画像情報特論 (6)

- **アダプテーション** (1)

  RTP/RTCP、**メディア同期**

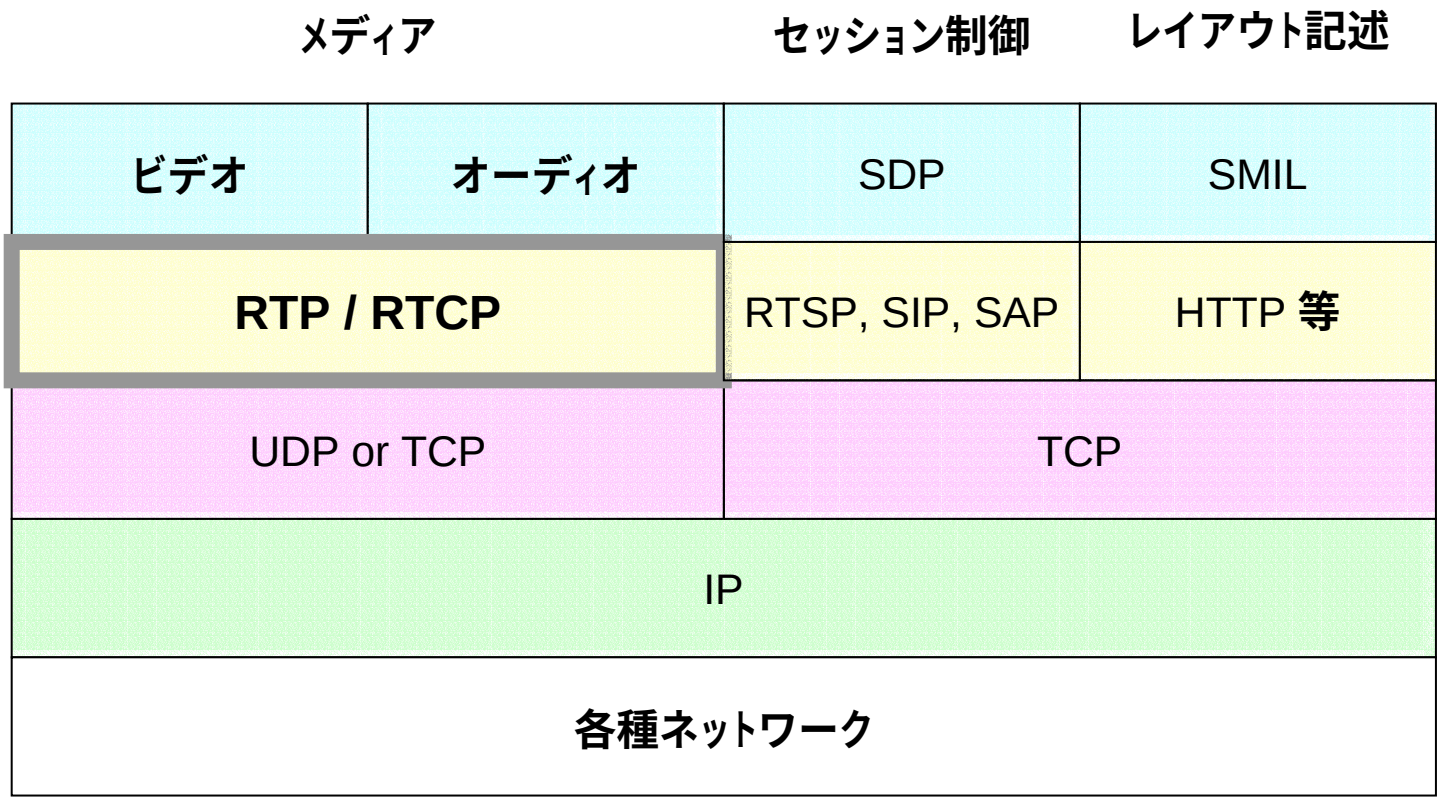
2003.05.30

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

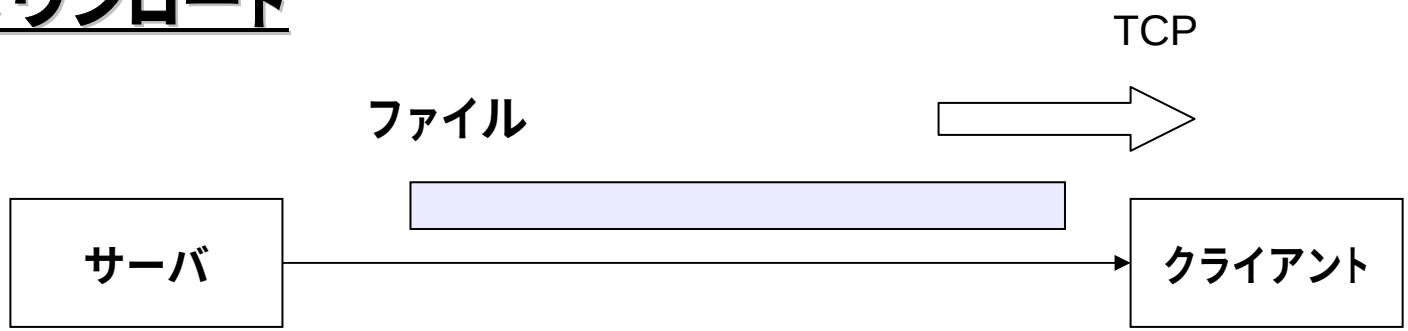
- インターネットAVの仕組



アダプテーション

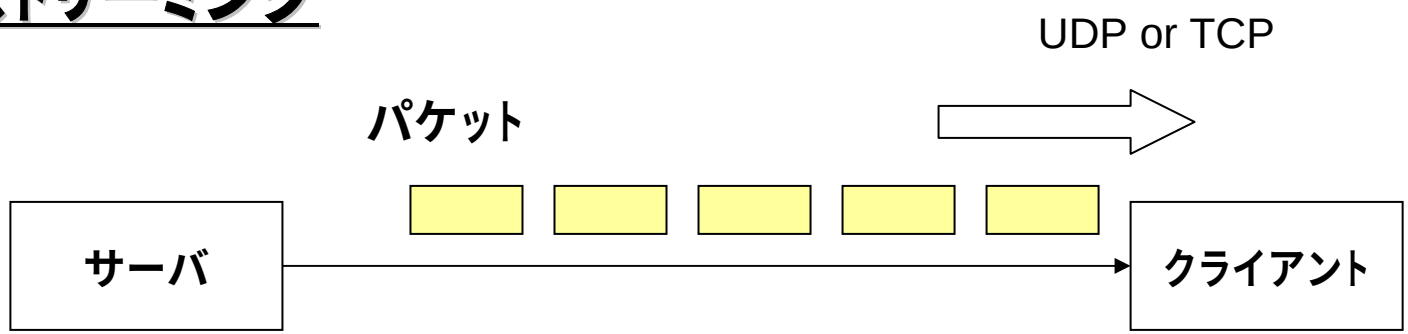
ストリーミング

ダウンロード



ダウンロードしてから同期再生
(待ち時間が我慢できない)

ストリーミング



受信しながら同期再生
(待ち時間がほとんどない)

