2.

Filter: ip.addr == 133.9.250.229 && udp.port == 5060

RTSPの実例

The screenshot shows a Wireshark capture of network traffic. The main pane displays a list of packets, with packet 3062 selected. The packet list pane shows the following details:

No.	Time	Source	Destination	Protocol	Info
3057	257.612308	133.9.250.234	210.150.14.100	RTSP	OPTIONS rtsp://ondemand.stream.co.jp:554 RTSP/1.0
3058	257.618303	210.150.14.100	133.9.250.234	TCP	554 > 3891 [ACK] Seq=2290611431 Ack=4193171852 win=30
3059	257.621482	210.150.14.100	133.9.250.234	RTSP	RTSP/1.0 200 OK
3060	257.658587	133.9.250.234	210.150.14.100	RTSP	DESCRIBE rtsp://ondemand.stream.co.jp:554/yomiuri/res
3061	257.677409	210.150.14.100	133.9.250.234	TCP	554 > 3891 [ACK] Seq=2290611728 Ack=4193172237 win=32
3062	257.779980	210.150.14.100	133.9.250.234	RTSP/SDP	RTSP/1.0 200 OK, with session description
3063	257.842409	133.9.250.234	210.150.14.100	RTSP	SETUP rtsp://ondemand.stream.co.jp:554/yomiuri/result
3064	257.850715	210.150.14.100	133.9.250.234	RTSP	RTSP/1.0 200 OK
3065	258.027032	133.9.250.234	210.150.14.100	TCP	3891 > 554 [ACK] Seq=4193172654 Ack=2290613182 win=63
3066	258.039776	133.9.250.234	210.150.14.100	RTSP	SET_PARAMETER rtsp://ondemand.stream.co.jp:554/yomiur
3067	258.046455	210.150.14.100	133.9.250.234	RTSP	RTSP/1.0 200 OK
3068	258.046647	133.9.250.234	210.150.14.100	RTSP	PLAY rtsp://ondemand.stream.co.jp:554/yomiuri/result/
3069	258.053513	210.150.14.100	133.9.250.234	RTSP	RTSP/1.0 200 OK
3070	258.053692	133.9.250.234	210.150.14.100	RTSP	SET_PARAMETER * RTSP/1.0
3078	258.060590	210.150.14.100	133.9.250.234	RTSP	RTSP/1.0 451 Parameter Not Understood
3082	258.277311	133.9.250.234	210.150.14.100	TCP	3891 > 554 [ACK] Seq=4193173009 Ack=2290613542 win=63

Below the packet list, the details pane for frame 3062 is expanded, showing the following structure:

- Frame 3062 (1267 on wire, 1267 captured)
- Ethernet II
- Internet Protocol, Src Addr: 210.150.14.100 (210.150.14.100), Dst Addr: 133.9.250.234 (133.9.250.234)
- Transmission Control Protocol, Src Port: 554 (554), Dst Port: 3891 (3891), Seq: 2290611728, Ack: 4193172237
- Real Time Streaming Protocol
 - RTSP/1.0 200 OK\r\n
 - CSeq: 2\r\n
 - Date: Mon, 17 Jun 2002 11:01:13 GMT\r\n
 - Set-Cookie: cbid=rfjjhheimjikhldmeiooopltmrjrkltufkcgkielnjcfclplpnpooprtfrqntmuffhjchpi;path=/;expires=Thu, 31-Dec-21
 - vsrc: http://ondemand.stream.co.jp:80/viewsourc/template.html?nuyhtg8slz26t1k5bfyaysz6lg270hh4ppbgenrDr eh30157cacmmf
 - X-TSPort: 7802\r\n
 - Last-Modified: Thu, 06 Jun 2002 09:11:46 GMT\r\n
 - Content-base: rtsp://ondemand.stream.co.jp:554/yomiuri/result/project01/kikaku/digitalw.smi\r\n
 - ETag: 2028192288-1\r\n
 - Session: 2028192288-1\r\n
 - Content-type: application/sdp\r\n

At the bottom, the packet bytes pane shows hex and ASCII data for packets 0300, 0310, and 0320. The filter bar at the bottom is set to: `ip.addr == 210.150.14.100 && ip.proto == 0x06` with buttons for `Reset`, `Apply`, and `Session Description Protocol (sdp)`.