画像情報特論 (9)

- アダプテーション (3) ふくそう制御

  - TCPフレンドリ
  - RTCP

2001.06.12

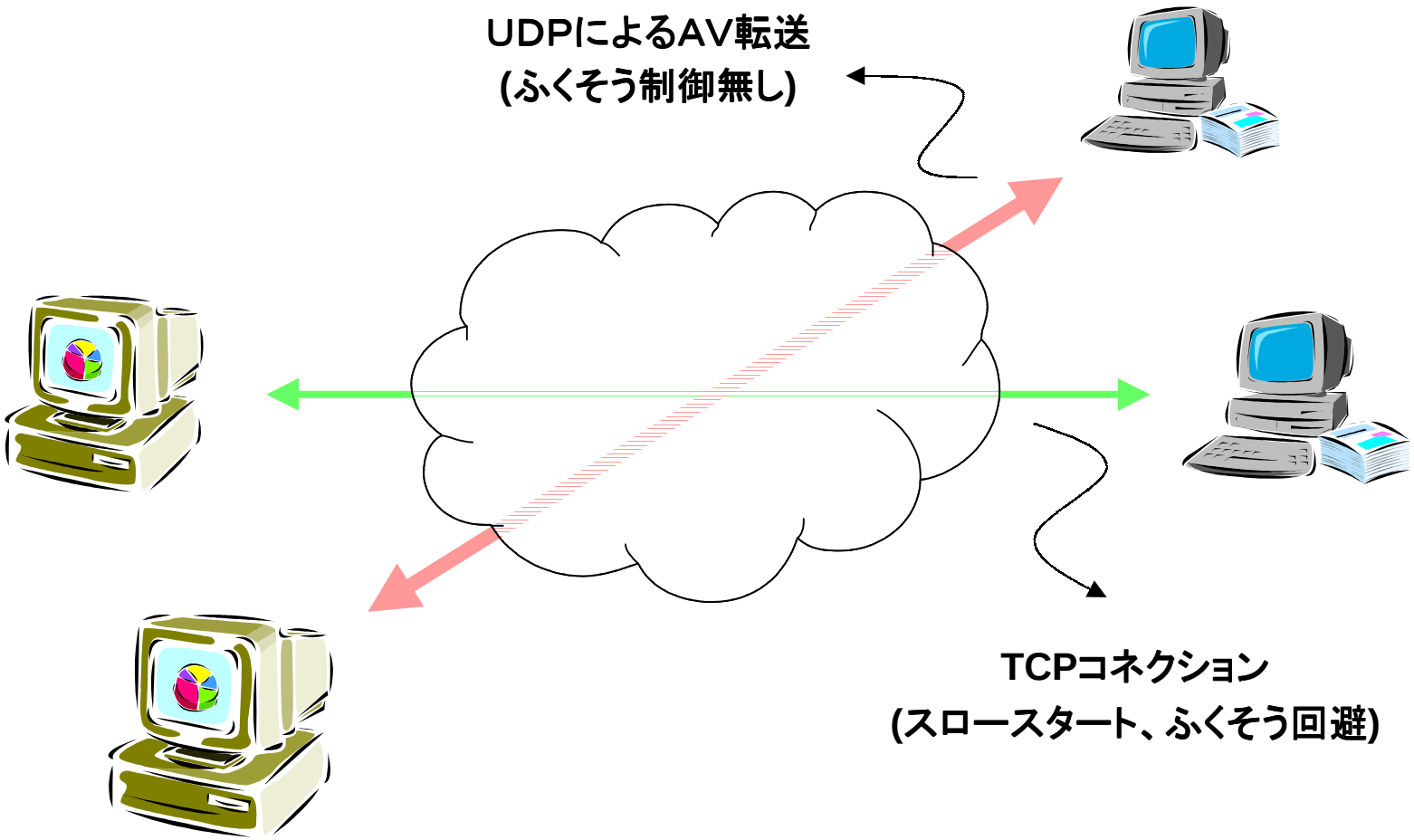
電子情報通信学科　甲藤二郎

E-Mail: katto@katto.comm.waseda.ac.jp

ふくそう制御と
TCPフレンドリ

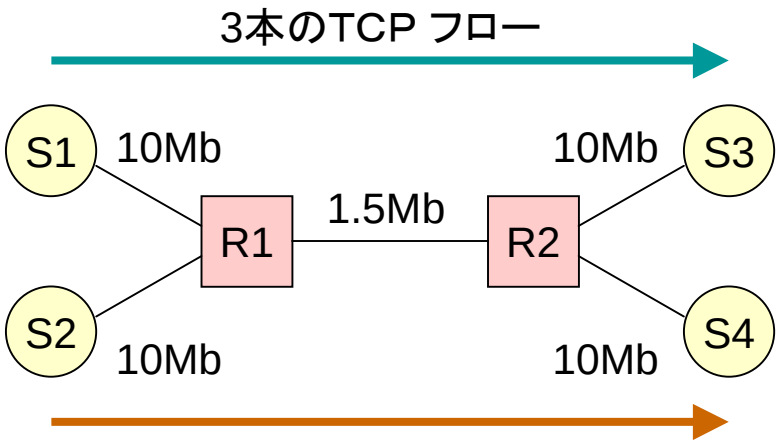
ふくそう制御の必要性 (1)

- UDP 帯域が増えると TCP 帯域はどうなるか？

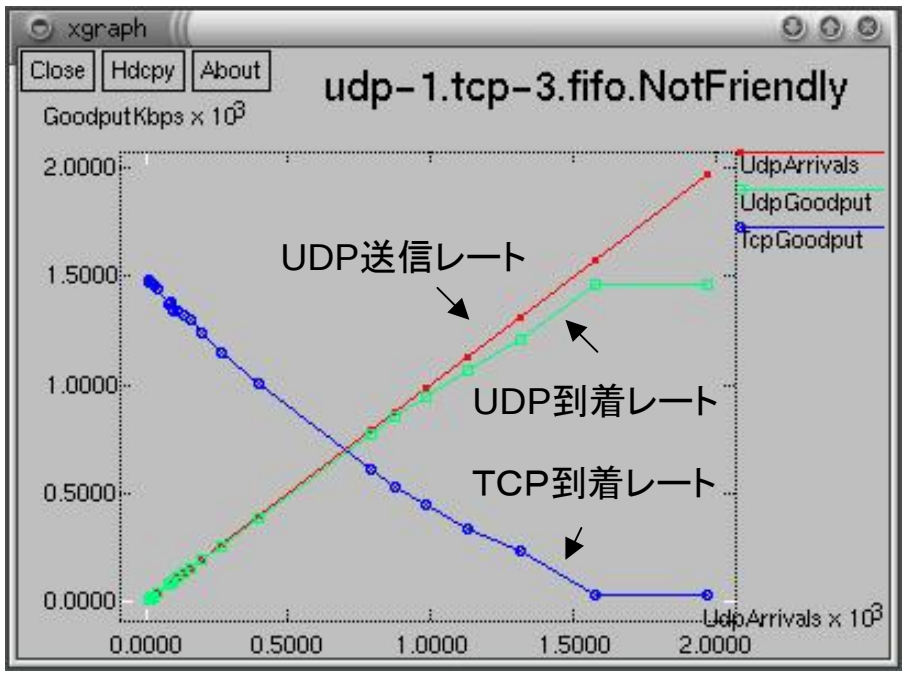


ふくそう制御の必要性 (2)