ストリーミングの課

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**

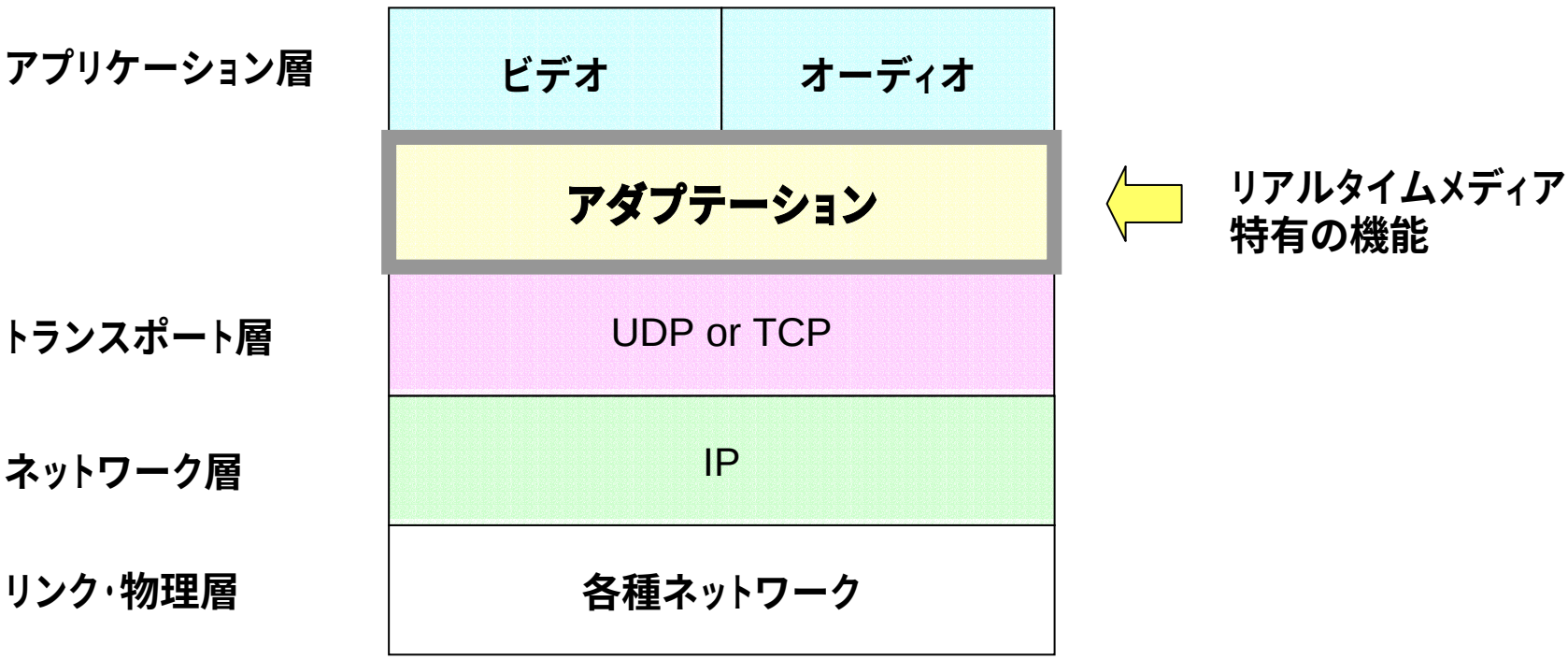
   廃棄されたパケットの影響をどのように抑えるか。

3. **ふくそう制御**

   レートを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



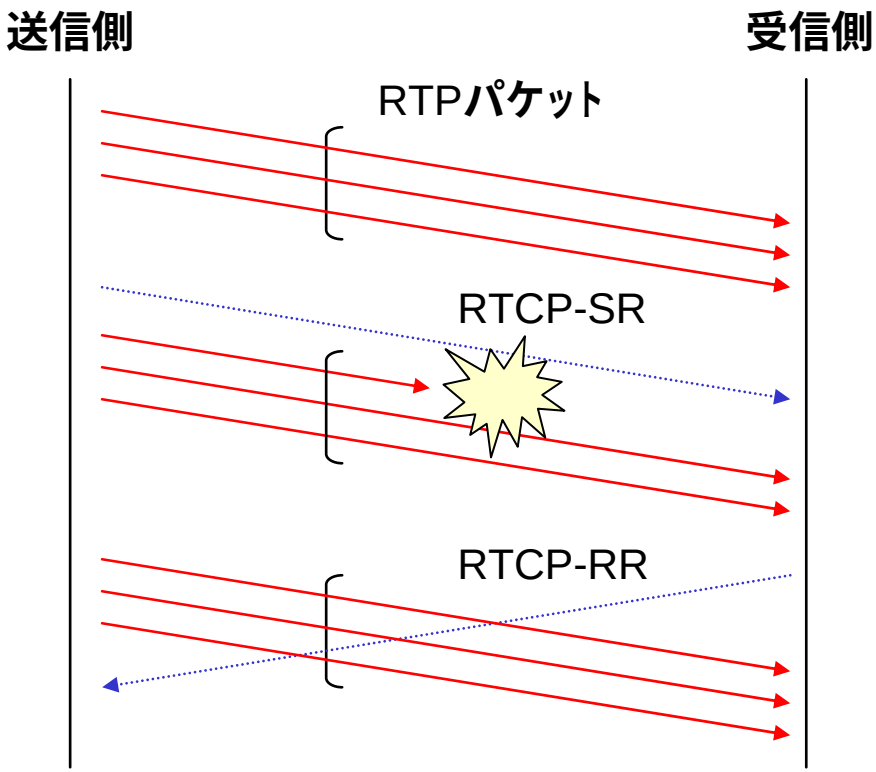
RTP/RTCP

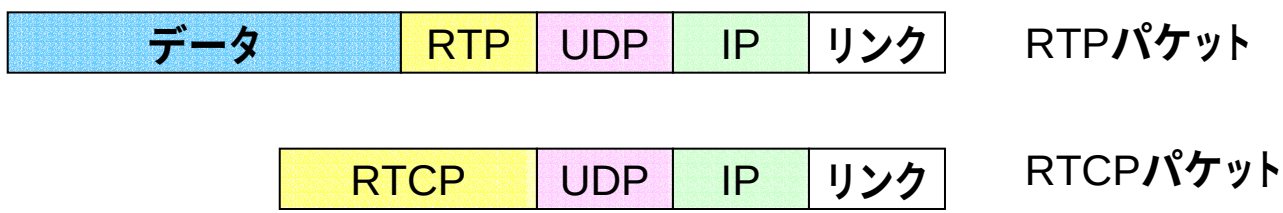
RTP: Real-time Transport Protocol

RTCP: RTP Control Protocol

RFC1889/1890 ... **現在** RFC **の更新を検討中**

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

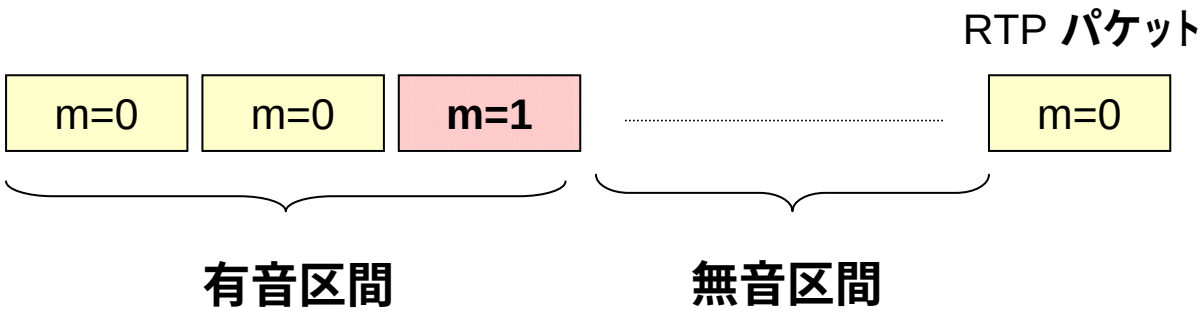
最近の符号化アルゴリズム (H.263、MPEG-4等) は SDP を用いた動的割り当て

M ビット

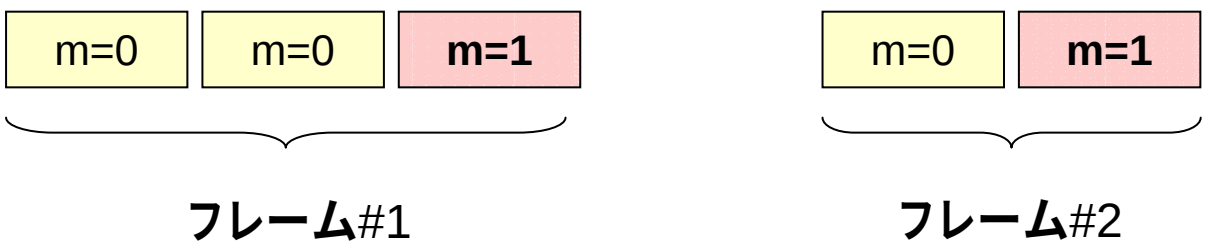
- アルゴリズムに応じたパケット区切りの解釈

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)

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 (total_packets)
送出バイト数 (total_bytes)

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)	

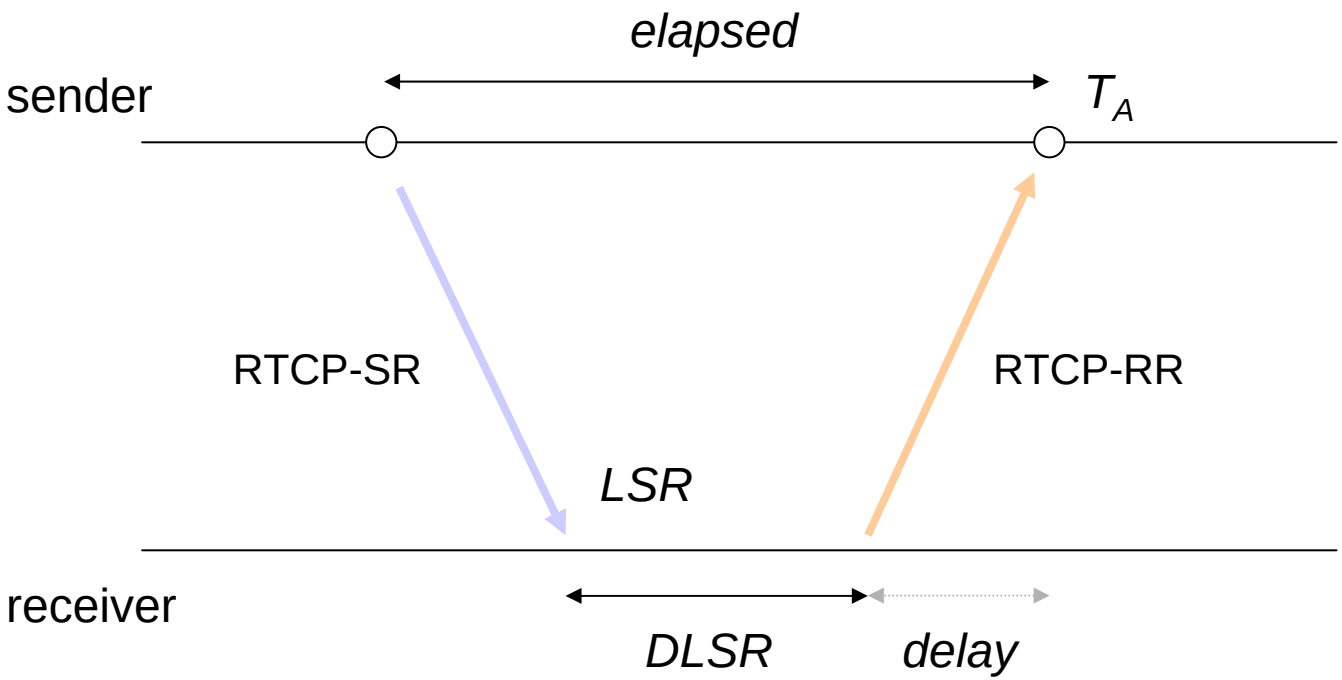
RTCP-RR (Receiver Report)

