

## 画像情報特論 (5)

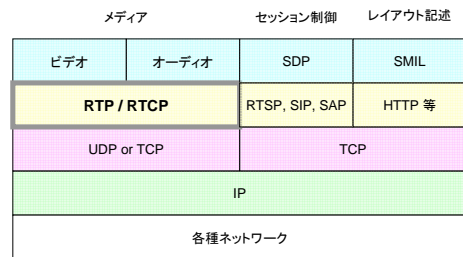
### - アダプテーション (1)

RTP/RTCP、メディア同期

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

## プロトコル階層

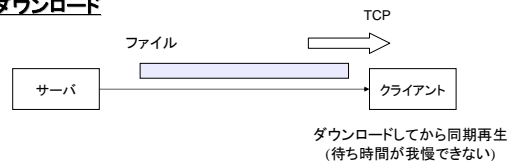
### • インターネットAVの仕組み



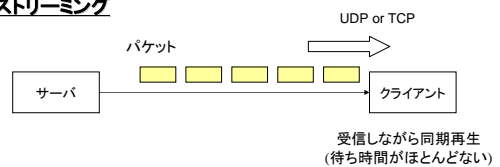
## アダプテーション

## ストリーミング

### ダウンロード



### ストリーミング



## ストリーミングの課題

インターネット: もともとリアルタイムメディア用のネットワークではない

### 1. 同期再生

到着時間がばらばらのパケットからどのように同期再生するか。

### 2. パケット廃棄対策

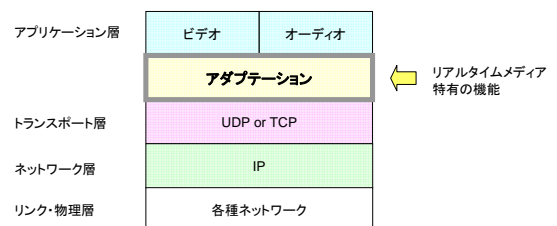
廃棄されたパケットの影響をどのように抑えるか。

### 3. ふくそう制御

レート上げすぎるとふくそうが起こり、下げると品質が劣化する。

## アダプテーション

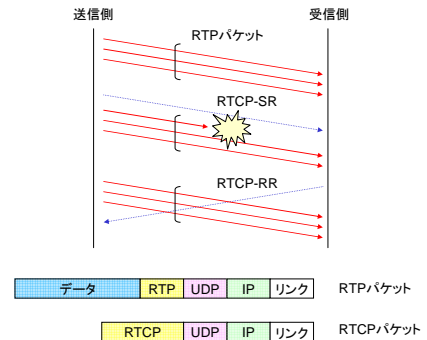
### • 同期再生、パケット廃棄対策、ふくそう制御の問題解決



# RTP/RTCP

RTP: Real-time Transport Protocol  
RTCP: RTP Control Protocol

## RTP・RTCPの基本的な使い方



## RTP



パケットタイプ: 転送データの符号化アルゴリズム  
 シーケンスナンバ: パケット廃棄の検出 (for パケット廃棄対策)  
 タイムスタンプ: 同期再生 (for メディア内同期)  
 Mビット: フレーム境界の通知  
 SSRC: ストリームの識別子 (セッション内でユニーク)

## パケットタイプ

PT (packet type)	encoding name	audio/video (A/V)	clock rate (Hz)	channels (audio)
0	PCMU	A	8000	1
2	G721	A	8000	1
3	GSM	A	8000	1
8	PCMA	A	8000	1
9	G722	A	8000	1
14	MPA	A	90000	
15	G728	A	8000	1
26	JPEG	V	90000	
31	H261	V	90000	
32	MPV	V	90000	
33	MP2T	AV	90000	
96~127	dynamic			

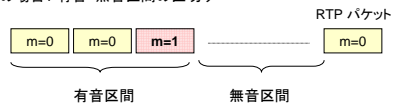
タイムスタンプの解像度  
 最近の符号化アルゴリズムは SDP を用いた動的割り当て

## M ビット

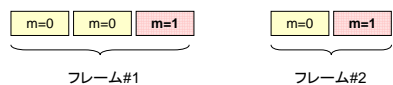
### • アルゴリズムに応じたパケット区切りの解釈

典型的な使い方:

音声の場合: 有音・無音区間の区切り



ビデオの場合: フレームの区切り



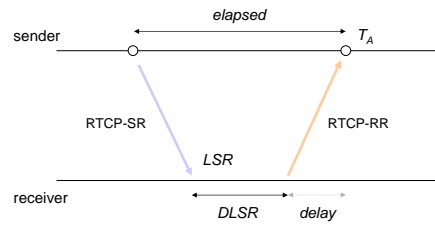
## RTCP-SR (Sender Report)