- UDP 帯域が増えると TCP 帯域はどうなるか？



UDP の送信レートの増加に伴い、
TCP のスループットが低下する。
リンク容量を越えた UDP は廃棄。

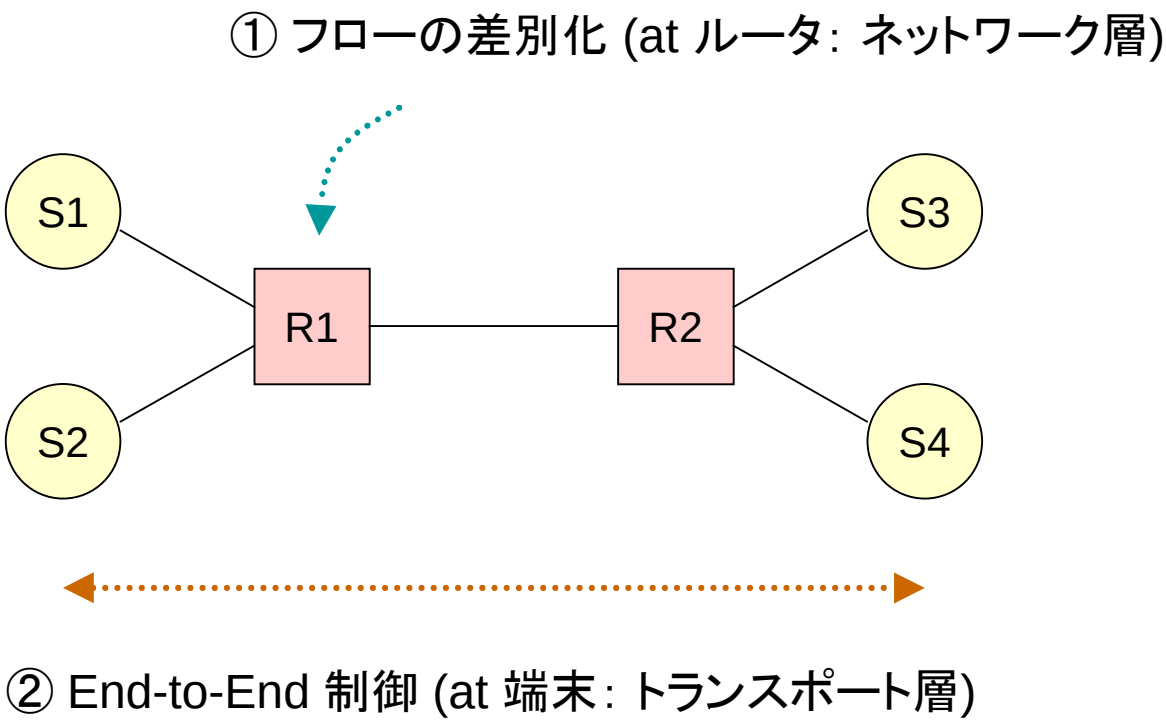


ネットワークシミュレータ

S.Floyd: “Promoting the Use of End-to-End Congestion Control,” IEEE/ACM Trans on Networking, 1999.

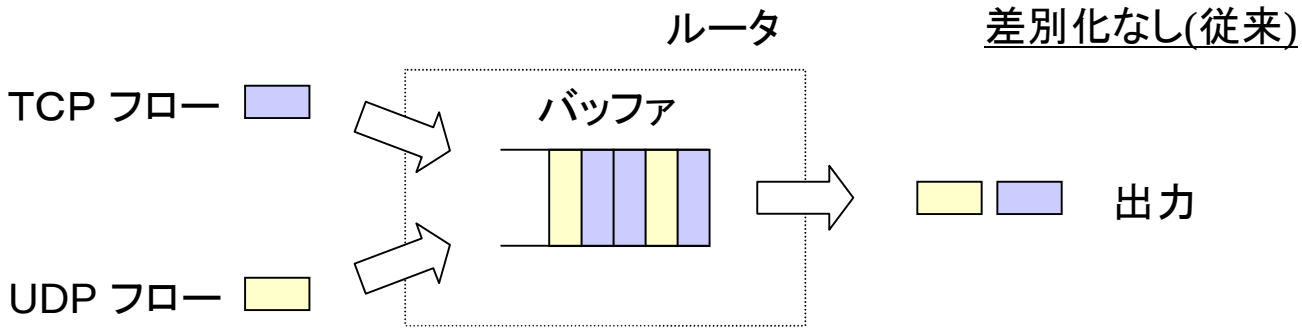
ふくそう制御の必要性 (3)