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

report block * n

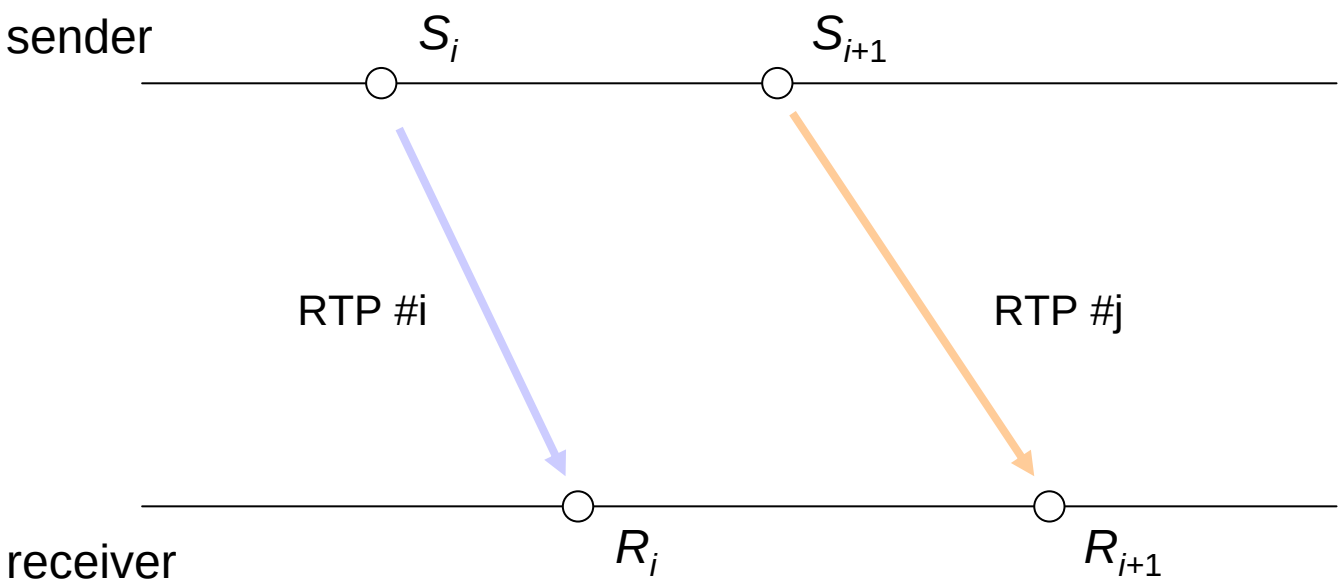
SSRC 識別子 #n	
瞬時廃棄率	累積廃棄パケット数
シーケンスナンバの最大値	
ジッタ遅延	
SSRC #n の最新のSR受信時の NTP タイムスタンプ (LSR)	
LSR から現在までの遅延 (DLSR)	

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： $Jitter(i) = (R_{i+1} - R_i) - (S_{i+1} - S_i)$

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

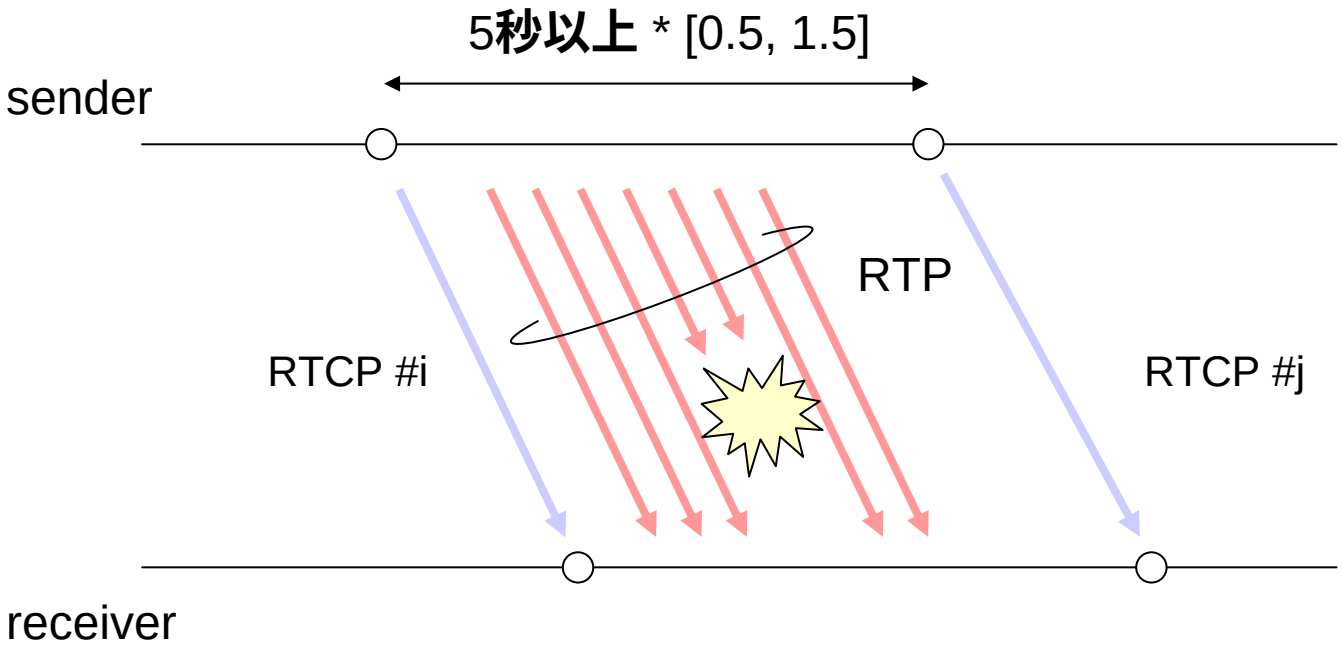
名称	目的
SR (Sender Report)	送信者からの NTP タイムスタンプと統計情報 (送信パケット数、送信バイト数) の通知 ＋ 受信者でもある場合は Receiver Report
RR (Receiver Report)	受信者からの統計情報 (廃棄率、ジッタ、等) の通知
SDES (Source Description)	セッション参加者の情報 (CNAME、メールアドレス、等)
BYE	セッションからの離脱
APP (Application Specific)	アプリケーション拡張

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

メディア同期

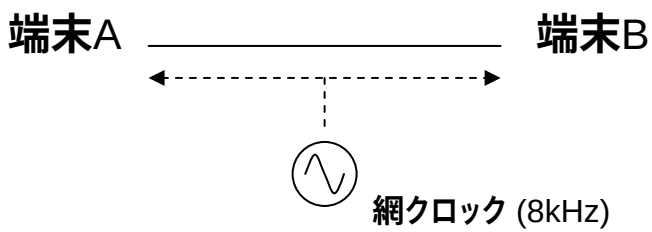
バッファリング + タイムスタンプ

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

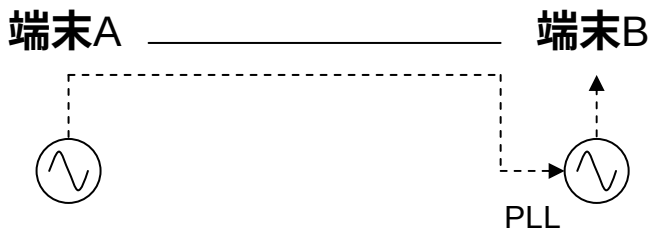
(例) 電話網、ISDN、モバイル



(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

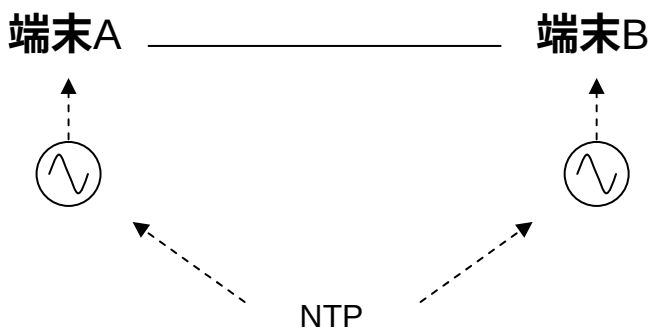
(例) ATM、ディジタル放送



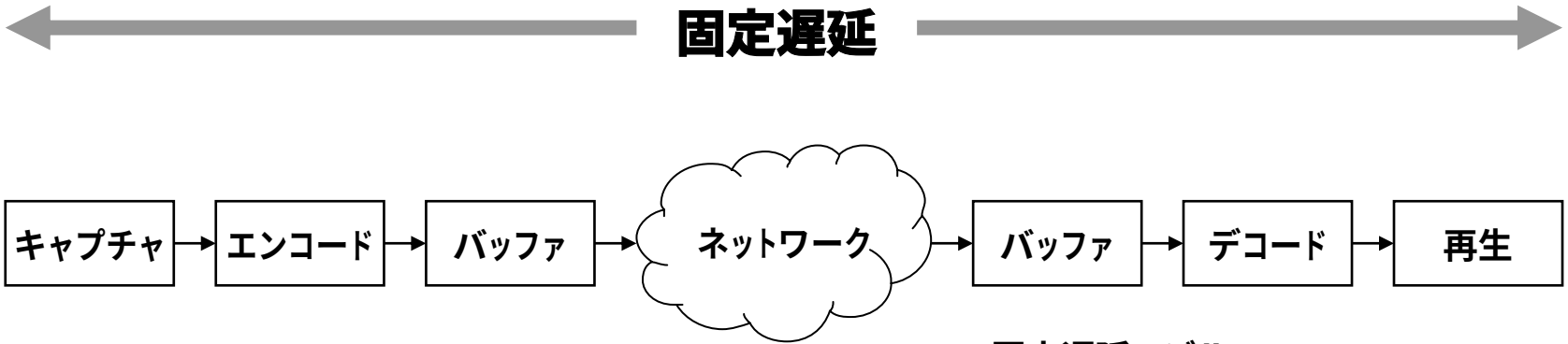
(3) 自走クロック (インターネット)

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)