## RTCP-RR (Receiver Report)

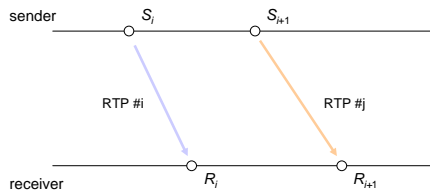
v=2	P	RC	PT=SR=201	パケット長
送信元 SSRC 識別子				
report block * n				
SSRC 識別子 #n				
瞬時廃棄率		累積廃棄パケット数		
シーケンスナンバーの最大値				
ジッタ遅延				
SSRC #n の最新のSR受信時の NTP タイムスタンプ (LSR)				
LSR から現在までの遅延 (DLSR)				

## ラウンドトリップ遅延の計算



- 送受信端末間のNTP同期が信頼できる場合:  $delay = T_A - DLSR - LSR$
- NTP同期が信頼できない場合:  $delay \approx (elapsed - DLSR) / 2$

## ジッタの計算



瞬時ジッタの計算:  $Jitter(i) = (R_{j+1} - R_i) - (S_{i+1} - S_i)$

平均ジッタの計算:  $Jitter_{avg} = (1 - \alpha) \cdot Jitter_{avg} + \alpha \cdot Jitter(i)$  ( $\alpha = 1/16$ )

## 交換情報の整理

	RTP	RTCP
メディア同期	RTP タイムスタンプ	NTP タイムスタンプ RTP タイムスタンプ
パケット廃棄対策	シーケンスナンバー ペイロードフォーマット (後述)	
フロー制御		パケット廃棄率 ラウンドトリップ遅延 ジッタ

## RTCP パケット一覧

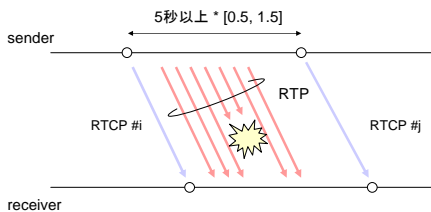
名称	目的
SR (Sender Report)	送信者からの NTP タイムスタンプと統計情報 (送信パケット数、送信バイト数) の通知 + 受信者でもある場合は Receiver Report
RR (Receiver Report)	受信者からの統計情報 (廃棄率、ジッタ、等) の通知
SDES (Source Description)	セッション参加者の情報 (CNAME、メールアドレス、等)
BYE	セッションからの離脱
APP (Application Specific)	アプリケーション拡張

## SDES アイテム一覧

SDES アイテム	識別番号	目的
END	0	SDES アイテムの終了
CNAME	1	参加者毎に固有の識別子 (例: foo@PC.waseda.ac.jp)
NAME	2	参加者の名前
EMAIL	3	参加者の電子メールアドレス
PHONE	4	参加者の電話番号
LOC	5	参加者の住所
TOOL	6	参加者の使用しているアプリケーション名
NOTE	7	参加者の状態 (例: 電話中で話せません)
PRIV	8	アプリケーション拡張

- RTP の SSRC: ストリーム毎に固有の識別子
- RTCP-SDES の CNAME: 参加者毎に固有の識別子

## RTCP の送信間隔

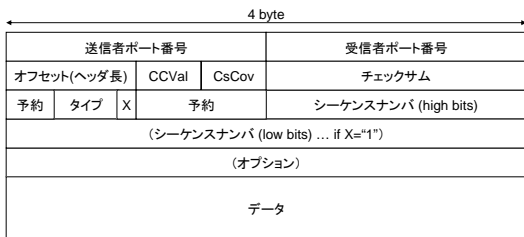


- RTCP の情報量は全体の5%を超えてはいけない
- 受信者数の増加と共に (マルチキャスト時) 送信間隔も増加させる

## DCCP

Datagram Congestion Control Protocol

## DCCP ヘッダ



CCVal (Congestion Control Value) : 別途既定されるプロフィールに応じて、送信者が輻輳制御情報を通知するために使用。

CsCov (Checksum Coverage) : チェックサムがパケットのどの部分までカバーするかを通知。通常 UDP では1ビットでも誤ったらパケットを廃棄するが、CsCov は無線応用を想定してデータ部のビット誤りを許容 ... UDP-Lite (RFC 3828) の考えを踏襲。