- どこで解決するか？

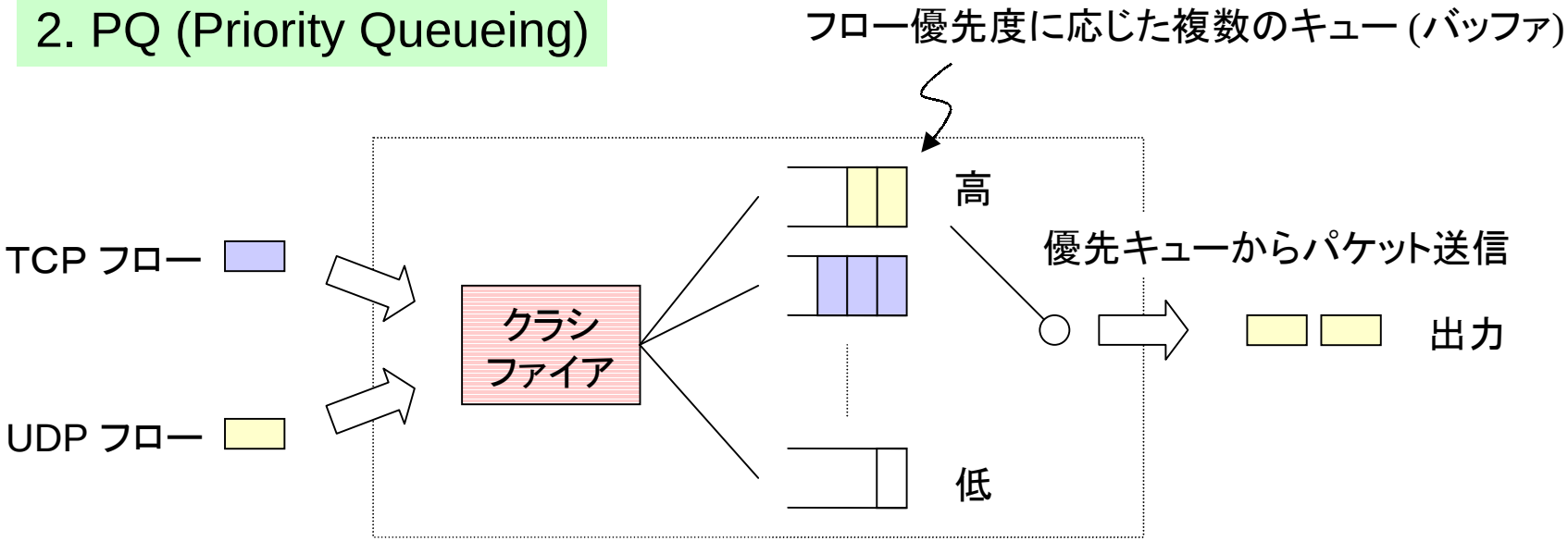


フローの差別化 (1)

1. FIFO (First-In First-Out)

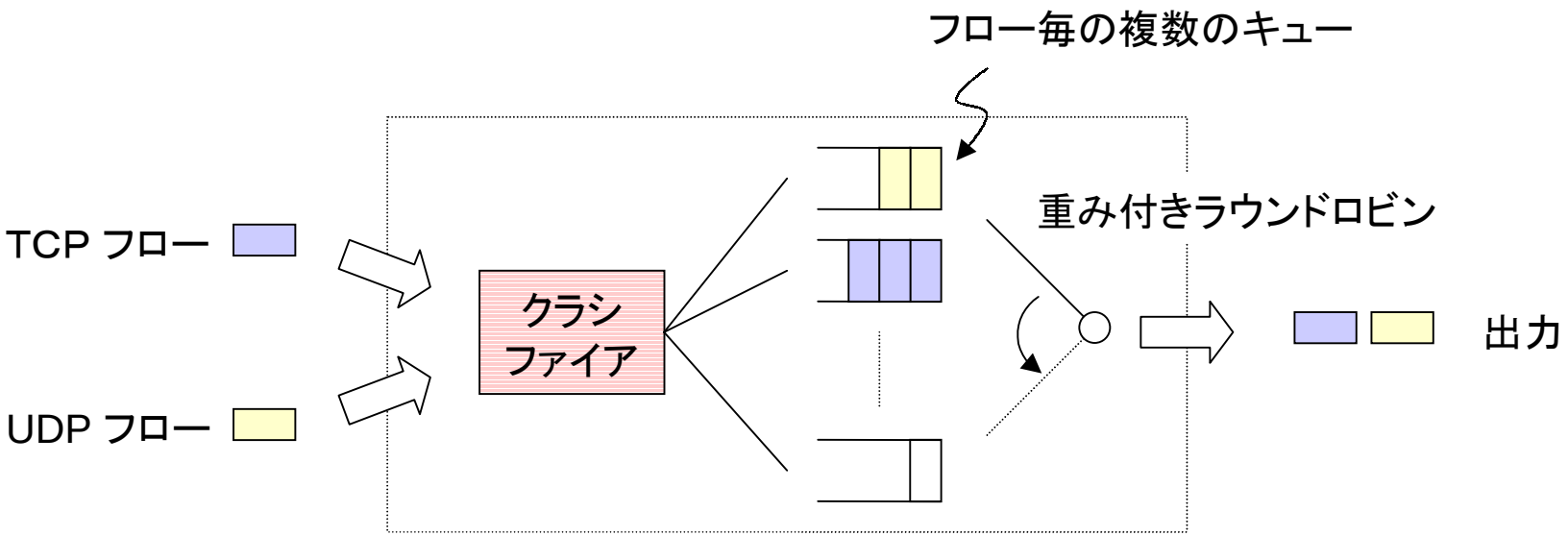


2. PQ (Priority Queueing)



フローの差別化 (2)

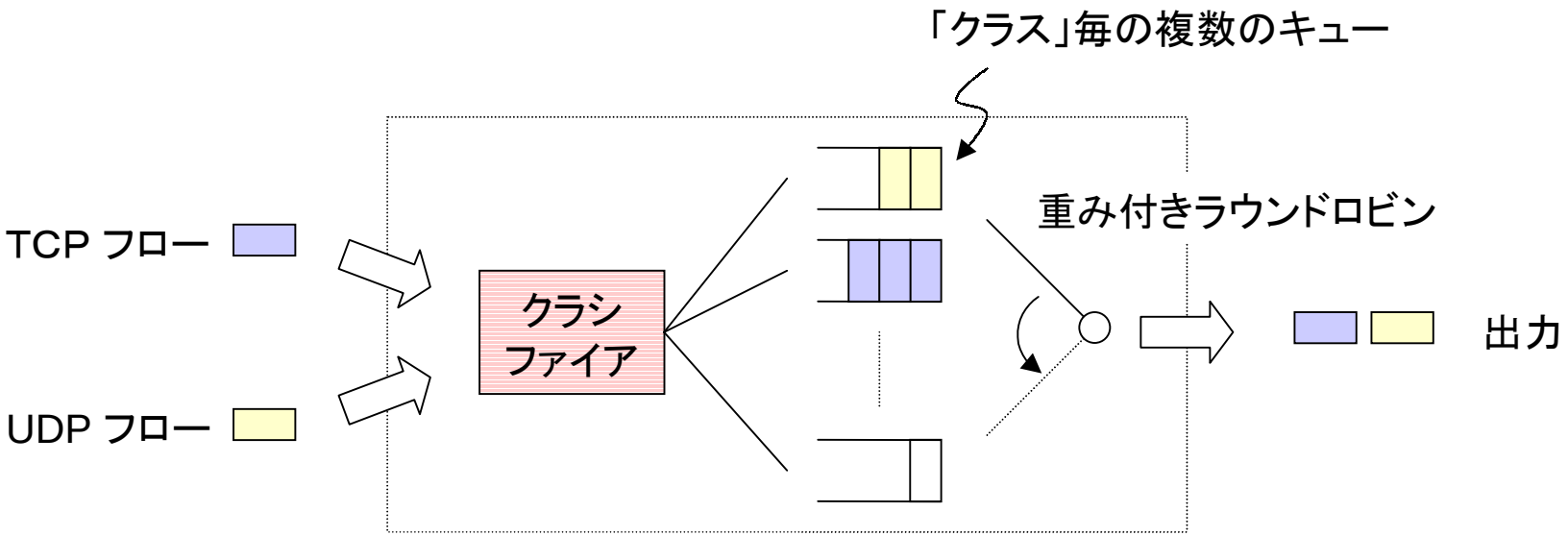
3. WRR (Weighted Round Robin)