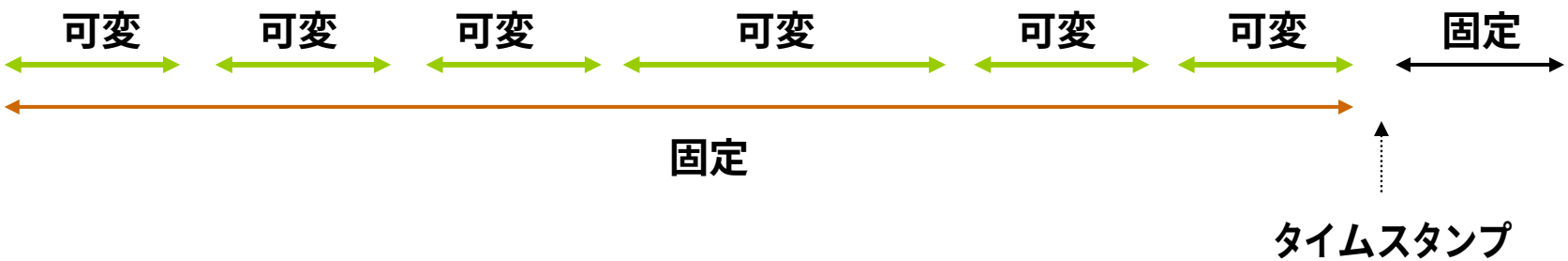


固定遅延モデル (constant delay model)

(1) 共通クロック方式 (回線交換)

すべてが網クロックに従って動作

(2) タイムスタンプ方式 (パケット交換)



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、ディジタル放送)

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

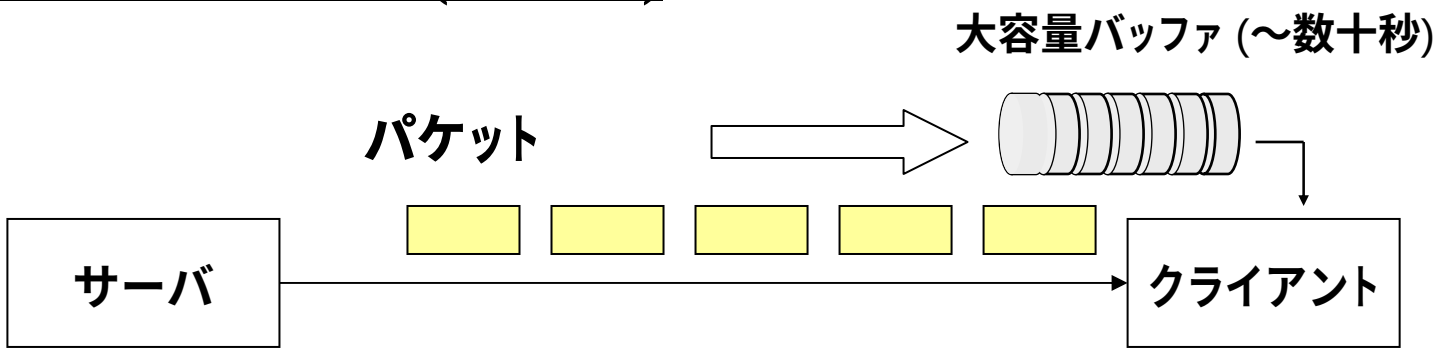
→「バッファリング」で遅延のばらつきを吸収

→「タイムスタンプ」(キャプチャ時刻) で再生時刻を決定

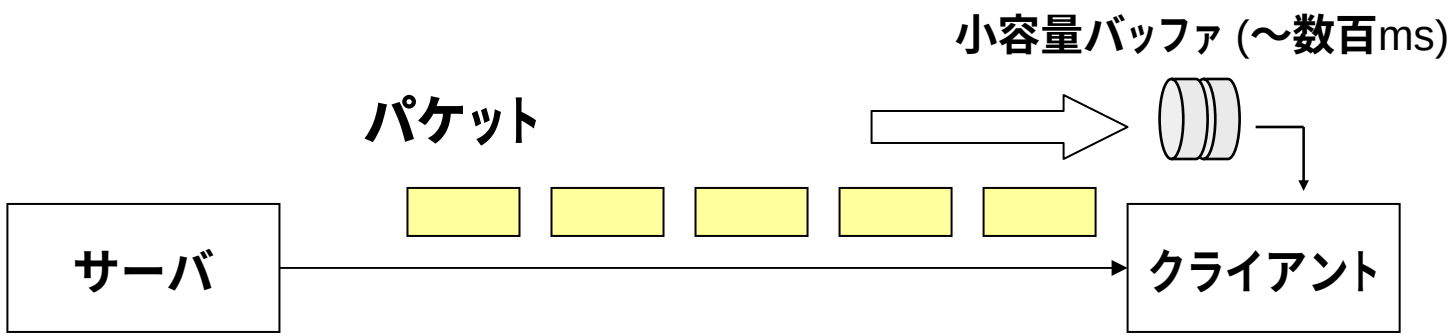
→ キャプチャから再生まで固定遅延

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

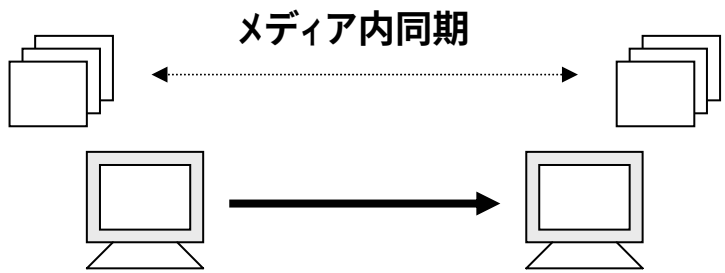
→ ＵＤＰを使用 (一般的にパケット廃棄対策が必要)

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

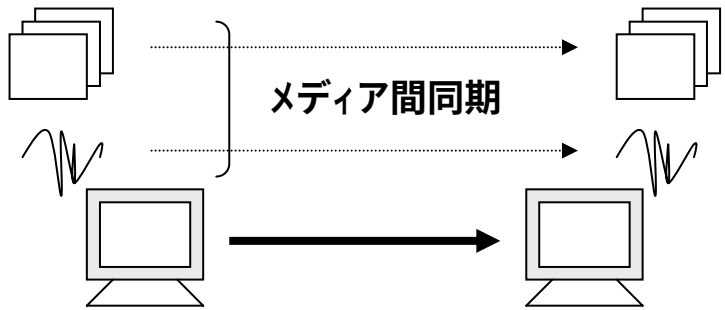
(手段) RTPタイムスタンプ (固定遅延モデル)



(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)

