卒業論文　　2003年度 (平成15年度)

分散協調環境における効率的な高品質映像制御に関する研究

慶應義塾大学 総合政策学部

氏名：堀場 勝広

指導教員

慶應義塾大学 環境情報学部

村井 純

徳田 英幸

楠本 博之

中村 修

南 政樹

平成16 年1 月5 日

卒業論文要旨 2003 年度 (平成 15 年度)

分散協調環境における効率的な高品質映像制御に関する研究

本研究では，DV フォーマットを用いた資源対効果の高い True VoD システムの設計と実装を行った．既存の VoD システムと比較して，時間粒度の細かい高品質映像の再生制御が可能なシステムを構築した．

本研究は，高品質映像の配信，時間粒度の細かい再生制御をターゲットとし，既存研究と比較してより効率良く計算機資源が活用可能な VoD システムの構築を目的とした．

インターネット上で映像の再生制御可能な VoD システムの多くは，フレーム間圧縮フォーマットを用いるため，時間粒度の細かい再生制御を実現できない．また，大容量映像データを転送する VoD システムである DVBS では，IP マルチキャストを用いる．IP マルチキャストは，ユニキャストにおける個別映像配信と比較して，サーバの二次記憶装置，およびネットワークインターフェースの負荷軽減対策となる．しかし，IP マルチキャストを用いる場合，個々のクライアントが要求する映像制御要求に応えられない．

計算機資源を大きな負荷をかける高品質映像フォーマットを用いて個々のクライアントが要求する映像制御に応えるには，ボトルネックとなる映像配信を行うサーバの二次記憶装置，およびネットワークインターフェースをはじめとした計算機資源を効率的に利用する必要がある．

本研究では，効率良く計算機資源を活用する環境として，高精度な実時間処理と，厳密な時間管理に基づくタスク管理を可能にするリアルタイム OS に着目した．リアルタイム OS をサーバに用いることで，汎用マルチタスク OS と比較して，映像配信を行うサーバの二次記憶装置，およびネットワークインターフェースを効率的に使用できることを示した．その結果，サーバが生成可能な映像ストリームの向上，および安定した映像データの入力と，ネットワークへの送信を可能にした．

本研究では，映像の再生制御はフレームの制御で実現できることを示し，フレーム内圧縮フォーマットとフレーム間圧縮フォーマットの比較から，フレームの制御にはフレーム内圧縮フォーマットが適していることを示した．フレーム内圧縮を用いる DV フォーマットの特徴と，タイムコードを利用して，高精度なフレーム制御を可能にし，True VoD システムを実現した．

本研究により，有限な計算機資源上において，より計算機資源の利用効率のよい高品質映像配信サービスが実現できる．

キーワード

1．高品質映像，2．VoD，3．映像制御，4．パフォーマンス

慶應義塾大学 総合政策学部

堀場 勝広

Abstract of Bachelor's Thesis

Academic Year 2003

Research based on Effective High Quality Video Playback Control in Distributed and Coravorated Environment

This research proposes the design and implementation of resource aware true VoD system establishing high quality and precise playback control of video stream.

This research targeted high quality video transport and precise playback control of video streams as a goal of the demand, resulting effective resource managed vod system compared to traditional vod.

Most of the vod systems used in the Internet infrastructure uses the interframe compression method for a video stream influencing with precise control of video playback.

In addition, vod system generating massive video streams such as DVBS uses IP multicast as a transport. compared to unicast based transport, multicast transport decreases the load of secondary storage and network interface. But multicast transport lacks the operability in control of playback since images are shared with multiple clients.

controlling indivisual client playback request using high quality video format requiring high computational power, effective resource management of secondary storage unit, such as hard disk, or network interfaces are necessary.

This research focuses on realtime operating system enabling precise realtime processing and time managed task management based on realtime processing.

using realtime OS as a server, compared to generous OS, secondary storage and network inte rfaces used in server can be effectively managed.

In addition stable video playback, network output, with improved scale factor, are provide d.

This research proved the frame based playback control by the client. Compared to the inter frame compression to frame comptrssion algorithms, frame based playback control is more effective using frame compression.

Using DV format which uses frame compression algorithm, and time code built inside the DV format, are used to establish True VoD system: precise video frame playback control.

This research establishes the high quality video transport system environment in resource aware computational environment by managing effective use of resources.

Keywords :
1. High Quality Video, 2. VoD, 3. Video Control,4. Performance

Keio University , Policy and Management

Katsuhiro HORIBA

目 次

図 目 次

表 目 次

第1章　序論:分散環境における映像配信

1.1　背景

計算機や，ディスプレイなどの計算機を取り巻く周辺機器の高性能化と，デジタル映像技術の発達により，計算機上で DV(Digital Video)[1] や MPEG2(Moving Picture Image Coding Expert Group Phase 2)[2] をはじめとした高品質映像の視聴や，編集作業用のアプリケーションが急速に普及した．また，インターネットの普及とともにデータリンクの広帯域化が急速に進み，エンドユーザまで 100Mbps の広帯域データリンクが普及し始めている．

これらの背景から，DVTS[3] や MPEG2-TS over IP[4] など，これまでのデータリンクの帯域や計算機環境では困難とされてきた高品質映像転送アプリケーションが利用可能になった．これらのアプリケーションの応用例として，デジタルシネマコンソーシアム [5] などではすべての映画作成環境をデジタル化し，作成した映画をネットワークを用いて映画館へ配信するサービスを模索している．また，近年急速に普及している技術としてハードウェアに汎用的な組み込み型 OS を備え, ネットワーク接続性を持つ情報家電がある．映像配信の分野では, ストリーミングサーバの機能を併せ持つ民生用ハードディスクレコーダが登場した．また，DV フォーマットを用いたリアルタイム映像転送機器の試作品が DVTS コンソーシアム [6] で開発されている．

映像品質や，計算機・映像出力デバイスは今後も成長を続け，HD(High Definition)TV[7] と同等品質の映像データが，エンドユーザまでネットワークを介して配信される環境を本研究では想定する．

1.2　高品質映像配信サービスへの要求と必要性

高品質映像を用いた映像転送を行うための環境や技術は整いつつある．そのため，高品質映像を用いたリアルタイムストリーミングによる放送型配信や，コミュニケーションツールだけではなく，蓄積された映像データを用いた VoD(Video on Demand) や，ネットワーク上のファイルサーバに点在する映像データを用いたノンリニア編集などの新しいサービスが要求されている．

しかし，現在の主流は Microsoft[8] や RealNetworks[9] などが提供する MPEG4[10] などの画質，解像度，映像操作性の低い映像フォーマットを用いた映像配信サービスである．また，情報家電においても同様のアプローチがとられてきた．しかし，図 1.1のデジタル映像フォーマットの映像品質とデータ量の関係が示す通り，これらのサービスは，計算機やディスプレイなどの映像出力デバイスの性能が低く，データリンクが低帯域だった環境に適応した技術である．そのため，現在の計算機環境や，ネットワーク環境での新しい映像配信サービスへの要求に応えることはできない．