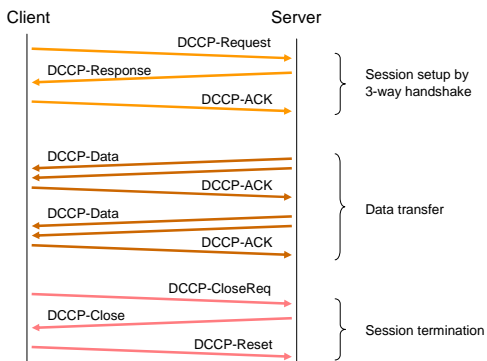
## DCCP パケット

- タイプフィールドと DCCP パケット

タイプ	DCCPパケット	目的
0	DCCP-Request	クライアントによる接続要求
1	DCCP-Response	接続要求に対するサーバの応答
2	DCCP-Data	アプリケーションデータの送信
3	DCCP-Ack	アプリケーションデータを伴わないACK
4	DCCP-DataAck	アプリケーションデータを伴うACK
5	DCCP-CloseReq	サーバからの接続終了要求
6	DCCP-Close	クライアントからの接続終了要求
7	DCCP-Reset	接続の終了通知
8	DCCP-Sync	障害発生時の同期回復要求
9	DCCP-SyncAck	同期回復要求に対するACK

データの送受信  
セットアップ  
障害対策  
終了処理

## DCCP 手順



## DCCP 輻輳制御

- CCID と DCCP の輻輳制御手段

CCID	輻輳制御手段
0	Reserved
1	Unspecified Sender-based Congestion Control
2	TCP-like Congestion Control (TCPと同様のAIMD制御: オンラインゲーム用)
3	TFRC Congestion Control (TCPフレンドリ制御: AV系アプリケーション用)

CCID=2: 2個のパケットを受信するたびにACKを返す ⇒ AIMD制御  
CCID=3: RTTごとにACKを返す ⇒ TFRC制御 (後日、説明)

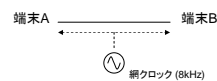
# メディア同期

バッファリング + タイムスタンプ

## 三種類の同期メカニズム

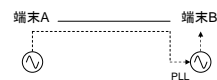
### (1) 共通クロック (狭帯域回線交換)

ネットワーク、あるいは外部から、「信頼できる共通クロック」を提供する → 狭帯域ネットワーク  
(例) 電話網、ISDN、モバイル



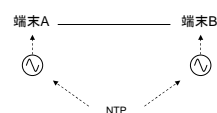
### (2) 搬送クロック (広帯域回線交換)

クロックを相手端末に搬送する。受信側は、PLL でクロックを生成 → 広帯域ネットワーク  
(例) ATM、デジタル放送

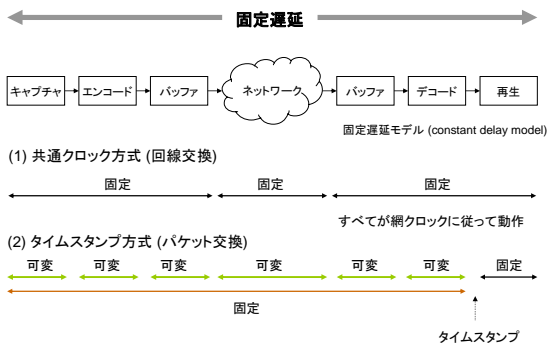


### (3) 自走クロック (インターネット)

まず、端末自身の自走クロックを信頼する。端末間同期はNTPを利用 → 精度に問題  
(例) インターネット



## 固定遅延モデル (1)



## 固定遅延モデル (2)

### • キャプチャから再生までの遅延時間が一定

#### (1) 共通クロック方式 (電話網、デジタル放送)

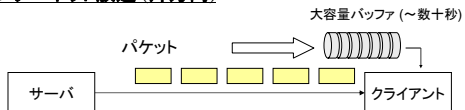
エンコード、デコード、ネットワークの処理時間 (遅延) が一定  
→ キャプチャから再生まで固定遅延

#### (2) 自走クロック方式 (インターネット)

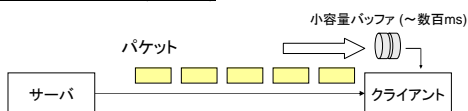
エンコード、デコード、ネットワークの処理時間 (遅延) が可変  
→ 「**バッファリング**」で遅延のばらつきを吸収  
→ 「**タイムスタンプ**」(キャプチャ時刻)で再生時刻を決定  
→ キャプチャから再生まで固定遅延