- 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 (3)

4. CBQ (Class Based Queuing)

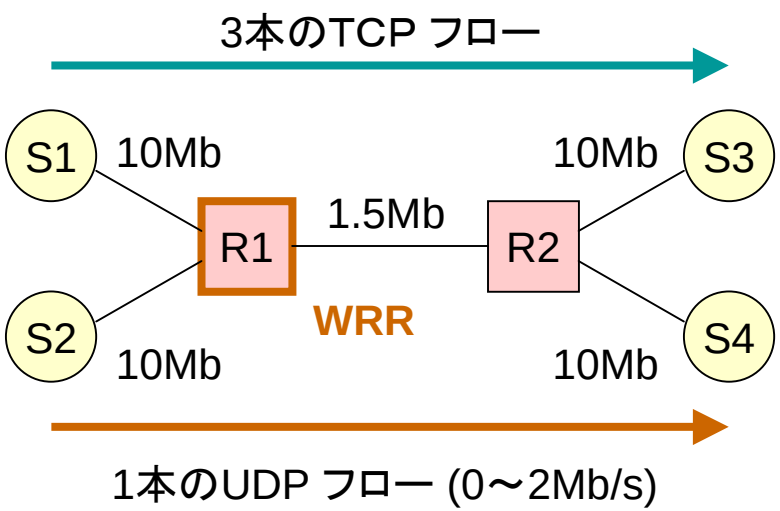


- フローを集約した「クラス」単位のWRR
- ラウンド周期毎の各キューの出力トラヒック量をバイト数で重み付け

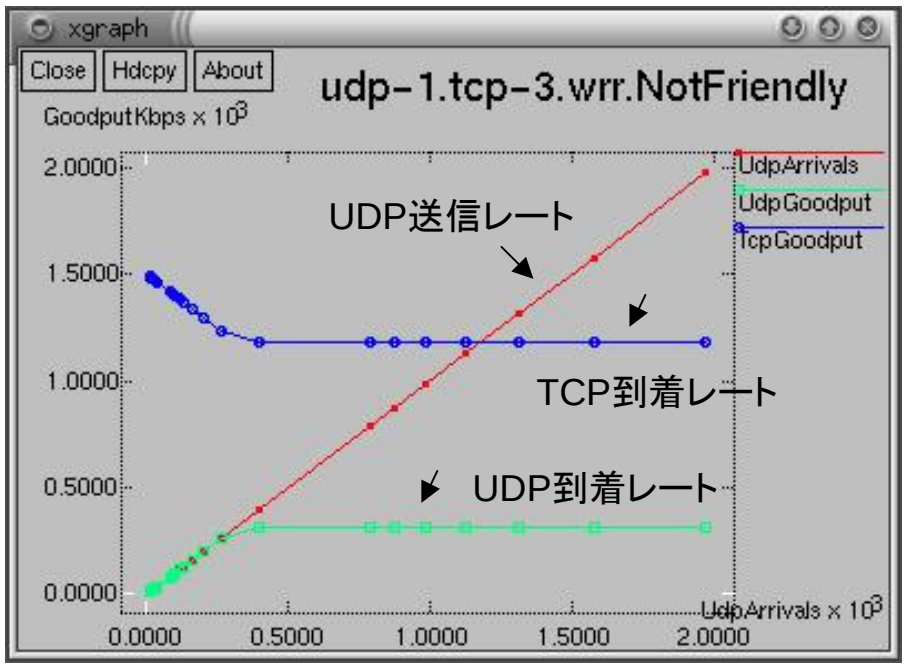
＊このほか WFQ (Weighted Fair Queuing) が著名

フローの差別化 (4)

- WRR/CBQ 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

S.Floyd: “Promoting the Use of End-to-End Congestion Control,” IEEE/ACM Trans on Networking, 1999.

フローの差別化 (5)

- フローの差別化 (Diffserv) ですべて解決するか？

Diffserv によって、TCP と UDP の公平性はある程度確保される。

UDP フロー間の公平性はなんら解決していない (別途申告が前提)。

申告を超えた UDP パケットは廃棄され続ける (フィードバック制御が必要)。

すべてのルータが Diffserv 対応になるにはまだ時間が必要。

いずれにせよ、End-to-End 制御による UDP フローの帯域制御が必要。

TCPフレンドリ (アプリケーションによる TCP、UDP フローの共存メカニズム)

TCPフレンドリ (1)

- TCP と UDP をどのように共存させるか？

UDP レートを TCP レートと等しくなるように符号量制御する。

方法1: UDP に対して TCP と同じふくそう制御メカニズムを適用する。

問題：レート変動が激しすぎて、AV アプリケーションには適用できない。

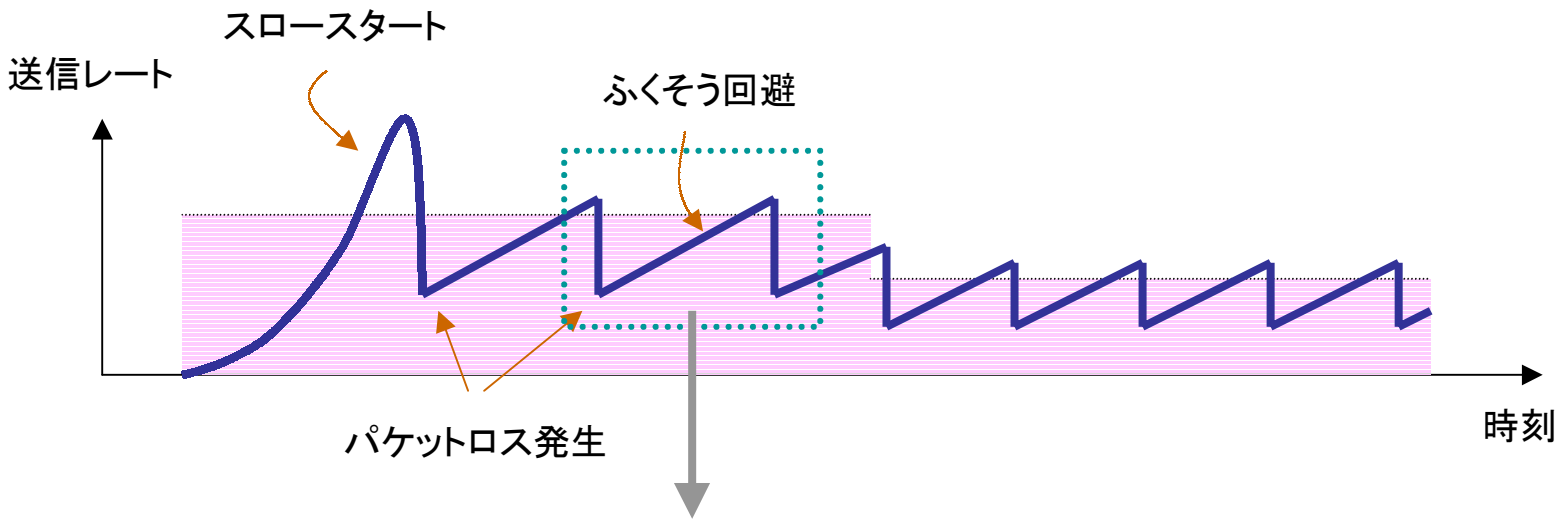
方法2: 測定可能なパラメータから TCP と等価なレートを見積もり、そのレートに適合するように UDP フローを制御する。