本研究では，高品質映像配信サービスが満たすべき環境を，TV 放送と同等品質以上の映像

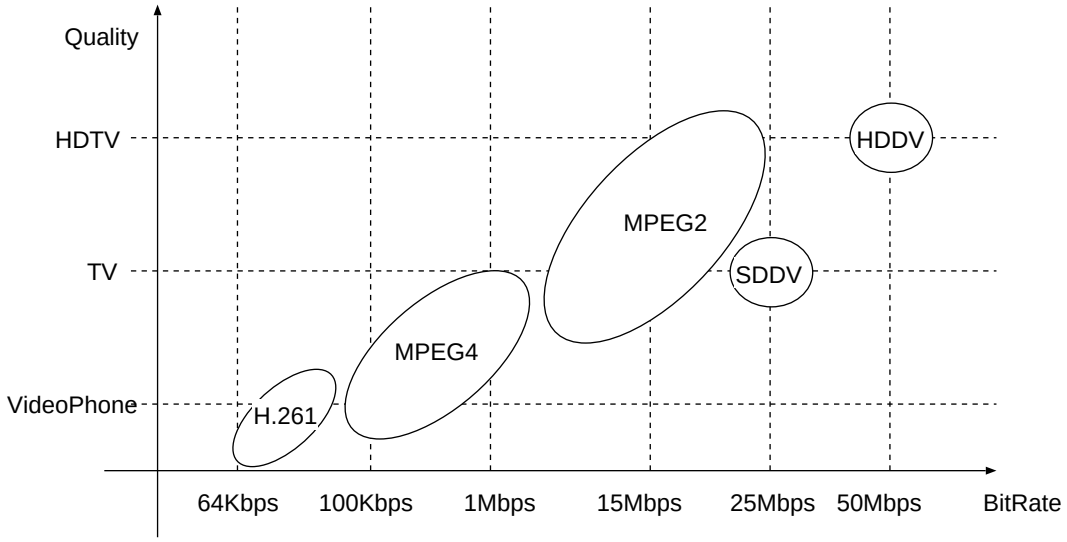

図 1.1: 映像品質とデータ量の関係

を IP ネットワークを介して複数のクライアントへ配信すると同時に，配信されている映像を任意に制御可能な環境であると定義する．

図 1.2に本研究が想定する高品質映像配信サービスの利用環境を示す．サーバはストレージ上に高品質な映像データ A,B を持ち，クライアントホスト 1 は映像データ A,B を用いて映像編集を行い，クライアントホスト 2,3 はそれぞれ映像データ B,A を試聴している．各クライアントホストは映像編集や視聴の際に細かい時間粒度で再生制御が可能である．また，クライアントホストが多数存在することを想定する．

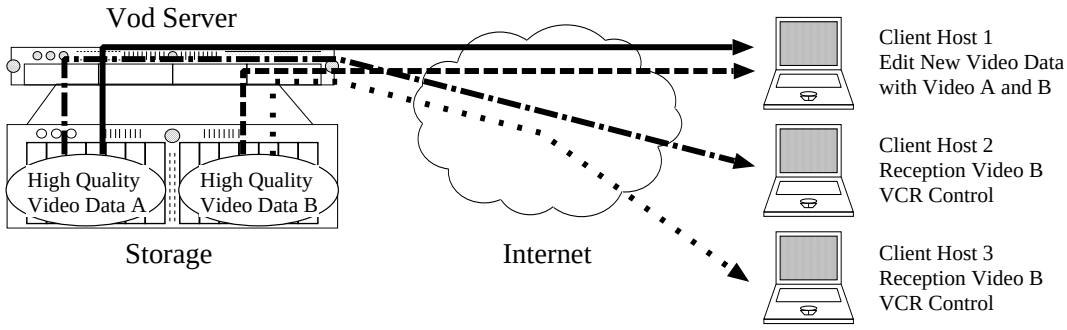

図 1.2: 本研究の想定する利用環境

これらの要求に応えるためには，時間粒度の細かい映像再生制御は不可欠な要素であり，映像配信を行うサーバに制御要求の正確性・応答性が要求される．また，多数のクライアントを想定するため，計算機資源対効果が高い VoD システムが必要である．

1.3 本研究の目的と意義

本研究では,1.2節の要求である, 高品質映像配信サービスにおける，時間粒度の細かい映像操作性と，資源の有効活用を満たす VoD システムを構築することを目的とする．また，情報家電などの計算機資源が比較的少ない場面においても，安定して動作可能な VoD システムの構築を目標とする．

具体的に本研究が目標とするシステムは以下の 3 つの条件を満たすものである．

- 高品質映像フォーマットの利用
  現行の TV 放送と同等以上の画質や解像度を実現する映像フォーマットの利用
- 時間粒度の細かい映像操作性
  IP ネットワークを介してクライアントが視聴している映像の時間粒度の細かい再生制御
- ハイパフォーマンス
  限られた計算機資源の中で，資源を最大限に活用することで資源対効果の効率を上げ，可能な限り多くのクライアントへの対応

これらの要件は，デジタル映像フォーマットや IP ネットワーク上におけるデータ転送プロトコルなど，既存技術の組み合わせ方を考慮し，計算機を構成するプロセッサ，ストレージ，ネットワークインターフェース，そしてそれらを接続するバスインターフェースなどが相互に協調し，有効活用される環境がなければ実現することができない．そのため，これらの基盤となる環境を構築し，今後インターネット上に展開される映像配信技術を支えることが本研究の意義である．

1.4 本論文の構成

本論文は 8 章によって構成される．第 2章において，本研究と関連の深い VoD の要素技術として，デジタル映像フォーマット，VoD の概要，デジタル映像を IP ネットワーク上で転送する技術を述べ，VoD の要素技術の組み合わせ方について議論する．第 3章において，関連研究として二次記憶装置上に蓄積された映像データ転送システムとして，映像制御の視点から WMT を，大容量映像データ転送の視点から DVBS の紹介をする．第 4章において，関連研究が用いる要素技術の組み合わせから，関連研究の問題点を導きだし，その解決手法について述べる．第 5章において，その手法を実現するための設計に関して述べ，第 6章で実装に関して述べる．第 7章において，本研究の実装を定性的・定量的な側面から評価する．最後に第 8章で本研究の結論と，今後の指針を述べる．

第2章　VoDシステムの要素技術

本章では，1章で述べた目的を達成するために，デジタル映像とその扱われかた，圧縮方式について述べ，デジタル映像を利用して映像配信サービスを行う VoD の概要，IP ネットワーク上での映像転送に関する技術について述べる．また，映像配信形態に適した要素技術の組み合わせについて述べる．

2.1　デジタル映像フォーマット

本節では映像のデジタル化，高品質デジタル映像フォーマットについて述べる．デジタル映像フォーマットは大きく分けて，フレーム内圧縮フォーマットとフレーム間圧縮フォーマットの二種類がある．本節では，デジタル AV 機器上と，計算機上でも広く普及している高品質デジタル映像フォーマットとして DV と MPEG2 を例として，フレーム内圧縮フォーマットとフレーム間圧縮フォーマットについて述べる．

2.1.1　映像のデジタル化

映像・音声のデジタルデータ化は対象情報の一定時間間隔でサンプリングした情報を複数の階調に分解して量子化が行われ，細かな時間間隔で細かな階調に分けるほど高品質な映像となる．動画の場合，フレームレートが単位時間を表し，解像度が 1 フレームあたりの細かさを，色数・色解像度が色の階調の細かさを表す．そのため高品質映像・音声のデジタルデータは大容量になる．