## バッファリング (1)

### インターネット放送 (片方向)



### インターネット電話 (双方向)



## バッファリング (2)

### • ジッタ (遅延のばらつき) の吸収

#### (1) インターネット放送

片方向なので、数十秒間のバッファリングを行ってもユーザは気にならない  
→ ユニキャスト時はTCPが利用可能 (パケット廃棄対策が不要)  
(注) マルチキャスト時はUDPを使用 (パケット廃棄対策が必要)

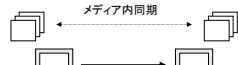
#### (2) インターネット電話

双方向なので、あまり長時間のバッファリングを行うと会話にならない  
→ TCPは利用困難  
→ UDPを使用 (一般的にパケット廃棄対策が必要)

## 三階層のメディア同期

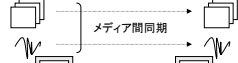
### (1) メディア内同期

一つのメディア (ビデオ、あるいはオーディオ) の同期再生の実現  
(手段) RTPタイムスタンプ (固定遅延モデル)



### (2) メディア間同期

複数のメディア (ビデオとオーディオ) の同期再生の実現 (リップシンク)  
(手段) RTCPによるタイムスタンプの関連付け

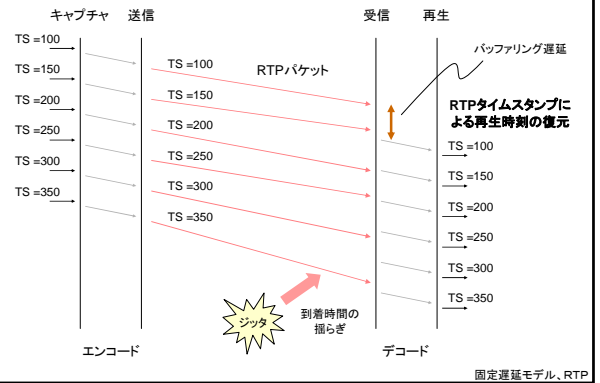


### (3) システム間同期

端末毎のシステム時刻の調整  
(手段) ネットワーク・タイム・プロトコル

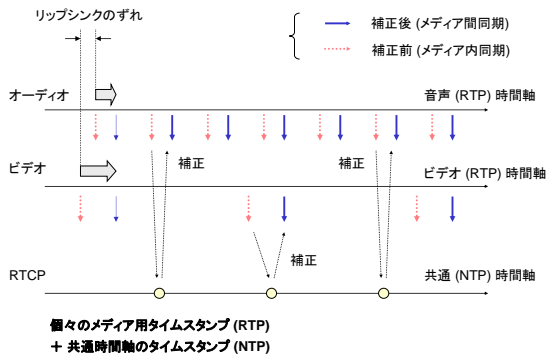


## メディア内同期



固定遅延モデル、RTP

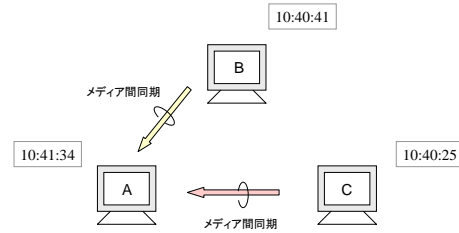
## メディア間同期



RTCP

## システム間同期

複数人による実時間会議の問題点



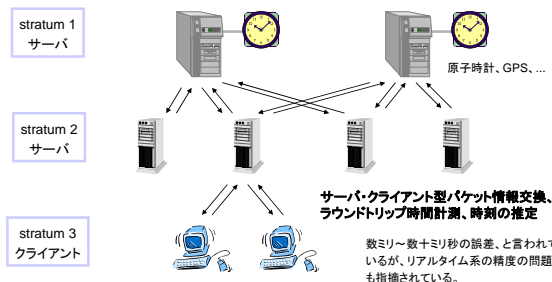
Bから来るメディアとCから来るメディアの時間軸を整理することができない  
→ 複数人が同時に話しても、それを同時に再生できない  
→ 結局、すべての端末の時刻をそろえるしかない

NTP

## NTP (Network Time Protocol)

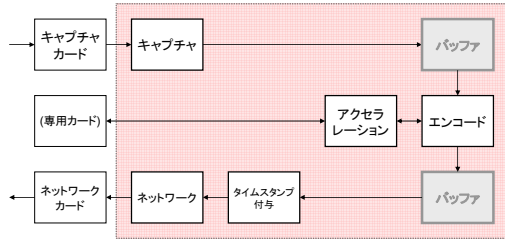
ネットワーク上の端末の時刻を合わせるためのプロトコル

(例) UNIX: xntpd, Windows: 桜時計、など...



