# 画像情報特論 (2)

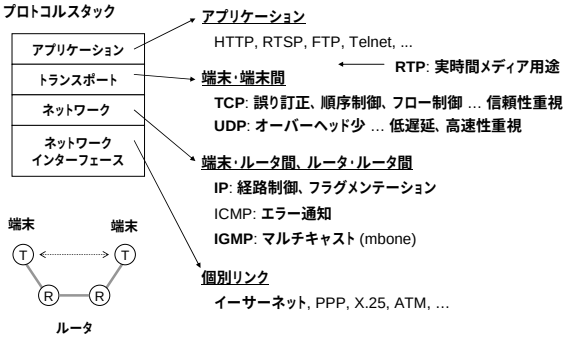
- TCP/IP
  - **インターネットプロトコル** (IP)
  - Network-Assisted QoS
  - TCP (Transport Control Protocol)
  - UDP (User Datagram Protocol)

**情報ネットワーク専攻　甲藤二郎**
E-Mail: katto@waseda.jp

# IP

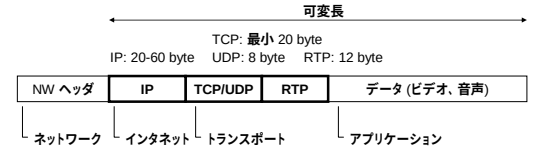
(Internet Protocol)

## プロトコルスタック

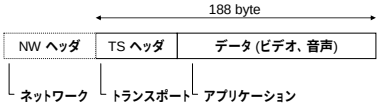



## IP データグラム

**IP データグラム**

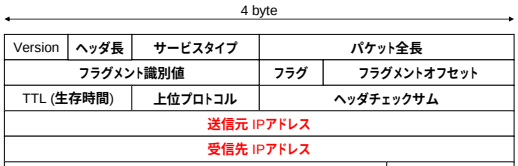



*cf.* MPEG-2 **トランスポートストリーム** (ITU-T H.222)




## IPv4 ヘッダ

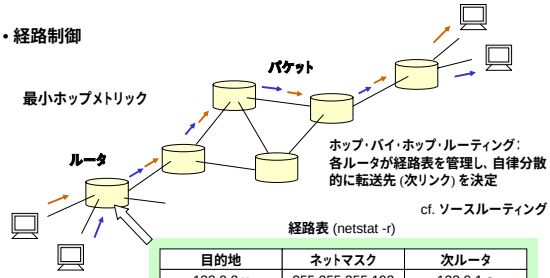
**IPv4ヘッダ**




**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

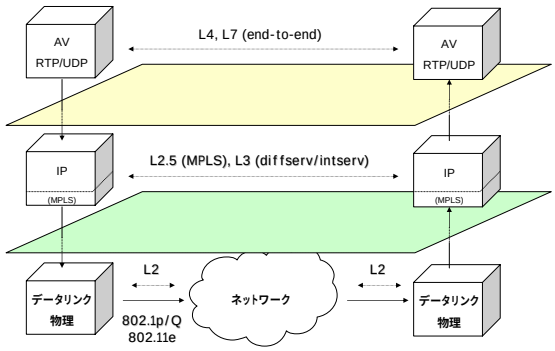
- 蓄積交換 (store and forward) 故に、パケット転送時間の増大 (delay)、転送時間の揺らぎ (jitter)、パケット廃棄の発生 (packet loss) 等の問題は避けられない。
- パケットの到着順序が逆転することもある (順序制御)。