問題：TCP と等価なレートをどのように推定するか？

TCPフレンドリ (2)

- TCP のモデル化

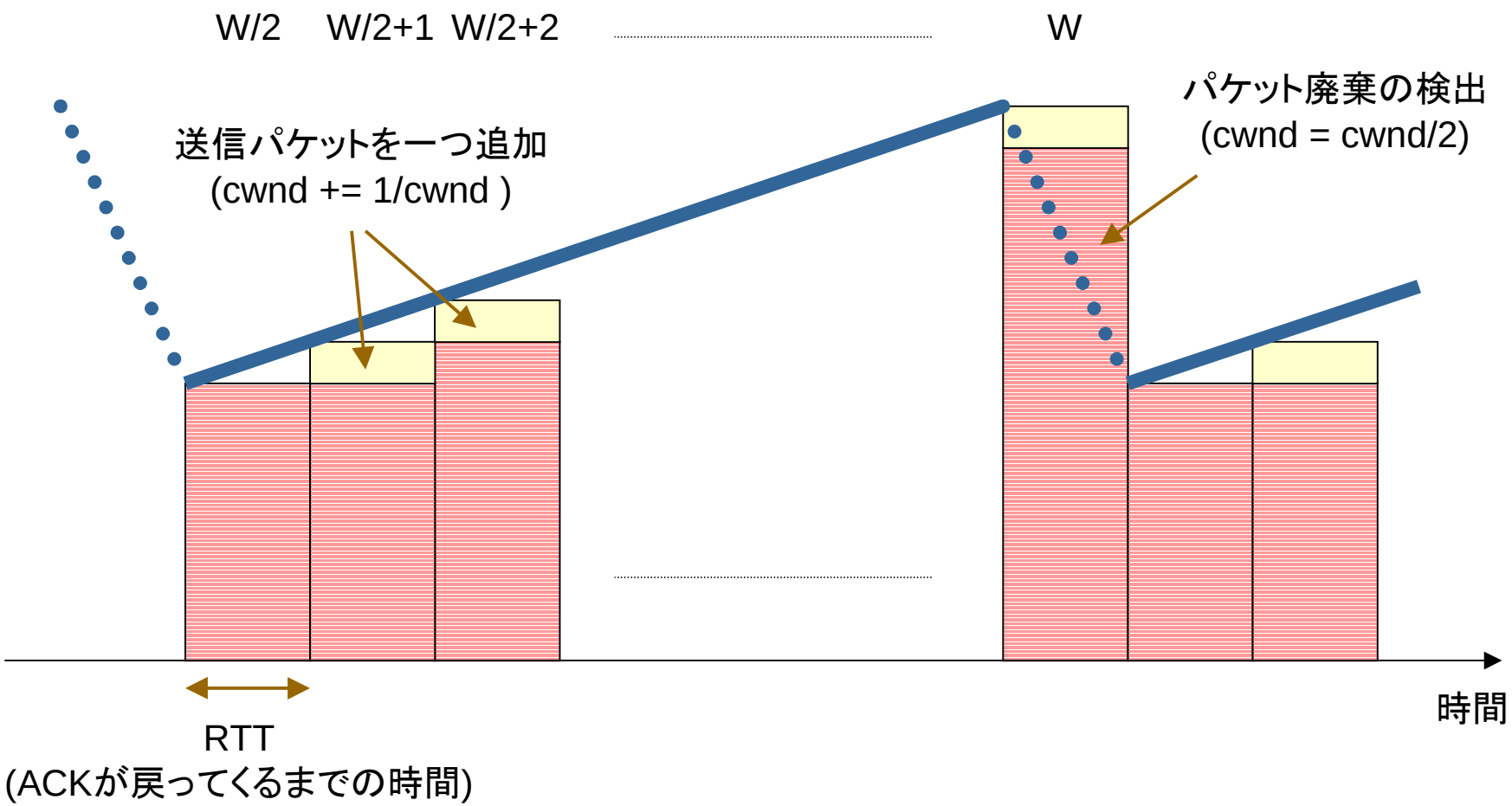
TCP Reno の場合



TCPの定常状態における
ふくそう回避のふるまいを
モデル化

TCPフレンドリ (3)

- ふくそう回避アルゴリズムのモデル化: モデル



第3回講義資料 (TCP/UDP) 参考.

TCPフレンドリ (4)

- ふくそう回避アルゴリズムのモデル化：定式化

パケット廃棄の発生間隔

$$\frac{W}{2}\cdot RTT$$

その期間の送信パケット数

$$\frac{W}{2}+\left(\frac{W}{2}+1\right)+\cdots+W=\frac{3}{8}W^2$$

パケット廃棄率 p

$$p=\frac{8}{3W^2}$$

送信レート R

$$R=\frac{\frac{3}{8}W^2\cdot B}{\frac{W}{2}\cdot RTT}\approx\frac{1.22\cdot B}{RTT\cdot\sqrt{p}}$$

RTT: ラウンドトリップ遅延、W: パケット数、B: パケットサイズ

S.Floyd: “Promoting the Use of End-to-End Congestion Control,” IEEE/ACM Trans on Networking, 1999.