動画は画像を連続して表示することにより実現されている．この連続する画像をフレームと呼ぶ．映像の操作はフレーム単位で行わる．フレームは用途に応じて様々な手段によって特定される．以下に，代表的なフレーム特定手段を挙げる．

- タイムコード
  タイトルの開始時を起点に，時間:分:秒:フレームで，以後順に値を増やしていく
- ATN(Advanced Tracking Number)(絶対トラック番号)
  テープの始まりからトラック一本一本にマーキングをする，デジタル映像ではフレームの絶対番号として使われる場合がある

解像度を $R_x \times R_y$，色解像度を n，フレームレートを f とし，未圧縮映像のデータ量を D_T(単位:bps) とすると式 2.1によって表される．

$$D_t = (R_x R_Y) n f \tag{2.1}$$

たとえば現行の TV 放送程度の画質である SD(Standard Definition) フォーマットで NTSC 方式の映像を表現する場合，解像度が 720 × 480，色解像度が三原色各色 8bit で 24bit，フレー

ムレートが 29.97fps と定義すると，データ量は

$$720 \times 480 \times 24 \times 29.97 = 237067382.8125(bps)$$

となり約 237Mbps となる．

このように，未圧縮のデジタル映像データ映像はデータ量が多く, デジタル AV 機器や, 計算機上での扱い, ネットワークを介したデータ転送に不向きなため, 圧縮技術を用いることでデータ量を減らして使用される．

2.1.2 フレーム内圧縮:DV

フレーム内圧縮方式である DV フォーマットは，1 フレーム毎に DCT(Discrete Cosine Transform) と VLC(Variable Length Coding) を利用した定レートの圧縮を行っている．そのためフレーム内圧縮方式では，個々のフレーム情報のみで画像を復号できる．また，DV フォーマットは編集用途に用いることを考慮して作成されたフォーマットである．そのため，1 フレーム毎のタイムコードや ATN の記述，テロップを表示するなどを記述するフィールドを備えている．画質は SD フォーマット NTSC 方式の場合で $720 \times 480 \times 29.97fps$ で，データ量は約 25Mbps となる．

DV フォーマットは，フレーム単位で抽象化されたフォーマットで，全てのデータ (映像，音声，システムデータ) はビデオフレーム単位で扱われる．フレーム単位で映像と音声を同時に扱う事により，音声と映像の同期などの問題が解決できる．DV データは，3 段階の階層構造で扱われる．DV フォーマットにおける階層構造を図 2.1に示す．

ビデオフレームデータは複数の「DIF(Digital Interface Format) sequence」に分割される．DIF sequence は 80byte の DIF block 150 個に分割される．DIF block は，DV データにおける基本構成単位であり，全ての DV 仕様で利用されている．各 DIF block は，3 バイト長の ID ヘッダを含む．ID ヘッダは，DIF block の種類とデータの位置を表す．DV フォーマットでは，DIF block の種類として「HEADER」,「SUBCODE」,「VAUX」,「AUDIO」,「VIDEO」の 5 種類が定義されている．DIF block の種類は DIF block の先頭 3Byte に確保されている DIF block ID を参照することで判別することができる．

2.1.3 フレーム間圧縮:MPEG2

フレーム間圧縮方式である MPEG(Moving Picture Expert Group)2 は画質に一意な仕様はい．通信・放送・蓄積メディアなどを想定し，様々な適用範囲を持つため，解像度，フレームレート，色情報の組み合わせが複数存在し，画質はプロファイルとレベルの組合わせで表現される．表 2.1にプロファイル@レベルとデータ量の関係を示す.

動画像は一般的に，あるフレームと前後のフレームは近似したデータであることが多いため，前後のフレームとの差分データを用いることで必要なデータ量を大幅に削減することができる．フレーム間圧縮フォーマットである MPEG2 は映像を動き保証と両方向予測に基づいた圧縮を用いて，前後のフレームの差分から，現在のフレームの復号を可能にしている．

動き保証では，動きベクトル (動きの方向と大きさ) を検出し，1 フレーム前の信号から検出された動きベクトル分の位置シフトを行う．

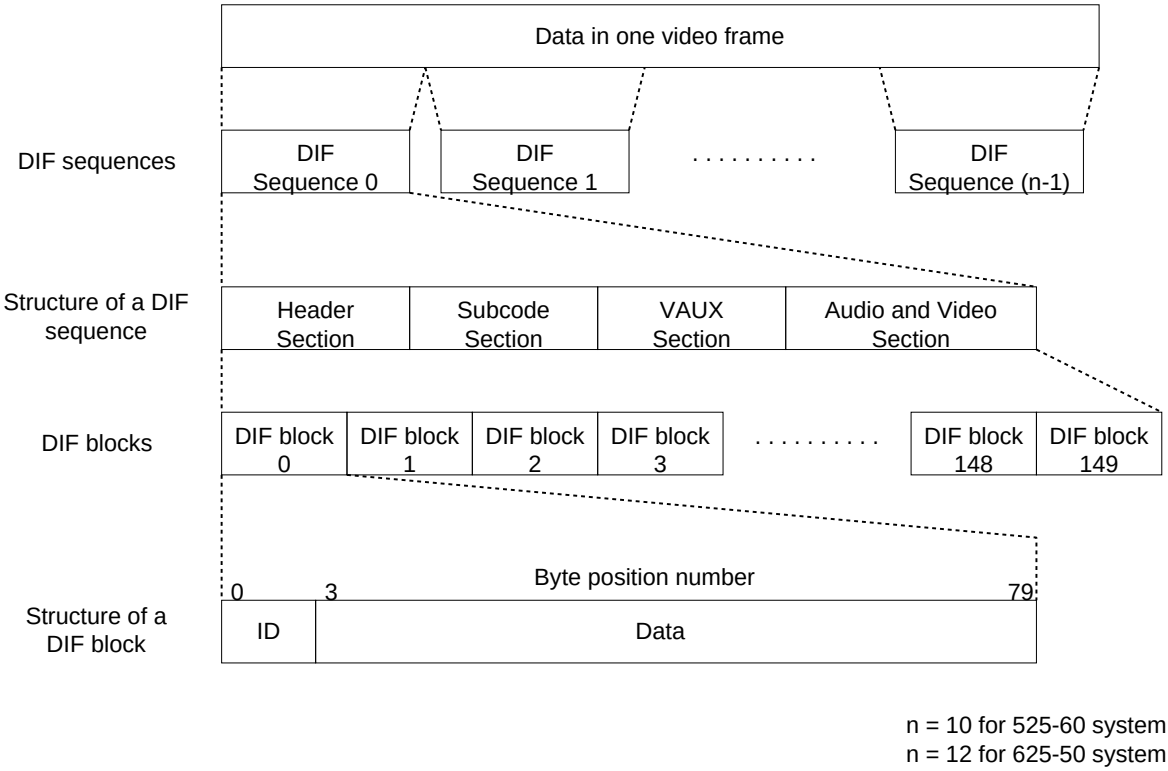

図 2.1: DV フォーマット

表 2.1: MPEG2 におけるプロファイル@レベルと最大ビットレートの関係

profile@level	MAX bitrate [Mbps]
SP@ML	15
MP@LL	4
MP@ML	15
MP@H-14	60
MP@HL	80
HP@ML	20
HP@H14	80
HP@HL	100