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

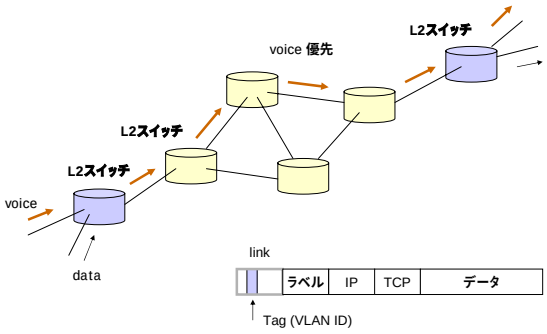
MPLS、Diffserv、トラヒックシェイピング、RSVP

## インターネットQoS



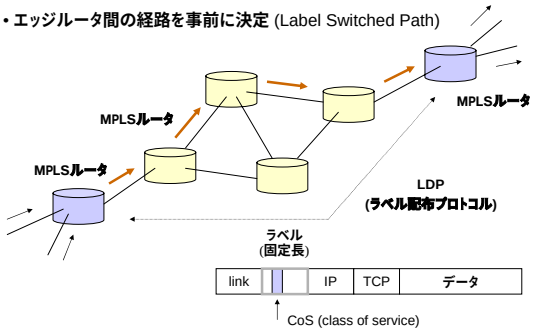

## IEEE 802.1p/Q

- Tag を用いたリンク毎の優先制御




## MPLS (ラベル・スイッチング)

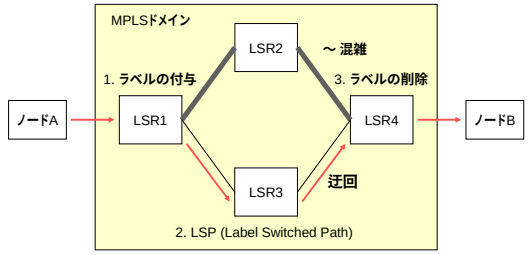
- 固定長ラベルによるハードウェアスイッチング
- エッジルータ間の経路を事前に決定 (Label Switched Path)




Multi-Protocol Label Switching

## MPLSのトラヒック・エンジニアリング

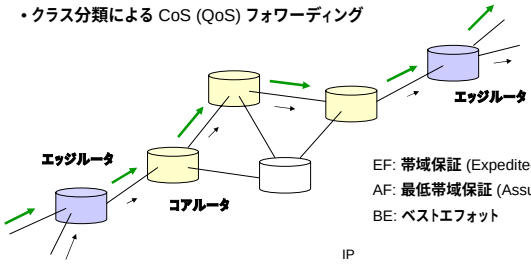
- 最小ホップ数以外のメトリックの活用



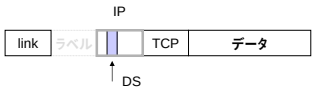

トラヒックエンジニアリング：負荷分散、高速迂回、…

## 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

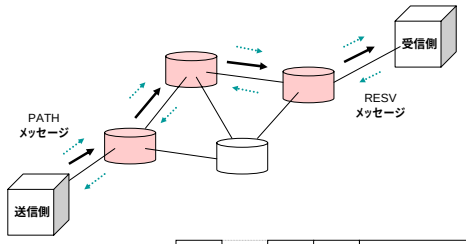



EF: 帯域保証 (Expedited)
AF: 最低帯域保証 (Assured)
BE: ベストエフォット




## 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

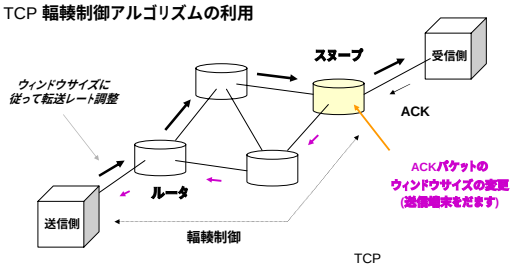


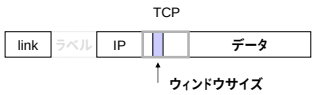


Integrated Services

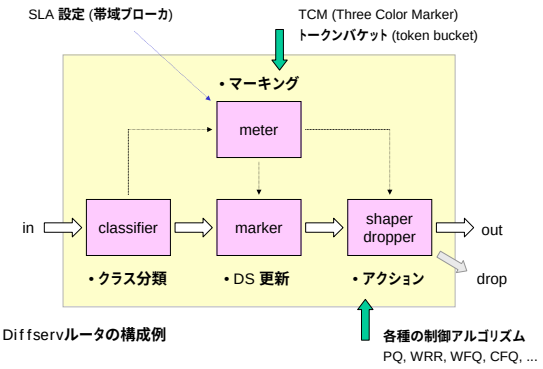
## TCP スヌーピング

- 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



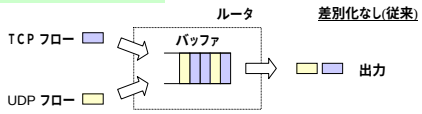




## フローの差別化 (1)

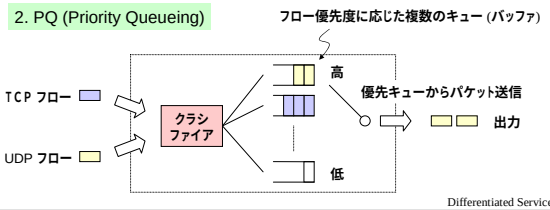



## フローの差別化 (2)

1. FIFO (First-In First-Out)



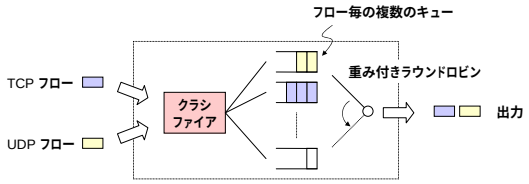

2. PQ (Priority Queueing)




Differentiated Services

## フローの差別化 (3)

3. WRR (Weighted Round Robin)

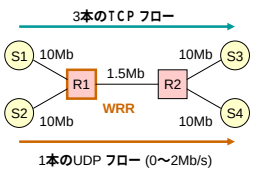



- PQ: 優先キューが空になるまで非優先パケットは送出されない (欠点)
- WRR: 既定の個数のパケットを送出すると非優先パケットを送出する (改善)