TCPフレンドリ (5)

- フロー制御の指針

(1) ラウンドトリップ遅延 (RTT)、パケット廃棄率 (p)、パケットサイズ (B) を測定

(2) 次式に従って目標送信レート (R) を更新

$$R \leq \frac{1.22 \cdot B}{RTT \cdot \sqrt{p}}$$

(3) 目標送信レート (R) を超えないように符号量制御

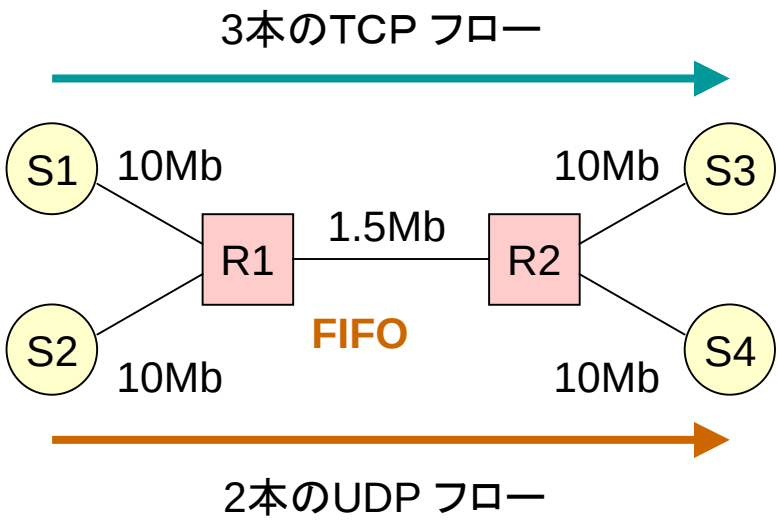
(注1) RTT の測定は RTCP の使用が前提 (UDP 自体は ACK を返さない)

(注2) パケットサイズ B は既定値 (576 byte 程度) とすることが多い (安全側)

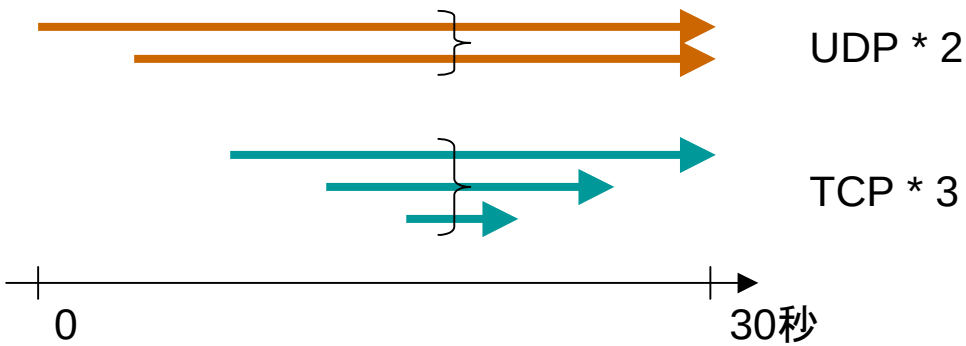
TCPフレンドリ (6)

- TCP フレンドリの効果: シミュレーション条件

トポロジー



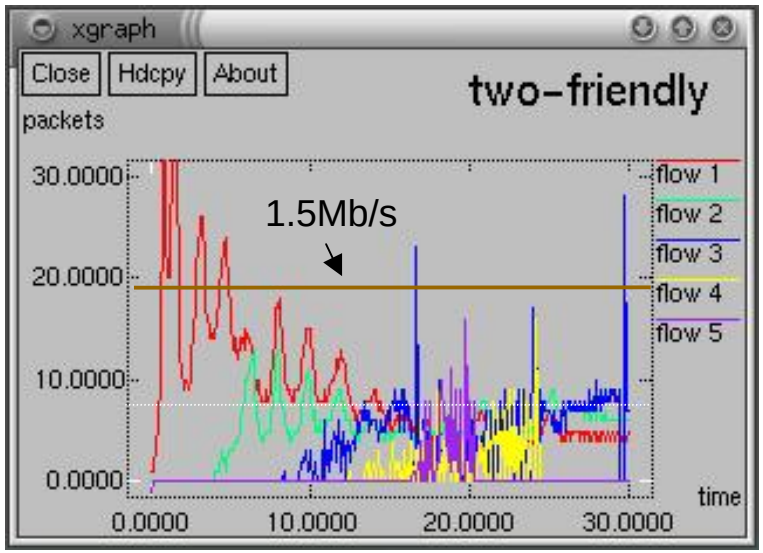
スケジュール



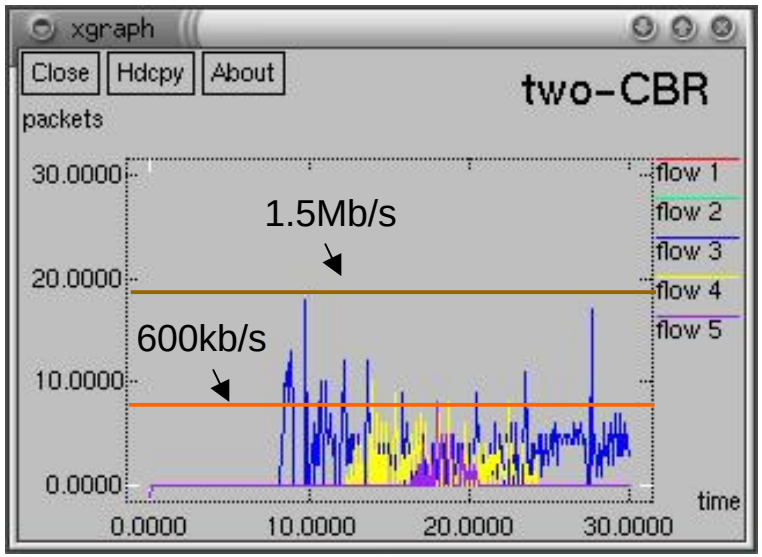
TCPフレンドリ (7)

- TCP フレンドリの効果: シミュレーション結果

TCPフレンドリ対応



TCPフレンドリ未対応 (600kb/s * 2)



(0.1秒毎の送信パケット数のカウント)

	UDP#1	UDP#2	TCP#1	TCP#2	TCP#3
TCPフレンドリ	698kb/s	459kb/s	352kb/s	159kb/s	292kb/s
CBR	600kb/s	600kb/s	234kb/s	111kb/s	100kb/s

TCPフレンドリ (8)