両方向予測では，予測符合化を用いて，現在よりも先のフレームの信号から予測を行う．MPEG[11] では I(Intra) フレーム，P(Prediction) フレーム，B(Bi-direction) フレームの 3 種類のフレームがある．P フレームは前方向予測，B フレームは両方向予測が用いられる．I フレームは予測による誤差を修正するためのフレームで，予測を利用せず，フレーム内で DCT で符号化が行われる．両方向予測を以下の図 2.2に示す．図中の四角はフレームを表す．四角内の英字はフレームの種類を表し，数字は伝送される順序を表す．また矢印は予測される方向を示す．

矢印の始点が予測に利用されるフレーム，矢印の終点が予測が行われたフレームである．たとえばIフレームはフレーム内で符号化が行われるため，Iフレームを矢印の終点とした矢印は存在しないが，Pフレーム・Bピクチャの符号化に必要なため，Iフレームを始点として，Bフレーム・Pフレームを終点とした矢印は存在する．

これらの複数のフレームの集まりを GOP(Group Of Picture) と呼ぶ．GOP 内には必ずIフレームが含まれ，ランダムアクセスは GOP 単位で行われる． そのため両方向予測を用いたフレーム間圧縮では，個々のフレーム情報単体で画像の復号を行うことができないフレームが存在する．

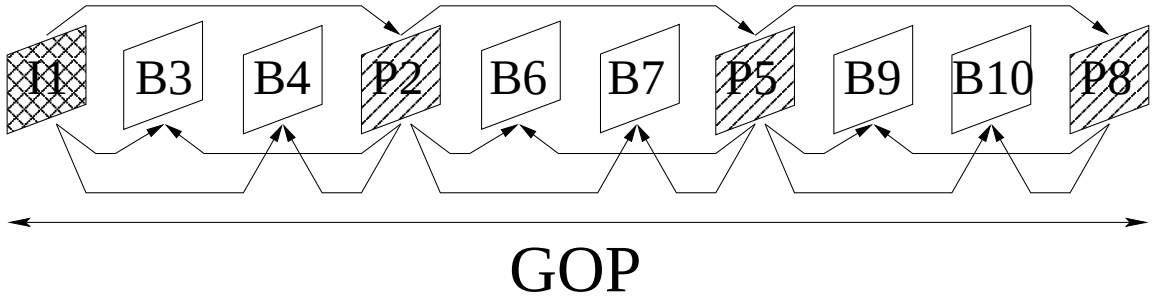

図 2.2: 両方向予測

2.1.4 フレーム内圧縮とフレーム間圧縮の比較

2.1.2節と 2.1.3節のまとめとして，表 2.2にフレーム内圧縮フォーマットの例である DV とフレーム間圧縮の例である MPEG2 のフォーマットの特徴の差異を示す．

表 2.2: フォーマットの比較

フォーマット	使用帯域	デコード時間	フレームのランダムアクセス	用途
DV	大きい/一定	短い	すべてのフレームで可能	編集・視聴
MPEG2	小さい/可変	長い	Iフレームでのみ可能	視聴

フレーム内圧縮フォーマットである DV は，使用帯域は大きいが，フレームレートが一定で，個々のフレーム情報で復号可能な特徴とタイムコードや ATN の連携によって，時間粒度の細かいフレーム制御を可能にしている．そのため，細かな再生制御に適している．また，デコード時間が短かいため編集作業にも適している．

逆にフレーム間圧縮フォーマットである MPEG2 は，複数の解像度・フレームレート・色情報を組み合わせを複数備え，再生する計算機の資源やデータリンクの帯域などに適用した映像配信が可能である．

2.2 VoD システムの概要

本節では VoD システムの映像配信形態による分類を述べ，その中で本研究が取り扱う True VoD について述べる．

2.2.1 VoD における映像配信形態別の分類

VoD システムは映像配信サービスにおいて，配信される映像に対してユーザが任意の操作を行うことができるシステムである．VoD システムにはユーザが実行可能な操作によって，表2.3に示される分類が行われている．[12]

表 2.3: VoD サービスカテゴリ

カテゴリ名	特徴
No-VoD(Broadcast)	ユーザは受動的に映像を閲覧する．ユーザは閲覧映像の選択を行えない
Q-VoD(Quasi VoD)	ユーザは閲覧するチャンネルの選択ができる
N-VoD(Near VoD)	ユーザは簡単な再生制御ができる．これは単一のコンテンツを時間差配信することで，擬似的に早送りや巻き戻しを実現している
T-VoD(True VoD)	ユーザは全ての再生制御ができる．ビデオデッキと同等の操作 (VCR 制御) 性を得られる

1.3節で述べたように，本研究の目的には映像の細やかな操作性があるため，True VoD はそれに近い映像配信サービスである．しかし，VCR 制御性と言う定義は曖昧で分かりにくいため，本研究では，True VoD を「フレームの操作によって制御可能な映像の操作をユーザに提供する映像配信サービス」として再定義する．

2.2.2 True VoD に求められる映像制御機能

True VoD サービス上で実行可能な操作の中で，ユーザが頻繁に利用すると予想される映像操作と，それらを実現する機能を以下に示す．

- 早送り再生/巻き戻し再生
  映像フレームを早送り・巻き戻しのスピードに応じて間引く，もしくはスピードに応じた速度で再生する
- 映像フレーム単位でのコマ送り/コマ戻し
  映像フレームを任意の数ずらす
- 任意のタイムスタンプへの映像フレーム単位での移動
  タイムコードや ATN を基に任意のタイムスタンプを探索し，任意の映像フレームへ移動する
- 一時停止/再開
  一時的に再生を停止し，要求に応じて再開する

2.3 インターネット上での映像転送技術

本節では，VoD システムにおいて，映像・音声の転送に用いられる技術に関して述べる．また, 転送方式と映像フォーマットの組み合わせによる適正環境について述べる．

2.3.1 ストリーミング

映像・音声データの全体を受信した後に再生を行うダウンロード再生に対して，データの受信と平行し，取得したデータから逐次再生を行う方法をストリーミング再生と言う．図 2.3はストリーミング再生を，図 2.4はダウンロード再生を示している．