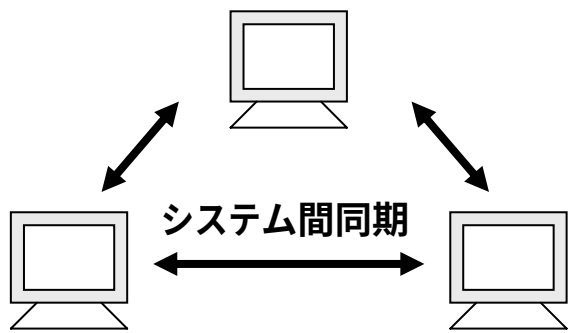
(手段) RTCPによるタイムスタンプの関連付け



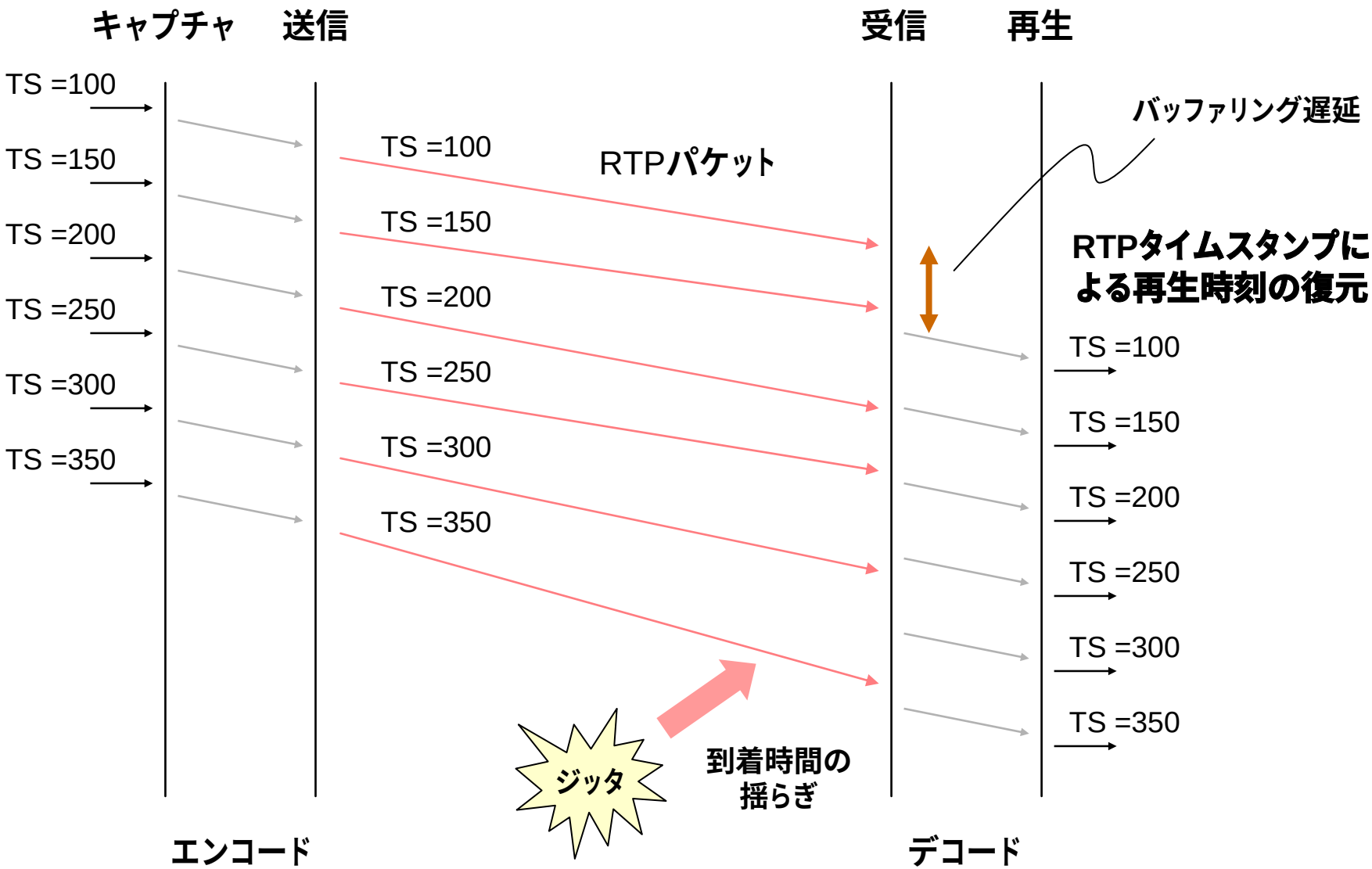
(3) システム間同期

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

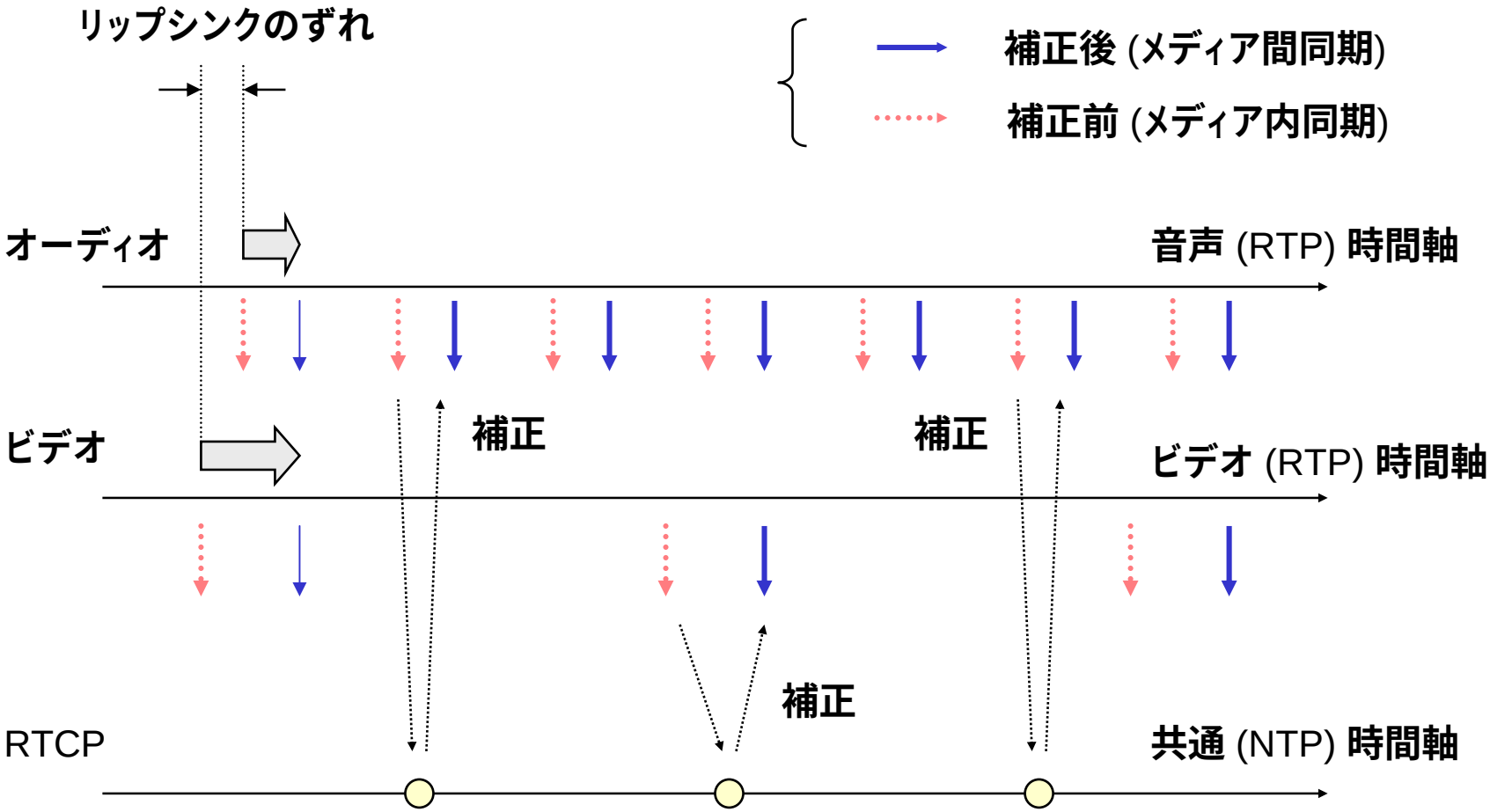


メディア内同期



固定遅延モデル、RTP

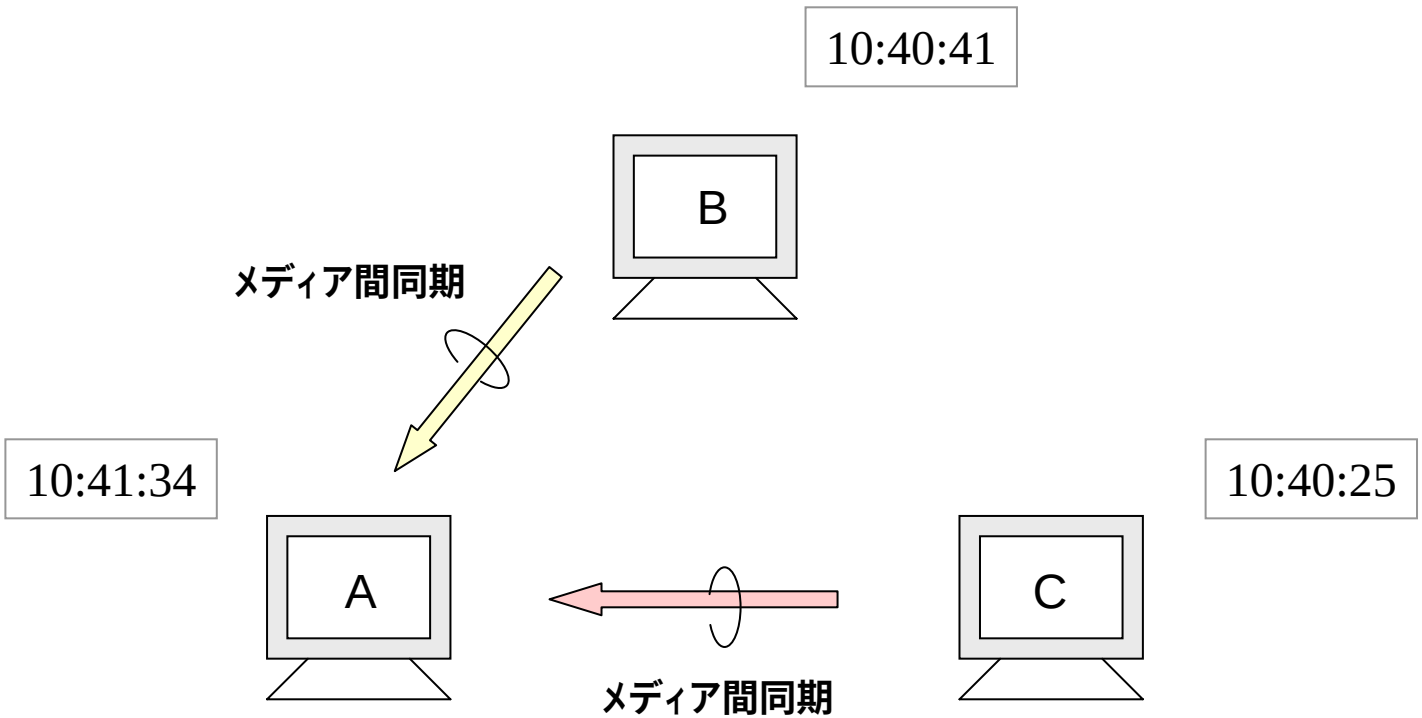
メディア間同期



個々のメディア用タイムスタンプ (RTP)
＋ 共通時間軸のタイムスタンプ (NTP)

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列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 (Network Time Protocol)