- 最近の TCP フレンドリ

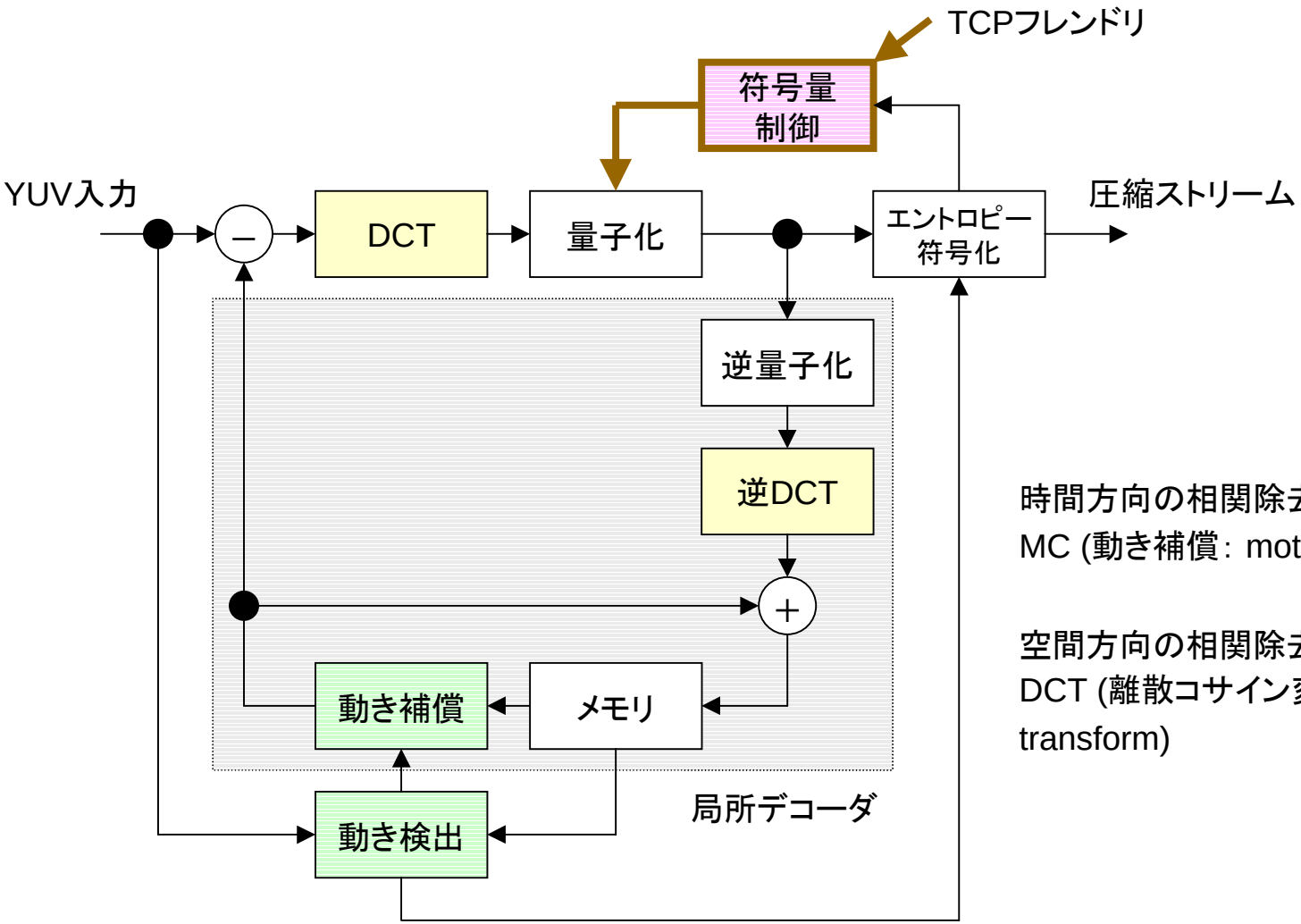
より厳密な TCP のモデル化

Equation-Based Congestion Control　　http://www.aciri.org/tfrc/

勉強中 ...

符号量制御 (1)

- 目標レートに合わせた量子化ステップサイズの制御



時間方向の相関除去:
MC (動き補償: motion compensation)

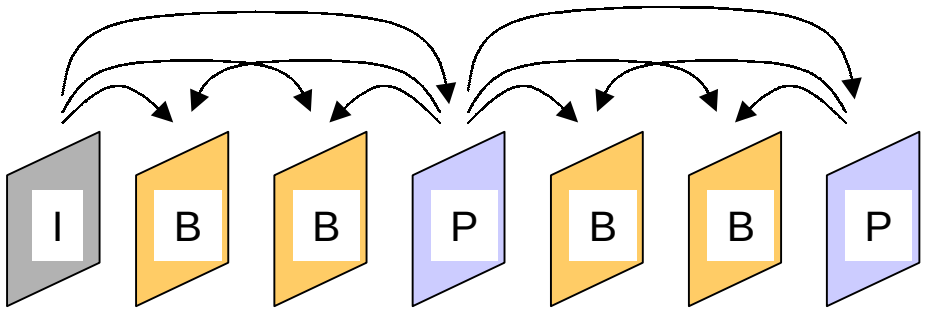
空間方向の相関除去:
DCT (離散コサイン変換: discrete cosine transform)

符号量制御 (2)

- 目標レートと量子化ステップサイズの関係

$$\log R = a \cdot \log Q + b \quad \Longleftrightarrow \quad R \cdot Q = const = X_{\{I,P,B\}}$$

ピクチャタイプ (I, P, B) 毎にほぼ一定



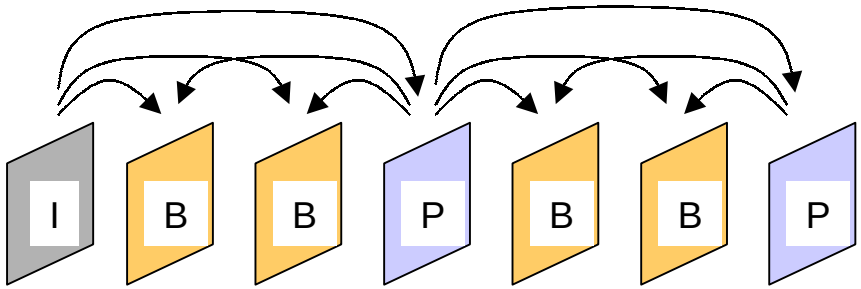
$X_{\{I,P,B\}}$ を事前に測定

目標レート R 確定 → 量子化ステップサイズ Q 確定

J.Katto et al: “Mathematical Analysis of MPEG Compression Capability ...,” IEEE ICIP’95, 1995.