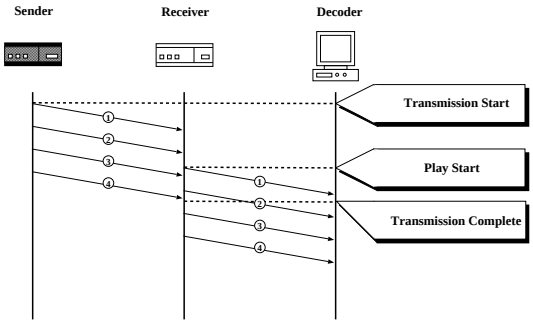

図 2.3: ストリーミング

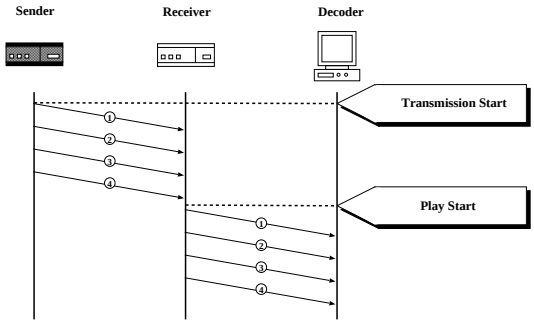

図 2.4: 非ストリーミング

IP ネットワークを用いてデータ転送を行う場合，パケットの到達時間は保証されない．そのため，ストリーミング再生を行う際，データ到着の時間の揺らぎ (ジッタ) が発生し，データの消費 (再生) に対してデータ転送が遅れる場合に映像の再生は一時的に止まる．この問題に対してストリーミング再生ではジッタの吸収のために，受信ホストで一時的に一定量のデータをバッファに蓄積し，蓄積されたデータを逐次再生するバッファリング技術が用いられる場合が多い．バッファリングはデータの消費開始時間を遅れさせるため，データ送信時間から再生時間までの遅延が大きくなる．そのため，バッファリングサイズが大きくなるほど，遅延時間も大きくなる．

ストリーミング再生は以下の様な状況下において効果的である．

1. データ転送・複製に長時間必要な場合
2. データ複製・格納にクライアント上で十分な領域を確保できない場合
3. 実時間性を強く要求し，映像の終了時刻が明確でない場合

第 1 項は，データ量の多い高品質映像や，長編映画などの再生時間が長いストリーミングメディアを再生する場合を指す．第 2 項は，再生を行う計算機上の一次記憶装置および二次記憶装置上に蓄積することが困難な場合を指す．第 3 項はライブ中継やビデオ会議などのリアルタイム配信の場合を指す．

2.3.2 ストリーミングに用いられるトランスポートプロトコル

IP ネットワークはパケット交換方式のネットワークであるため，到達性，帯域幅，パケット到達時間，データの完全性は保証されない．途切れること無く映像再生を行うためには，以下の項目を満たす必要がある．

- パケット到着順序，データの正確性
  復号時に正しい映像再生を行うために，映像データの順序が正しくなくてはならない

- 過不足ないクライアントバッファへのデータ転送
  映像再生を行うクライアントでは, 受信した映像データを格納するバッファが枯渇すると映像は止り, バッファが超過し, データが破棄されると映像が乱れる

ストリーミング再生では, これらの問題をエンドノード側の伝送処理によって解決している．IP ネットワーク上のデータ転送において，トランスポート層で用いられるプロトコルには TCP(Transmission Control Protocol)[13] と，UDP(User Datagram Packet)[14] がある．

TCP は, 欠損パケットの再送・シーケンス番号を用いたデータ順序の保証などを用いてデータの完全性を保証する．しかし，輻輳制御を行うため，データ送信を行うアプリケーションが任意の帯域を確保できない．

UDP は, 輻輳制御機構を備えておらず，欠損パケットの再送・データ順序の保証が無いため, パケットロスが起こった場合にデータの完全性を保証しない．しかし，アプリケーションが設定した帯域分のデータ送信を試みる．

映像ストリーミングでは，転送する映像フォーマットの特性や，実時間性の重要さに応じて TCP と UDP が使い分けられている．TCP と UDP の使い分けを表 2.4に示す．

表 2.4: トランスポートプロトコルの使い分け

プロトコル	遅延	データの完全性	適している映像フォーマット
TCP	大きい	保証する	フレーム間圧縮フォーマット
UDP	小さい	保証しない	フレーム内圧縮フォーマット

TCP は，データの完全性は保証されるため，一部のデータ欠損が致命的な映像の乱れにつながるフレーム間圧縮フォーマットには適している．しかし，遅延が大きく実時間性を重要視するストリーミング再生には適していない．また，バッファリングを行うことで停止することのない映像再生を実現するが，その分バッファリングを行うための一時記憶装置の領域を大量に確保しなくてはならないため，データ量の多い高品質映像フォーマット，特にフレーム内圧縮フォーマットには適していない．

UDP では，パケット到達順序の保証がないため，RTP(Real-time TransportProtocol)[15] を用いることにより，アプリケーションでのパケット到達順序への対応がなされている．しかし，UDP はデータの欠損に対して有効な対応手段がないため，パケットロスやジッタが発生した場合に，映像再生に対してデータの完全性を保証できない．そのため，個々のフレームで画像の複合が可能で，一部のデータ欠損に対して寛容なフレーム内圧縮が適している．また，UDP とフレーム内圧縮フォーマットの組合わせは，クライアントが実時間性を要求する場合に有効である．

2.4 まとめ:映像配信形態に適した要素技術の組み合わせ

本章では，デジタル映像フォーマット，VoD システムの映像配信形態，映像転送に用いられるトランスポートプロトコルの特徴と適正用途について述べた．本節では，True VoD，Near VoD,Quasi VoD, ビデオ会議の配信形態別に，適切な VoD の要素技術の組み合わせについて述べる．表 2.5に，映像配信形態，実時間性，適している映像フォーマット，適している転送プロトコルの関係を示す．

表 2.5: 配信形態に適した要素技術

配信形態	実時間性	適している映像フォーマット	適している転送プロトコル
T-VoD	高い	フレーム内圧縮フォーマット	UDP/RTP
N-VoD/Q-VoD	低い	フレーム間圧縮フォーマット	TCP
ビデオ会議	高い	フレーム内圧縮フォーマット	UDP/RTP

True VoD では，細かな映像フレームの操作を実現する必要がある．そのため，フレームのランダムアクセス性に優れたフレーム内圧縮フォーマットが適している．データ転送プロトコルについては，再生制御では再生制御要求から反映までの実時間性が要求され，映像編集などではデータの正確性の保証が要求される．そのため，TCP と UDP/RTP のそれぞれが持つ長所と短所を両立しなければならない．しかし，DV フォーマットなどのデータ量が多い映像フォーマットをバッファリングするほどの一時記憶装置領域を，現状のクライアント計算機に要求するのは現実的ではない．また，現状のインターネット網の帯域や，ジッタ，遅延などの要素から TCP では DV フォーマットのデータ量分のスループットを確保することは困難である．そのため，現状では大容量映像データになるフレーム内圧縮フォーマットを用いた True VoD システムに適している映像データ転送転送プロトコルは UDP/RTP である．

Near VoD,Quasi VoD は放送型の映像配信サービスで，実時間性への要求が低い．そのため，MPEG などをはじめとした，画像のエンコード・デコード時間は長いが，高圧縮率を実現している映像フォーマットが適している．また，データサイズが小さくでき，実時間性を求めないため，長時間かけてバッファリングし，TCP を用いてデータの完全性を保証することで安定した映像転送が可能である．

ビデオ会議は，実時間性が最も重要な映像配信サービスである．そのため，一部のデータ欠損に対しても，個々のフレーム情報で画像の復号が可能なフレーム内圧縮フォーマットが適している．また，映像データの完全性よりも実時間性を重要視するため，データ転送プロトコルは UDP/RTP が適している．

次章では，これらの要素技術を用いて True VoD を目標した Windows Media Technologies を映像制御の視点から，Quasi VoD を目標した DVBS を大容量映像データ転送の視点から紹介する．