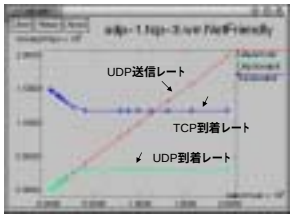
Differentiated Services

# フローの差別化 (4)

• WRR の効果



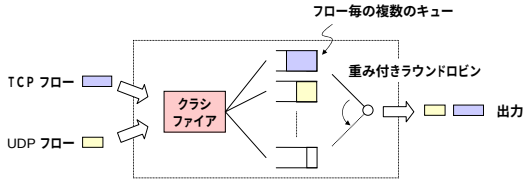

3本のTCPフローと1本のUDPフローがほぼ均等に帯域をシェアしている。
クラス許容量を超えたUDPは廃棄。




ネットワークシミュレータ

# フローの差別化 (5)

4. DRR (Deficit Round Robin)

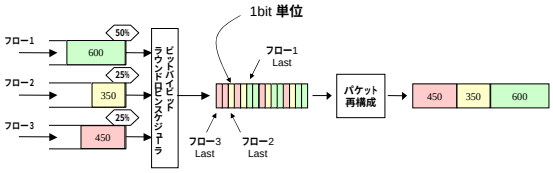



• WRR: 可変長パケットが来ると、長いパケットが優先 (欠点)
• DRR: パケットの「個数」ではなく、「バイト数」で重み付け (改善)



# フローの差別化 (6)

5. WFQ (Weighted Fair Queuing)



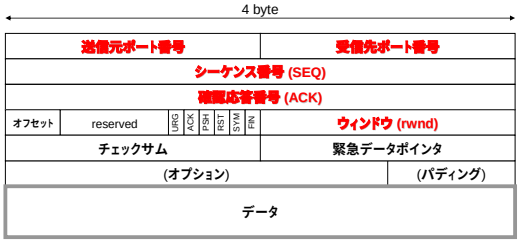

• WRR/DRR:「個数/帯域幅」は weighted fair だが、「到着時間」は not fair
• WFQ:「帯域幅」だけではなく、「到着時間」も weighted fair



# TCP

Transport Control Protocol

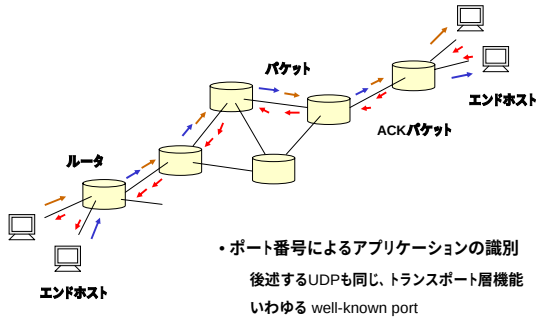
# TCP ヘッダ




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

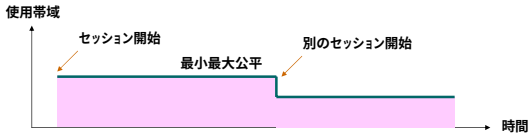



• ポート番号によるアプリケーションの識別
後述するUDPも同じ、トランスポート層機能
いわゆる well-known port

# TCPにおけるフロー制御

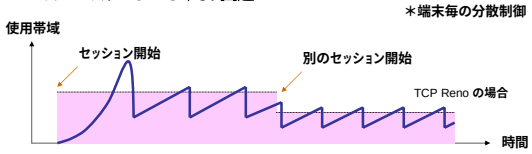
**理想:**

＊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

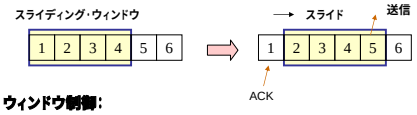
| | 要点 |
|---|---|
| TCP Tahoe | スロースタート + ふくそう回避 + 高速再送 |
| TCP Reno | Tahoe + 高速回復 |
| TCP Vegas | RTT (round trip delay) ベースのふくそう制御 |
| TCP SACK | Reno + 選択的再送 (selective repeat) |

- 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

# 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 個のパケットを送信する
  受信者が ACK を返すとウィンドウがスライドして次パケットが送出される
  しばしば n 個のパケット毎に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 (advertizement window)
  受信者が要求するセグメント (パケット) 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 (送信パケット数) 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;

V.Jacobson: "Congestion Avoidance and Control," SIGCOM'88.