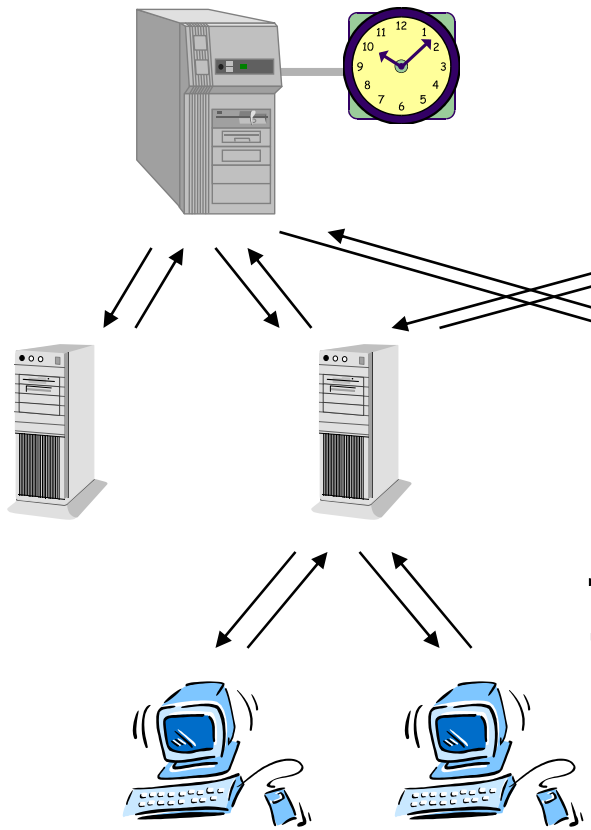
ネットワーク上の端末の時刻を合わせるためのプロトコル

(例) UNIX：xntpd、Windows: **桜時計、など...**

stratum 1
サーバ

stratum 2
サーバ

stratum 3
クライアント



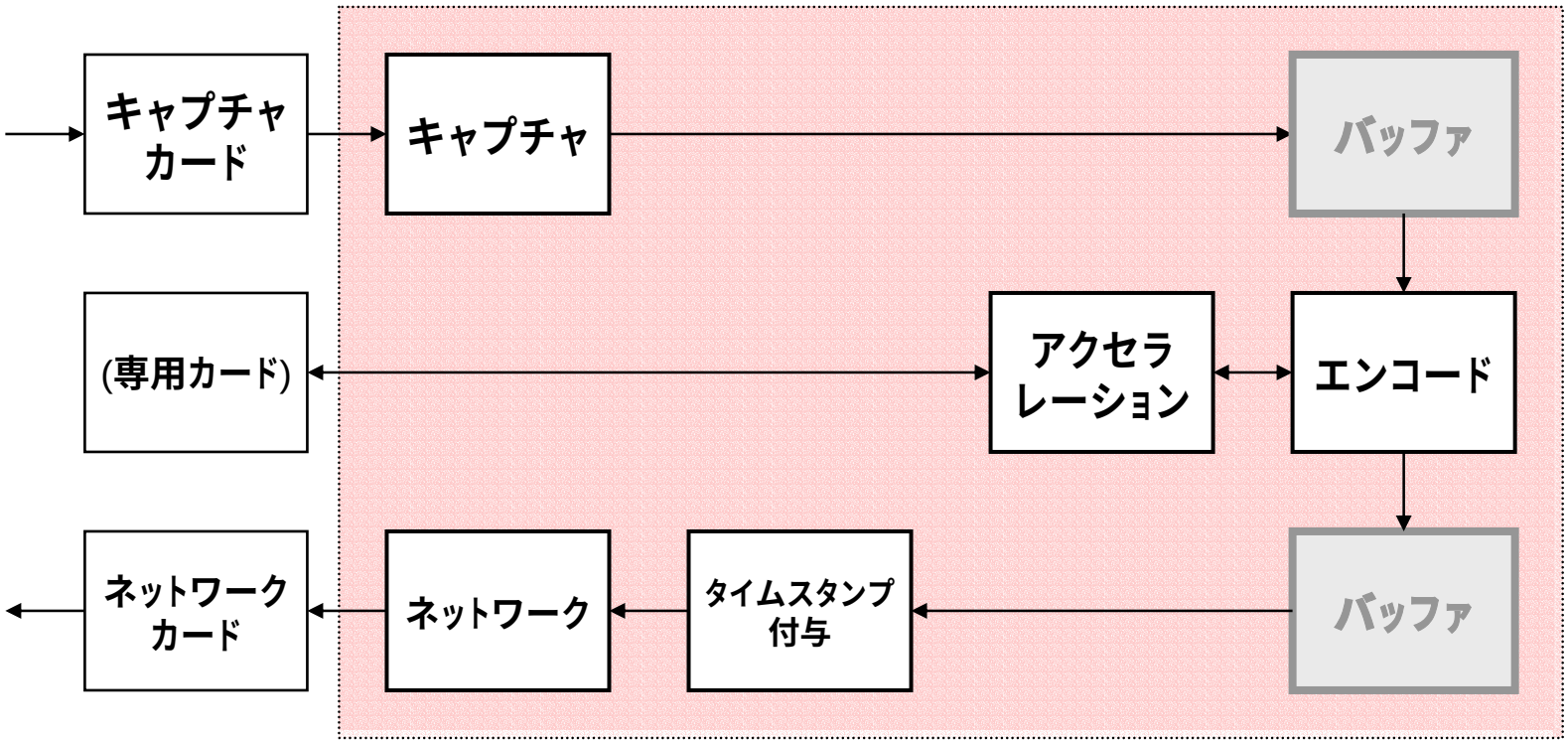
原子時計、GPS、...

サーバ･クライアント型パケット情報交換、ラウンドトリップ時間計測、時刻の推定

数ミリ～数十ミリ秒の誤差、と言われているが、リアルタイム系の精度の問題も指摘されている。

実践編

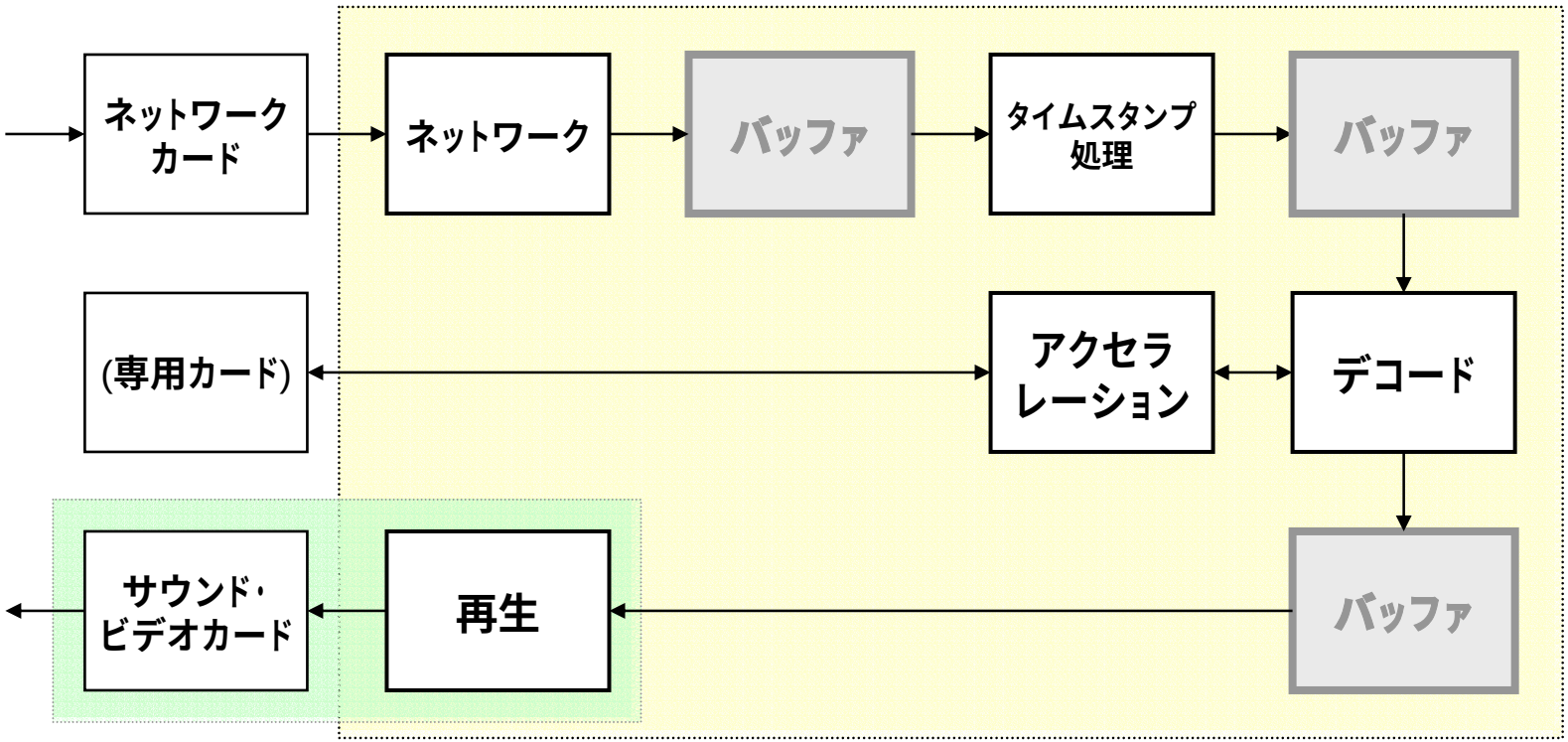
メディア同期の実際 (1. 送信側)



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)