第3章　関連するVoDシステム

本章では，二次記憶装置に蓄積された映像データを用いたVoDシステムを関連研究として挙げる．

簡単な映像再生制御機構を持ち，比較的低い映像転送レートで配信される例としてWMT(Windows Media Technologies) を挙げ，転送レートの高い映像データをIPマルチキャストを用いて放送型配信を行う例として，DVBS(DigitalVideoBroadcastingSystem)[16] を挙げる．

3.1　WMT

本節では，映像データにWMV(Windows Media Video) を用い，簡単な映像再生制御機能を備えたVoDシステムであるWMTの特徴を, 映像の制御と言う視点から述べる．

3.1.1　WMTの特徴

WMTはWindows Media Serverと呼ばれるサーバと，Windows Media Playerと呼ばれるクライアントで構成され，MPEG4をベースとしたフレーム間圧縮フォーマットであるWMVを，ユニキャストを用い低精度な映像再生制御可能なTrue VoDに近い映像配信形態，およびIPマルチキャストを用いた放送型映像配信であるQuasi VoDを可能にしている．

WMVの再生ビットレートは4Kbpsから20Mbpsまで対応していて，解像度や，フレームレートなどを高い自由度で選択可能である．

3.1.2　WMTの映像制御

WMVは既存のフレーム間圧縮技術の中で，Iフレームと同等の機能を持つキーフレームの配置を秒単位で選択可能な部分が特徴的な映像フォーマットである．そのため，映像再生制御の最小単位は1秒間である．

MSDN[17] が提供するマルチメディアSDKでは，WMVの制御に以下の関数が定義されている．

- STDAPI_(LONG) AVIStreamFindSample(PAVISTREAM pavi,LONG lPos,LONG lFlags);
  指定された位置からの相対的なサンプル (キーフレーム、空でないフレーム、フォーマット変更を含むフレーム) の位置を取得する
- STDAPI_(LONG) AVIStreamTimeToSample(PAVISTREAM pavi,LONG lTime);
  ストリームをミリ秒からサンプルに変換する

WMVの場合，サンプルは映像の場合ビデオフレームに相当する．これらの関数から，WMTではクライアントが映像再生開始時間と，要求するフレームの相対時間を再生制御要求として

サーバに送信し，サーバは相対時間を用いてキーフレームの位置を取得して，クライアントが要求する映像フレームを送信する．

3.2 DVBS

本節では，映像データに DV を用い，IP マルチキャストを用いた放送型映像配信を備えた VoD システムである，DVBS の特徴，アプローチについて，大容量映像データの送信という視点から述べる．

3.2.1 DVBS の特徴

DVBS は二次記憶装置上に蓄積された DV フォーマットのデータを読み込み，そのデータを UDP/RTP を用いてデータグラム化し，IP マルチキャストを用いて複数のクライアントに一斉転送を行うシステムである．また，現在配信サービス可能な映像チャンネルを SDP(Session DescriptionProtocol)[18] によって記述し，SAP(Session Announce Protocol)[19] を用いて広告する機能を持つ．このことから DVBS は QuasiVoD にカテゴリ分けすることができる．DVBS では，DVTS のクライアント (xdvshow･dvrecv) を用いて計算機上のディスプレイや，IEEE1394[20] を用いて計算機と接続された DV 機器を用いて視聴する．

DVBS は DV フォーマット特有の特徴として，データ送信レートが 32Mbps と高く，UDP/RTP を用いているため，輻輳制御や欠損データの再送ができない．そのため，ネットワークへの負荷を最小限にするためにバーストトラフィックを発生させないよう，一度に送信するバッファサイズを細かく設定されている．

3.2.2 DVBS のアプローチと実装手法

DVBS が解決した問題点にサーバ・クライアント間での DV データ入出力同期問題である．サーバからの DV データの送信レートが DV 再生ビットレートと比較して高い場合，クライアントの受信バッファを徐々に消費し，最終的に超過させ，映像にノイズが発生する．DV 再生ビットレートと比較して低い場合はクライアントの受信バッファは枯渇し，映像は停止してしまう．一般的にこの問題は，クライアントからの要求に応じたデータ量をサーバが送信するアルゴリズムや，クライアントが動的に可変可能なバッファを確保することにより解決されている．しかし，前者の場合は IP マルチキャストを用いた放送型配信をサポートする場合，個々のクライアントへのパケット到達時間が一定でないため適用することができない．また，後者の場合はクライアントに広大なバッファ領域を強いることになるため，現実的ではない．特に DV のように再生ビットレートが高い映像フォーマットの場合は，数十秒足らずでバッファは枯渇する．この問題を DVBS は DV 再生ビットレートと DV ストリーム転送レートを近似させることで解決している．

図 3.1に具体的な DVBS の処理の流れを示す．DVBS では，基本となるデータ入力サイズと DV データ入力・転送周期におけるインターバルをあらかじめ決定しておく．そして，DV データ入力ハンドラを実行し，実行された時刻を記録し，DV データの転送ハンドラを実行し，転送処理が終わった時刻を記録する．これらの時間の差分から次の DV データ入力ハンドラの際に読み込むデータサイズの誤差補正を行う．DVBS ではこのアルゴリズムを用いることによっ

て，DV 再生ビットレートと DV ストリームの転送レートを近似させ，クライアントに途切れのない映像を提供している．

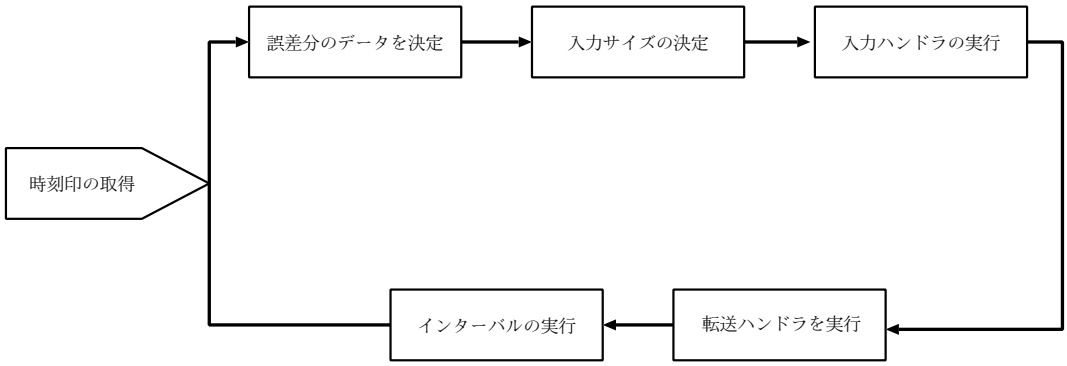

図 3.1: DVBS の処理の流れ

3.3 まとめ:関連する VoD システム

本章では，映像制御と言う視点から WMT を，大容量映像データ転送と言う視点から DVBS を VoD の関連研究とし挙げ，それぞれの特徴と実装について述べた．WMT は，キーフレームを任意の間隔で設定することによって，簡単な映像再生制御機構を実現している．DVBS は大容量映像データ転送の際に問題となるサーバ・クライアント間での入出力同期問題を，データ転送レートと DV 再生ビットレートを近似させることで解決している．

次章では，WMT と DVBS の映像配信形態と VoD の要素技術の組み合わせを，2.4節で前述した，映像配信形態に適した VoD の要素技術の組み合わせ方と比較をする．そして，そこから導き出される関連研究の問題点のまとめ，および本研究の目的実現へのアプローチを述べる．