符号量制御 (3)

- TM5 アルゴリズム: ピクチャタイプに応じたレート配分



$$R_I = \frac{R_{remain}}{N_I + \left(\frac{X_P}{X_I}\right)\cdot N_P + \left(\frac{X_B}{X_I}\right)\cdot N_B}$$

$$R_P = \frac{R_{remain}}{N_P + \left(\frac{X_B}{X_P}\right)\cdot N_B}$$

$$R_B = \frac{R_{remain}}{N_B + \left(\frac{X_P}{X_B}\right)\cdot N_P}$$

ピクチャ毎の符号量配分を決めるアルゴリズム

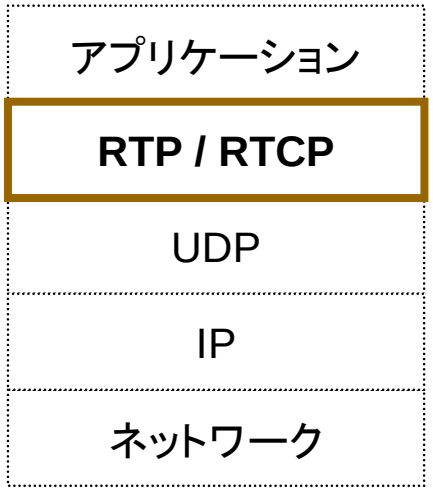
R_{remain}: 残余ビットレート (初期値: 目標レート R)
N_I, N_P, N_B: 各ピクチャの残余枚数
X_I, X_P, X_B: 一般的に $X_I > X_P > X_B$
1枚符号化する毎にパラメータを更新

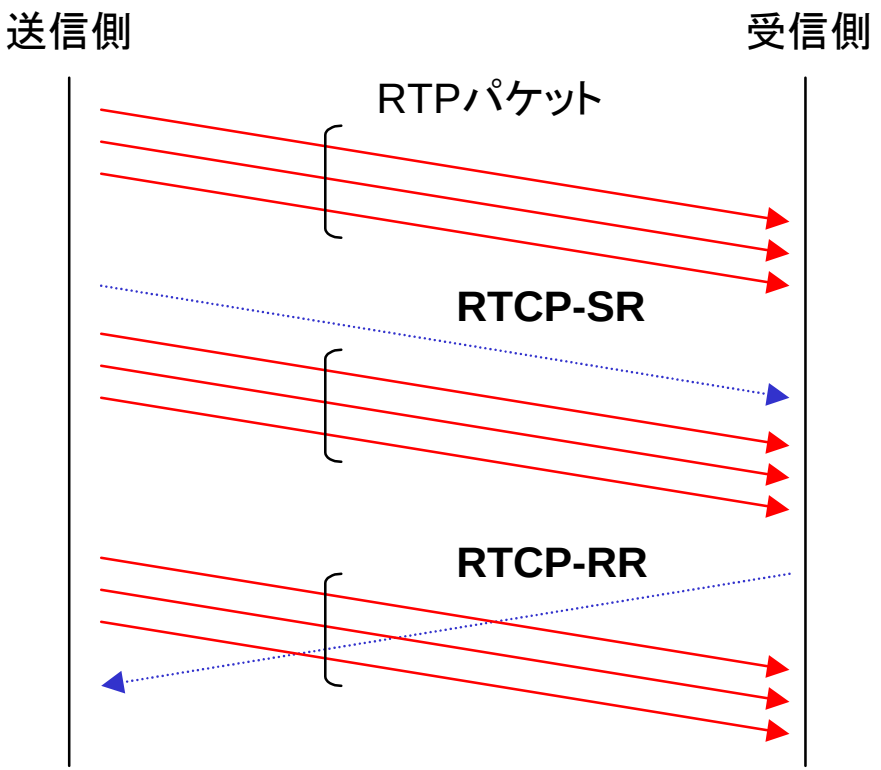
$R_I > R_P > R_B$ となる符号量配分 (準最適性の証明)

J.Katto et al: “Mathematical Analysis of MPEG Compression Capability ...,” IEEE ICIP’95, 1995.

RTCP とフロー制御

RTCP





- RTCP-SR: Sender Report
- RTCP-RR: RTCP Receiver Report

RFC 1889

RTCP (Sender Report)

v=2	P	RC	PT=SR=200	パケット長
送信元 SSRC 識別子				

sender report

NTP タイムスタンプ (MSB)
NTP タイムスタンプ (LSB)
RTP タイムスタンプ
送出パケット数
送出バイト数

report block * n

SSRC 識別子 #n
廃棄パケット数
シーケンスナンバの最大値
ジッタ遅延
最新の SR 受信時の NTP タイムスタンプ (LSR)
LSR から現在までの遅延 (DLSR)

RFC 1889

RTCP (Receiver Report)

v=2	P	RC	PT=SR=201	パケット長
送信元 SSRC 識別子				

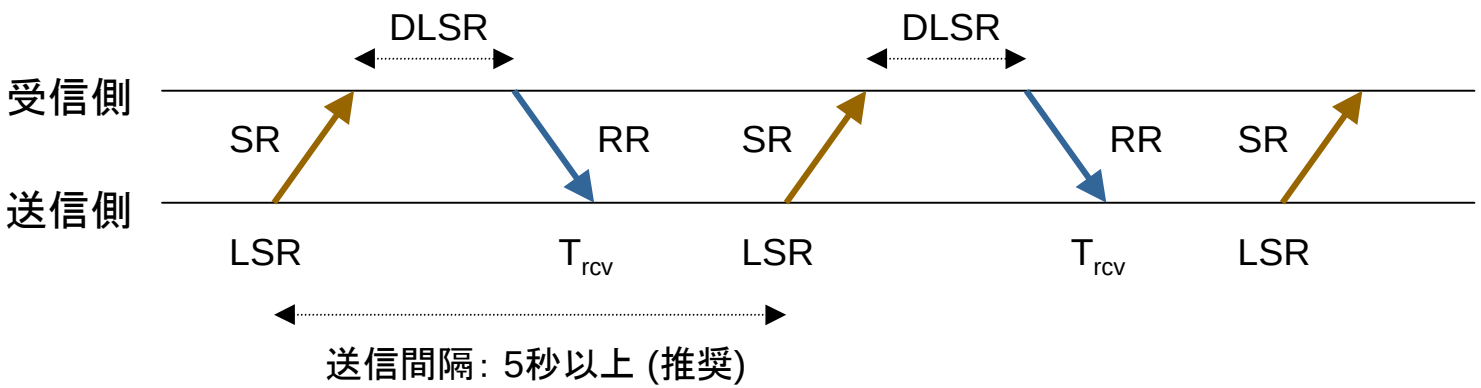
report block * n

SSRC 識別子 #n
廃棄パケット数
シーケンスナンバの最大値
ジッタ遅延
最新の SR 受信時の NTP タイムスタンプ (LSR)
LSR から現在までの遅延 (DLSR)

RFC 1889

Report Block

- 受信側から送信側に返される統計情報量



廃棄率: p = **廃棄パケット数** / 送信パケット数

ラウンドトリップ遅延: $RTT = T_{rcv}$ - **DLSR** - **LSR**

パケットサイズ: 送信側で測定可能

⇒ ふくそう制御 (TCPフレンドリ)

(注) RTCP 自体は、具体的なふくそう制御アルゴリズムは何も決めていない (標準というのはそういうもの)

RFC 1889