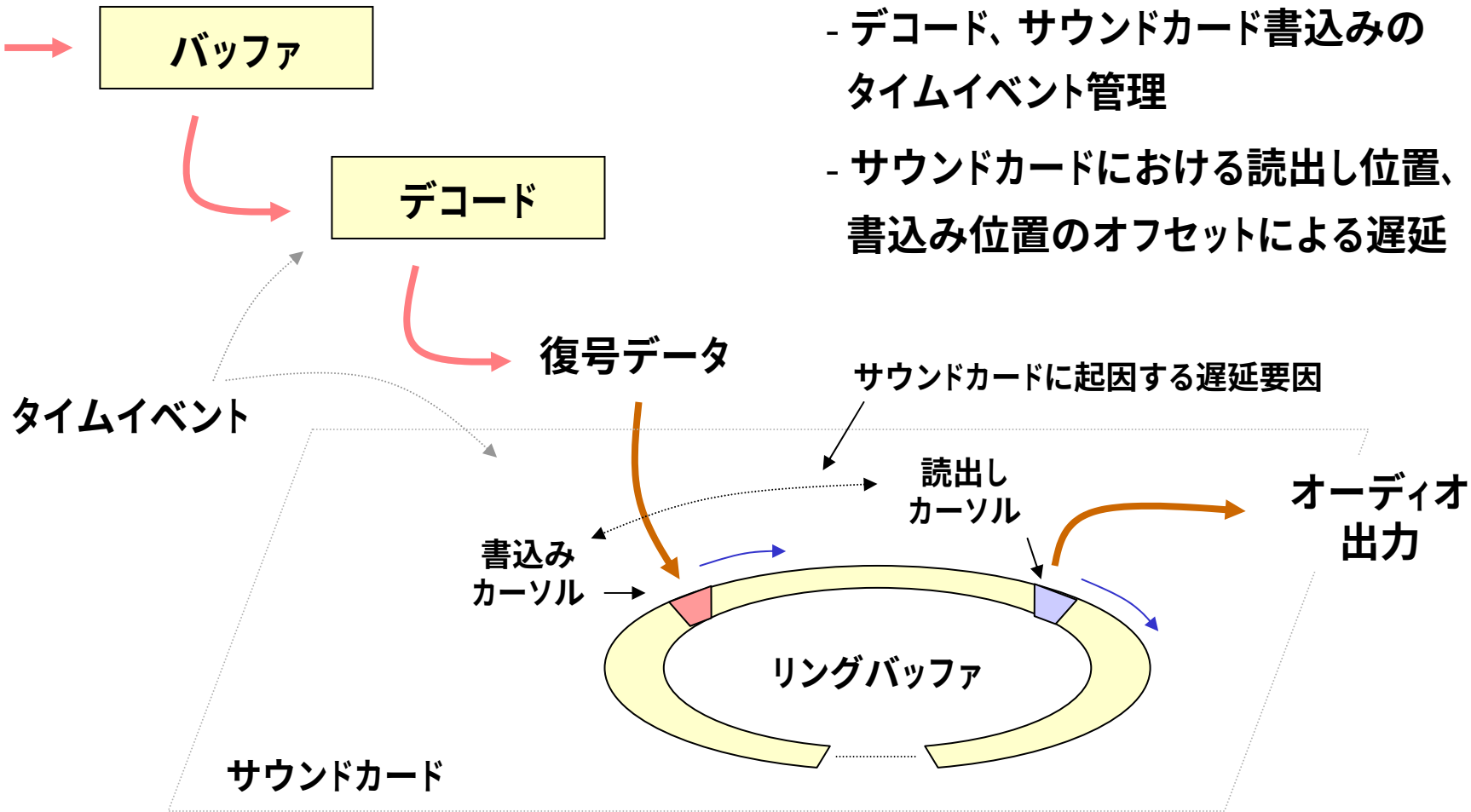


バッファの活用

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)

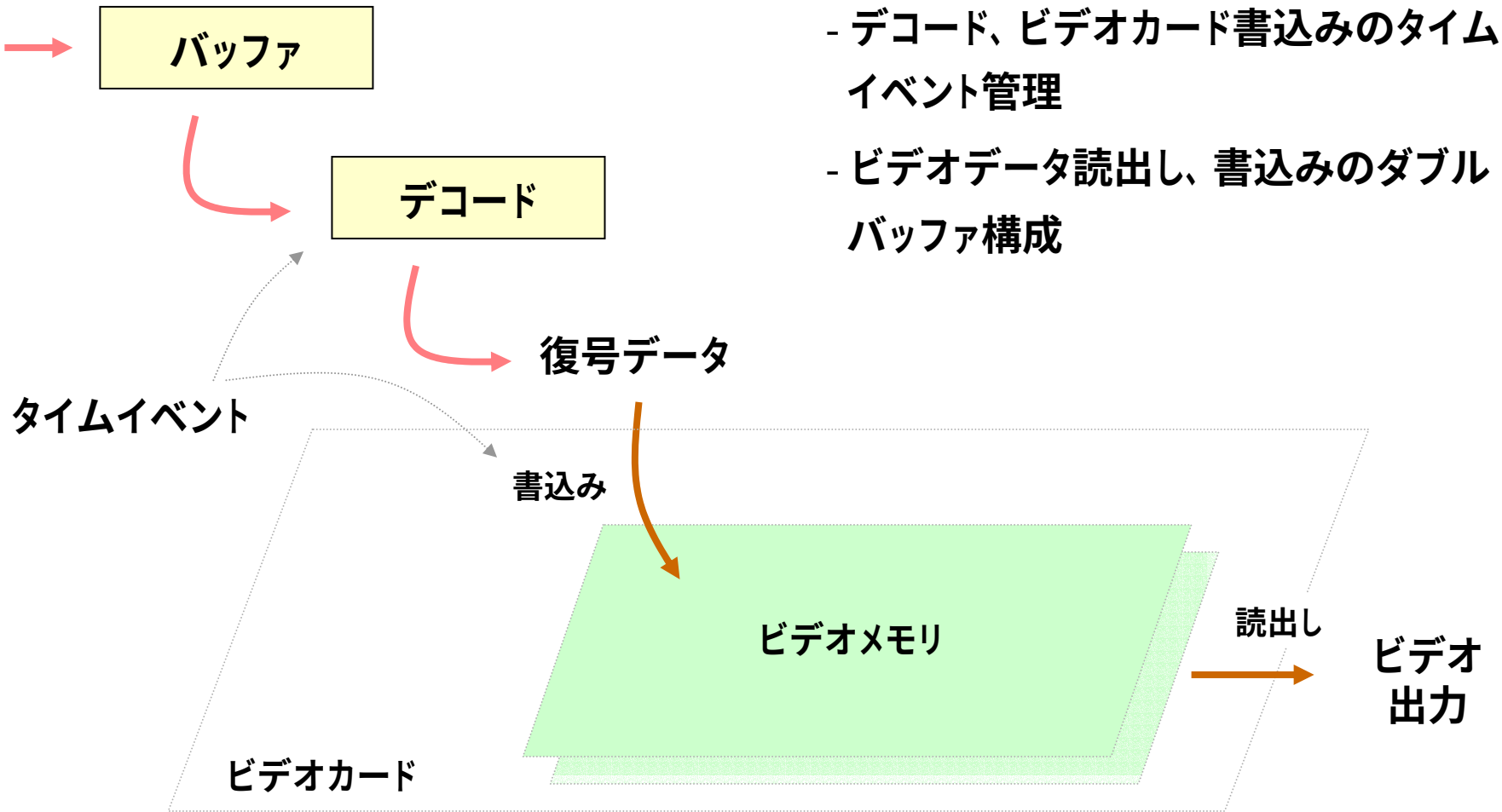
- オーディオのストリーミング再生



- デコード、サウンドカード書込みの
  タイムイベント管理
- サウンドカードにおける読出し位置、
  書込み位置のオフセットによる遅延

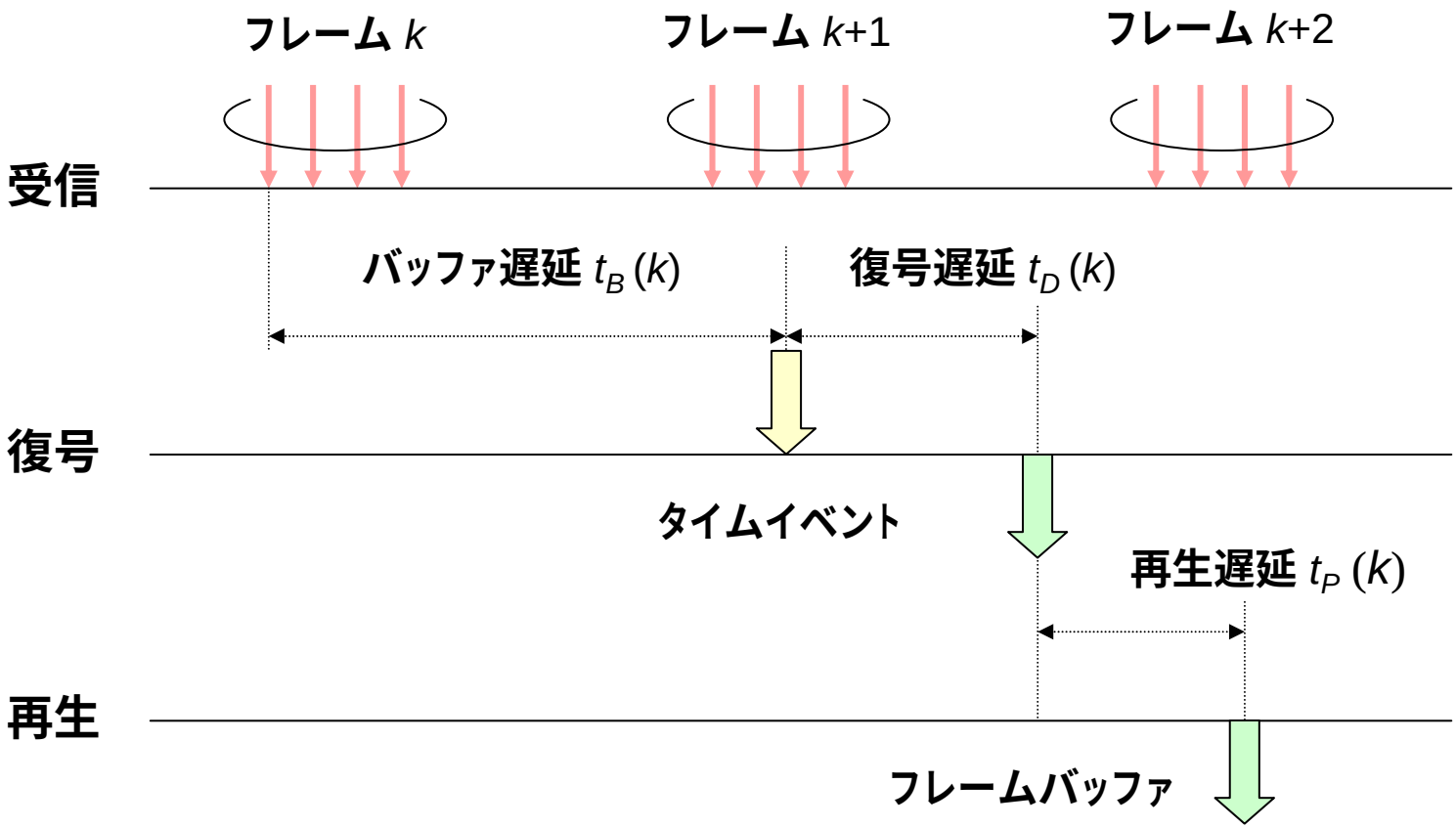
メディア同期の実際 (4)