第4章　問題点のまとめとアプローチ

2章で述べた True VoD に適している要素技術の組み合わせと，3章で述べた関連研究が抱える問題点をまとめ，それぞれの技術的な原因を明らかにする．1章で述べたパフォーマンスの高い計算機資源の有効活用と高品質映像の制御実現に対する本研究のアプローチを述べる．

4.1　問題点のまとめと原因の究明

本説では，2章で述べた True VoD に適した要素技術の組み合わせと，3章で述べた WMT と DVBS の問題点をまとめ，時間粒度の細かい映像制御，パフォーマンス資源の有効活用性の視点から技術的な原因を明らかにする．

4.1.1　WMT と DVBS が使用する要素技術の組合わせ

表 4.1に WMT と DVBS の VoD の映像配信形態，トランスポート層における転送プロトコル，利用している映像フォーマットを示し，表 2.5と比較し，本研究が考える映像配信形態に適した要素技術の組み合わせと比較する．

表 4.1: WMT と DVBS が利用している要素技術

	配信形態	映像フォーマット	転送プロトコル
WMT	True VoD	WMV(フレーム間圧縮フォーマット)	TCP
DVBS	Quasi VoD	DV(フレーム内圧縮フォーマット)	UDP/RTP

表 2.5に示した通り，本研究が考える映像配信形態に適した要素技術の組み合わせは，True VoD にはフレーム内圧縮フォーマットと UDP/RTP，Quasi VoD にはフレーム間圧縮フォーマットと TCP の組み合わせである．しかし，表 4.1で示した通り，WMT と DVBS は本研究の考える映像配信形態に適した要素技術の組み合わせとは異なる組み合わせで実装されている．2.2.1節で述べたように，本研究の目的である，高品質映像の時間粒度の細かな映像制御を実現する映像配信形態は True VoD である．時間粒度の細かい映像制御が要件である True VoD にはフレーム内圧縮フォーマットが適しており，True VoD にはフレーム内圧縮フォーマットの映像を用い，UDP/RTP を使用して転送するのが適切である．次節移行に詳細を後述するが，映像制御に適した映像フォーマットと，計算機環境とデータリンク帯域を考慮したトランスポート層プロトコルを用いなければ，本研究が目的とする高品質な映像フォーマットで，細かな映像制御機能を備える VoD システムを構築することはできない．

4.1.2 時間粒度の細かい映像制御に関する問題点

映像の細かな制御を可能にするために必要な要素は以下の 4 点である．

1. 要求されるフレームの特定
2. 個々のフレームでの画像復号
3. 再生速度の調節
4. クライアント毎に独立した映像配信

また，以上の 4 点を満たした上で，映像編集作業や試聴では，映像制御要求の発行から反映までの即時性と，制御要求データの正確性が要求される．

第 1 項は，2.1.4節で述べた通り，フレーム内圧縮フォーマットは個々のフレームのデータサイズが一定で，タイムコードや ATN などからフレームの特定が可能だが，フレーム間圧縮フォーマットでは個々のフレームのデータサイズが一定ではない．そのため，映像データの再生開始からの時間からフレームを割り出すなどの手法がとられている．しかし，GOP のサイズは可変なため，再生時間と単独で画像への復号が可能な I フレームが確実に一致する保証がない．そのため，フレーム間圧縮フォーマットは，フレーム内圧縮フォーマットと比較して，煩雑で必要な処理が増加するため，即時性と要求されるフレームへの確実性が低い．

第 2 項は，2.1.4節で述べた通り，フレーム内圧縮フォーマットでは個々のフレームで画像の復号が可能だが，フレーム間圧縮フォーマットでは GOP データが全て揃わなくては，I フレーム以外のフレームから画像への復号ができない．そのため，フレーム間圧縮フォーマットは GOP の全データ量と，計算処理が必要なため，フレーム内圧縮フォーマットと比較して即時性に欠る．

第 3 項は，フレーム内圧縮フォーマットの場合は，単純なフレームレートの変更，または，フレームの間引きによって簡単に実現することができる．ただし，フレームレートの変更による再生速度の上限は二次記憶装置のデータ読み込み速度によって制限される．しかし，前述した通り，フレーム間圧縮フォーマットは GOP のデータサイズが可変なため，確実に I フレームを捕らえることの難しさと，フレーム複合処理に必要な計算量から，二次記憶装置のデータ読み込み速度とは無関係に任意の速度調節ができない．

第 4 項は，映像の再生制御要求は，個々のクライアントが個別の要求を持つため，クライアント毎に独立した映像ストリームの生成が必要となることを意味する．そのため，IP マルチキャストなどを用いたデータの一斉配信で個々のクライアントが要求する映像制御の実現は不可能である．また，映像ストリームを生成するタスク，ストリーミングセッションが Quasi VoD と比較して格段に増加する．

そのため，映像データを送信するサーバでは，多数のクライアントを想定し，計算機資源利用効率の良い VoD システムを構築しなくてはならない．詳細を次節で述べるが，大容量映像フォーマットを用い，多数のクライアントに安定した映像を配信するのは困難である．

4.1.3 大容量映像データ転送時の計算機資源の有効利用に関する問題点

サーバから，クライアントに対して映像を送信し，クライアントで途切れない映像再生を可能にするために必要な要件は, サーバ上の映像データ送信タスクが，クライアントの映像受信バッファを過不足なく満たしておくことである．このように，あるタスクが定められた時間以

内に処理を終了させる必要がある場合に，定められた時間の上限を Dead Line と呼び，Dead Line を厳守し実行される処理を実時間処理という．

大容量データの読み込み・送信を必要とする VoD システムで，多数のクライアントを想定した場合，現状では以下の 2 点の問題点から，クライアントの受信バッファを過不足無く満たし，高い計算機利用効率を確保するのは困難である．

1. 汎用マルチタスク OS の粗雑なタスク管理
2. 二次記憶装置の不安定な読み込み速度

3.2.2節で述べた DVBS のアプローチの通り，DV などの大容量の

一方，DVBS のようにサーバ側で映像データ送信レートを制御する方法もあるが，DVBS は IP マルチキャストを用いた放送型映像配信であったため，複数の映像ストリームの映像が乱れること無くクライアントへデータを配信する仕組みとして，安定したデータ入力は考慮されておらず，実現もされていない．

映像の制御要求の実現と高い実時間性を持つ処理の保護には，多数の映像ストリーム生成タスクを管理し，クライアントの受信バッファが枯渇するまでの間，すなわち Dead Line までに，次の再生に必要なデータ転送を終えなければならない．

第 1 項は，サーバ側で映像データの送信レートを調節する場合，多数の大容量映像のデータ読み込み・送信を行うタスクが，個々のクライアントが持つ受信バッファが枯渇する Dead Line までにその処理終えるための厳密な時間管理に基づくタスク管理機構を備えていないことを示している．原因は通常の汎用マルチタスク OS がタイムシェアリングシステムを採用し，ハードウェアからの非同期なシグナルなどを優先して実行してしまう点にある．