## 実践編

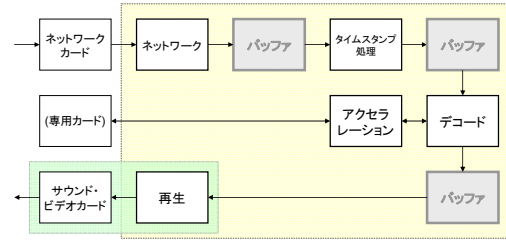
### メディア同期の実際 (1. 送信側)



バッファの活用 (実体はメモリ)

バッファ: 冗長構成 (ダブルバッファ、数十ミリ〜数百ミリ秒のバッファリング)

### メディア同期の実際 (2. 受信側)

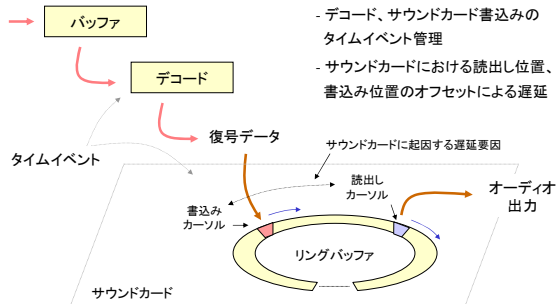


バッファの活用

バッファ: 冗長構成 (ダブルバッファ、数十ミリ〜数秒のバッファリング)

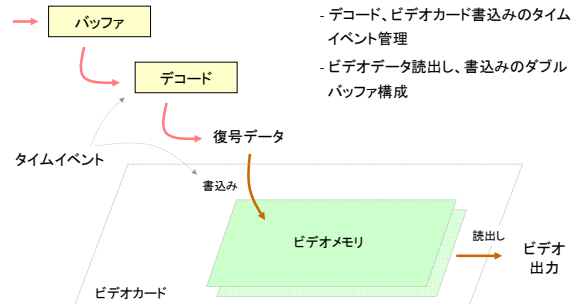
### メディア同期の実際 (3)

#### オーディオのストリーミング再生

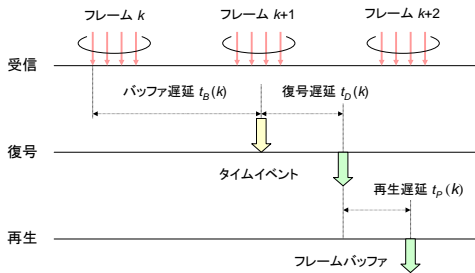


### メディア同期の実際 (4)

#### ビデオのストリーミング再生



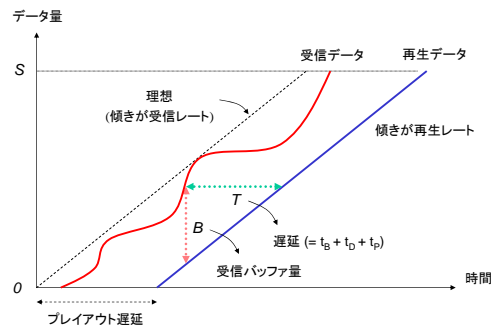
### メディア同期の実際 (5)



$t_b(k) = \text{variable}, t_d(k) + t_p(k) = \text{const}$

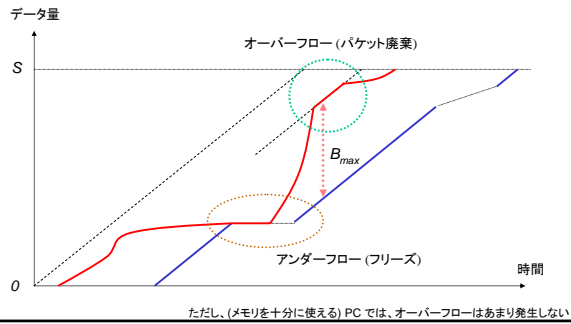
### メディア同期のモデル (1)

#### 定常モデル



## メディア同期のモデル (2)

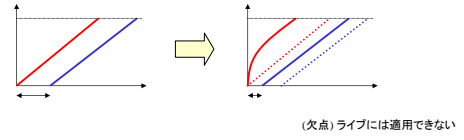
### 受信バッファのアンダーフローとオーバーフロー



## メディア同期のモデル (3)

### プレイアウト遅延・アンダーフロー対策

(1) データを送れるときに早めに送ってしまう (プリフェッチング)



(2) 送信レートを下げる (TCPフレンドリ: 次回参照)