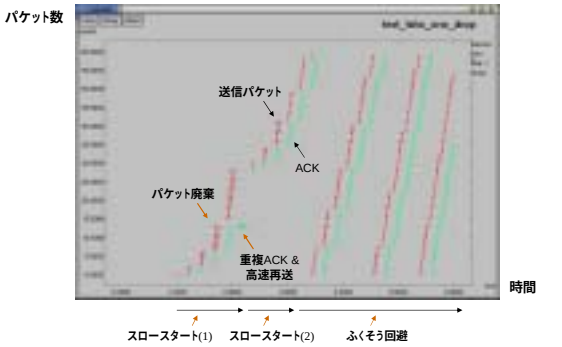
# TCP Tahoe (2)

- **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)




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; ⇐ ふくそう回避モードから再開 (ssthresh < cwnd)

  安全な値　　重複 ACK 分 (ACK が正しく返っている)

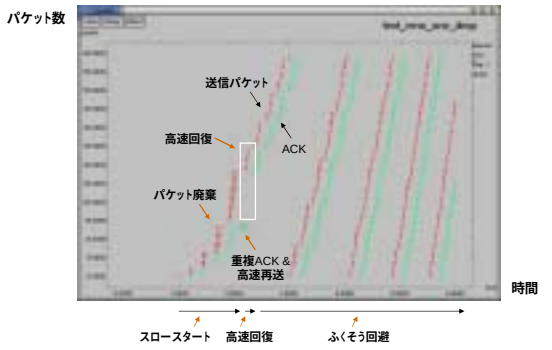
  if ( 重複 ACK 数 > cwnd/2 )

  --- 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)




NS (Network Simulator) によるシミュレーション例

# TCP Vegas (1)

- **Reno の問題点:**

  故意にパケット廃棄を発生させて最適なスループットを探っている。

  パケット廃棄を起こさなければ、スループットはもっと上がるはず。

- **ラウンドトリップ遅延 (RTT) に基づくふくそう回避:**

$$Diff = \frac{cwnd}{RTT\_\min} - \frac{cwnd}{RTT\_current}$$

⇐ ネットワーク内バッファの見積もり (未到達セグメント量)

最大送信レート　　実際の送信レート

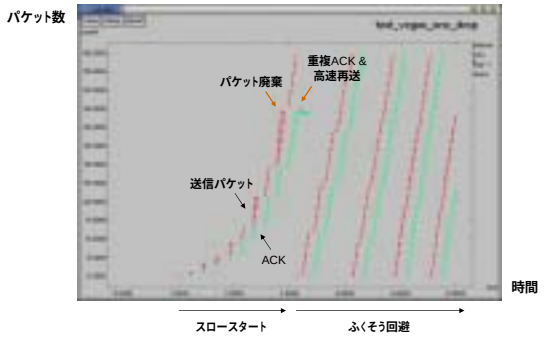
$$cwnd = \begin{cases} cwnd + 1 & (Diff < \alpha) \\ cwnd & (otherwise) \\ cwnd - 1 & (Diff > \beta) \end{cases}$$

⇐ ネットワーク内バッファの使用量が一定になるように制御

一定時間毎 ( RTT) に cwnd の値を更新

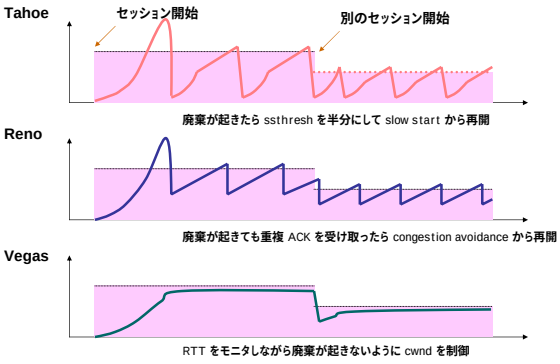
- **ラウンドトリップ遅延 (RTT) に基づくスロースタート:**

# TCP Vegas (2)




NS (Network Simulator) によるシミュレーション例

# 直感的な比較




# UDP

User Datagram Protocol

# UDP ヘッダ

4 byte

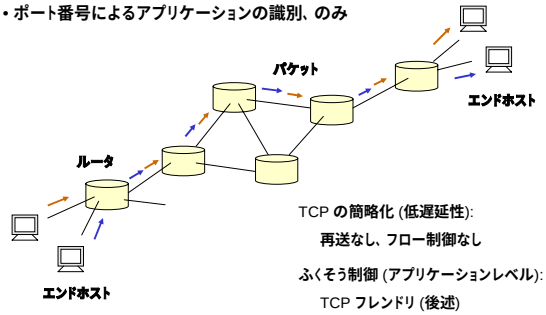
| 送信元ポート番号 | 受信先ポート番号 |
|---|---|
| UDP パケット長 | チェックサム |
| データ (+ RTPヘッダ) | |

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

・再送を行な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 は何も行なわないため、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) |

低遅延性と信頼性のトレードオフ