・ビデオのストリーミング再生



- デコード、ビデオカード書込みのタイムイベント管理
- ビデオデータ読出し、書込みのダブルバッファ構成

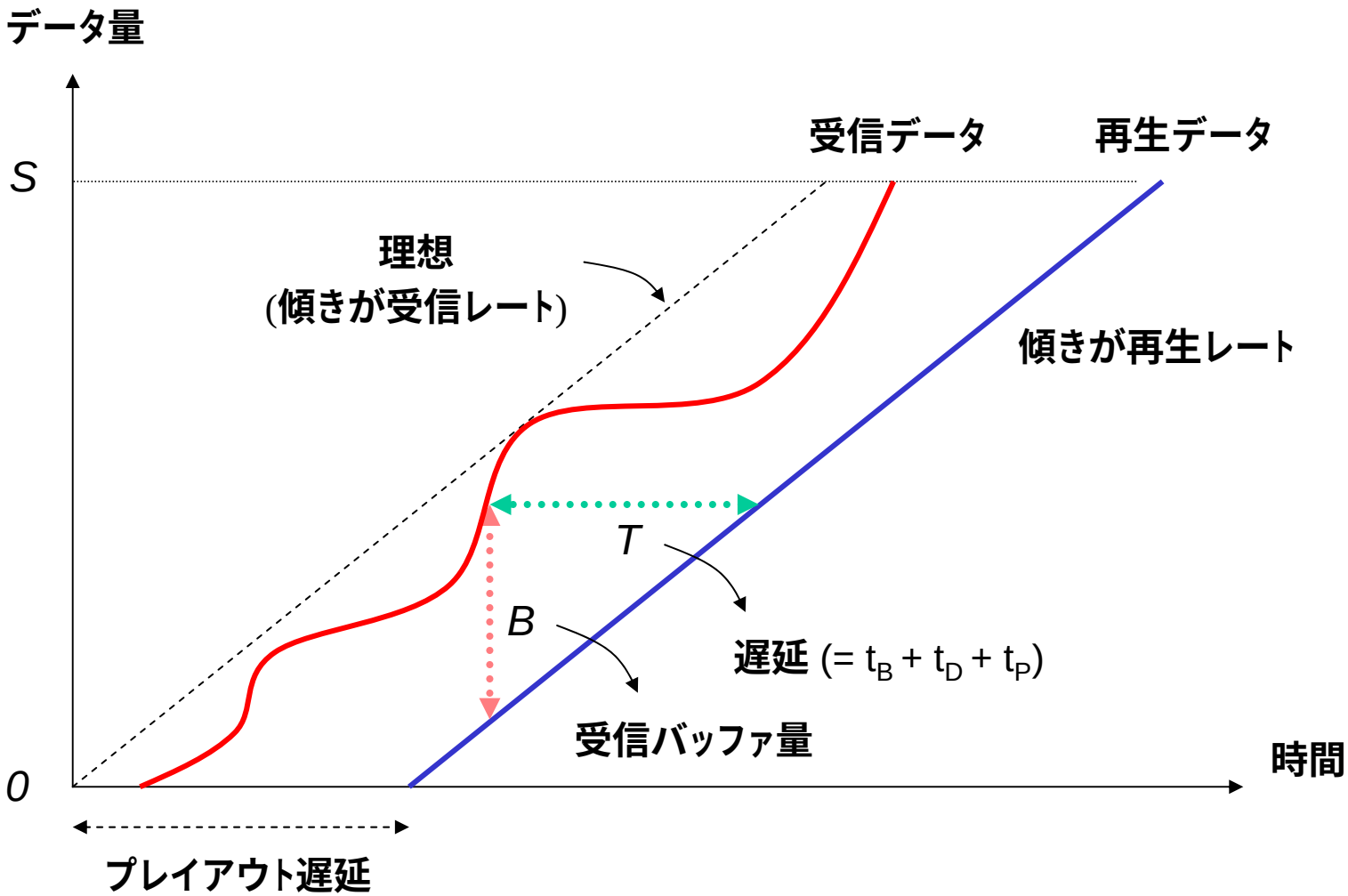
メディア同期の実際 (5)



$t_B(k)$ = *variable*, $t_D(k) + t_P(k)$ = *const*

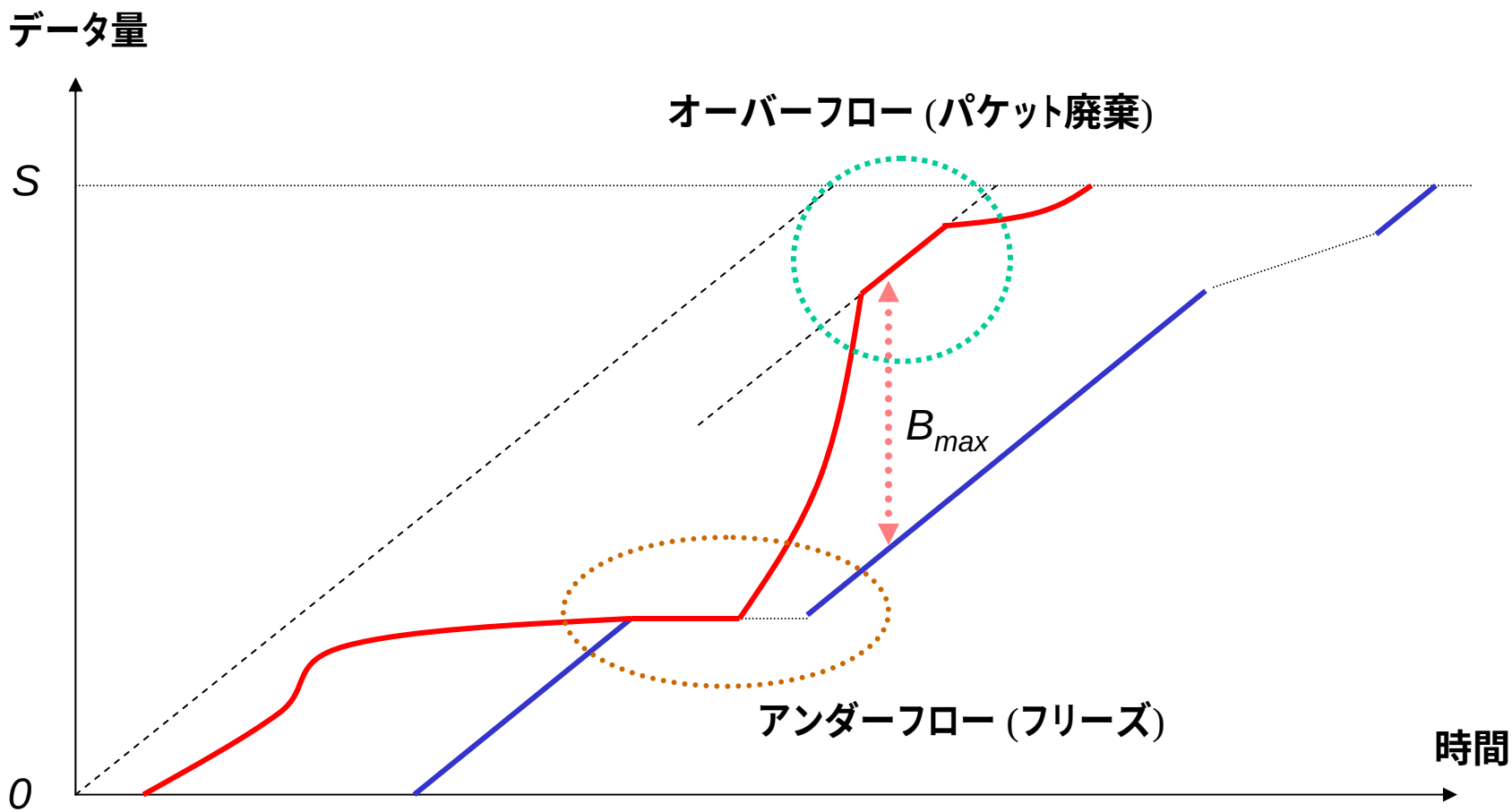
メディア同期のモデル (1)

- 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

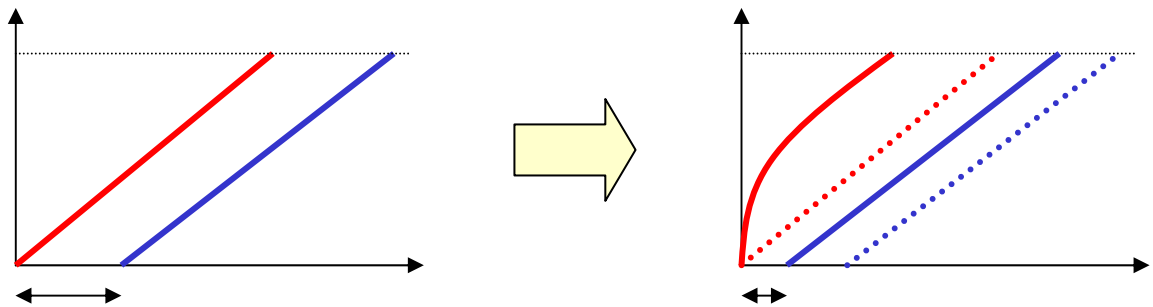


ただし、(メモリを十分に使える) PC では、オーバーフローはあまり発生しない

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)



(欠点) ライブには適用できない

(2) 送信レートを下げる (TCPフレンドリ：次回参照)