図 4.1に非同期に発生したタスクによって，Dead Line を持つタスクが，Dead Line を保証できない場合を示す．横軸に CPU 時間を示していて, 個々のタスクは均一な CPU 処理時間 (T) を割り当てられるシステムとする．実線の矢印は TASK C が発生した場合の TASK A,B,C の CPU 割り当て時間をしめし，点線の矢印は TASK C が発生しなかったときの TASK A の CPU 割り当て時間を示す．その際 TASK A のみが Dead Line を持っており，一定時間内に処理を終えなくてはならないとする場合,TASK A と B のみが動作している場合に TASK A は Dead Line までに処理を終えることができる．しかし, ハードウェアからの非同期な割り込みによって TASK C が発生した場合に TASK A は Dead Line を守ることができなくなる．

第 2 項は，二次記憶装置にハードディスクを用いた場合に起こる問題点である．大容量のファイルサーバはほぼすべてハードディスクを用いて構築され，1 ファイル当りのデータ量が多い映像データは，ハードディスクに格納される．ハードディスクは，データの読み込み速度は製品や規格によって大きく異なる上，セクタによって読み込み速度が大きく異なるため，非常に不安定である．IEEE1394 がサポートするアイソクロナス転送のように単位時間に対して一定のデータ入力を期待することができない．すなわち，ハードディスクを使う以上，データの読み込みに必要な時間は予測することができない．

そのため，第 1 項の問題点である汎用マルチタスク OS 上での粗雑なタスク管理と組み合わされることで，映像データの読み込み・送信までにかかる時間の揺らぎはさらに大きくなる．

これらの問題を抱えたまま，多数の映像ストリーミングセッションを管理する場合，映像データの読み込み・送信までの Dead Line を厳守することができず，映像の乱れを招く．図 4.2は映像データ読み込み・送信周期の回数を横軸にとり，その処理が終了するまでにかかる時間を縦軸にとった図である．

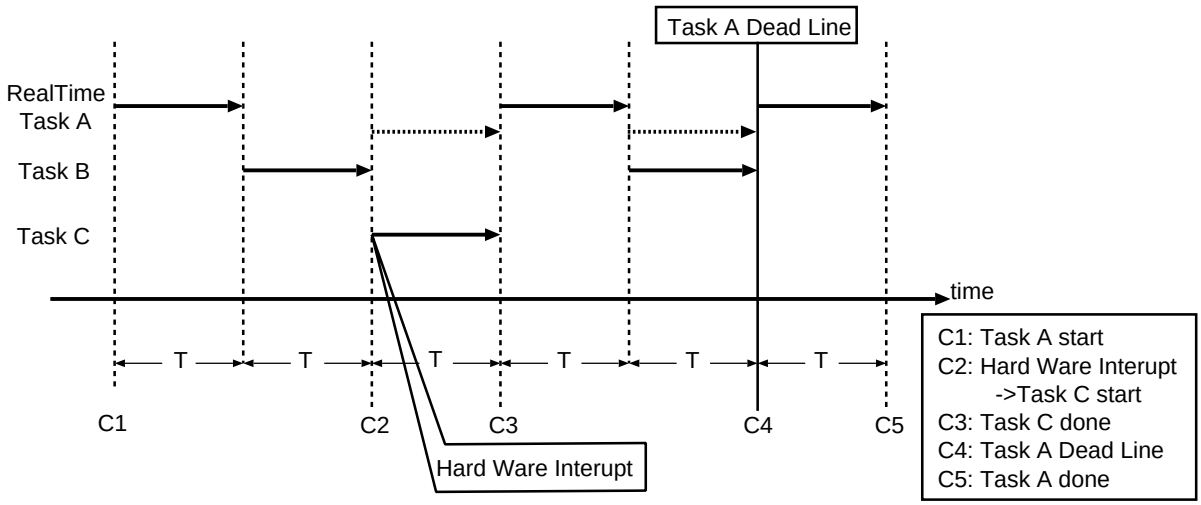

図 4.1: Dead Line を守れない場合の例

その際にこれまで述べてきた 2 点の問題点によって,処理完了までの時間が不安定な場合,DeadLine を越えてしまうことで映像の乱れが発生することを示している.

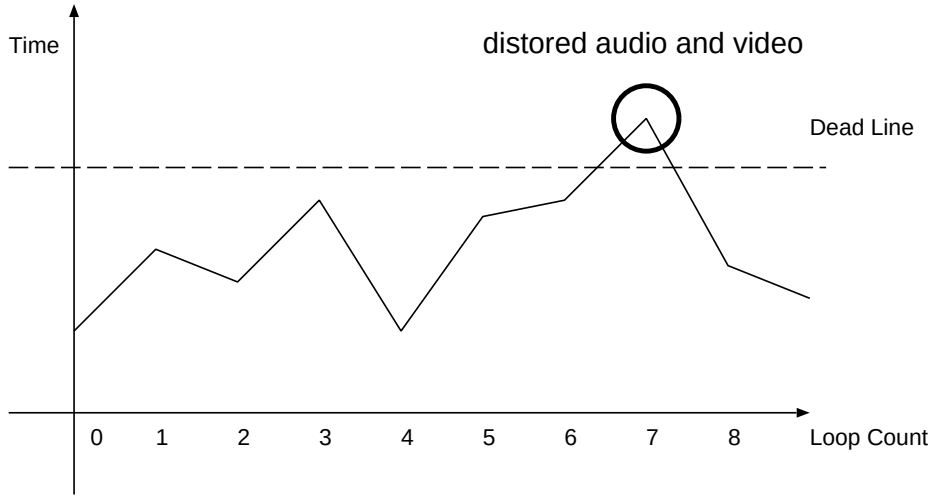

図 4.2: 映像データ読み込み・送信周期毎の処理時間

このように，汎用マルチタスク OS の粗雑なタスク管理，映像データを格納したハードディスクの不安定な読み込み速度を解決せずに，安定した処理時間での映像データ読み込み・送信は実現できない.

4.2 目的実現へのアプローチ

本節では, 緻密な映像制御を要求する True VoD システム実現へのアプローチを述べる．また,True VoD システムの実現によって発生する映像ストリームセッションの増加に対して, 資

源の有効活用と True VoD の実現に対するためのアプローチを述べる．

4.2.1 DV フォーマットの利用

2.2節で述べた通り，True VoD 環境で頻繁に要求される再生制御コマンドの大半は，ユーザが目視できる動画の最小単位であるフレームの制御によって実現することができる．そのため，4.1節でも述べた，細かなフレーム操作に適しているフォーマットとして，フレーム内圧縮フォーマットを利用する．そして，フレーム内圧縮フォーマットで TV 放送と同等品質以上で，現在のデータリンクの帯域で，広く利用することがこれから期待できる SDDV フォーマットを用いる．また，DV フォーマットを用いることで，HDDV を配信可能な計算機環境とデータリンク帯域が確保された場合の HDTV 相当の画質への移行が容易である．

4.2.2 タイムコードの利用

フレーム内圧縮フォーマットを用いることで，これまでデジタル映像を扱うに当たって考えられてきた様々なフレーム識別方法を用いることができる．個々のフレームをフレーム識別可能な方法の中で本研究はタイムコードと ATN に着目した．ATN は個々の DV データ内ではユニークなフレーム番号を識別できるが，他の DV データとの時間的な整合性をとることはできない．また，タイムコードは ATN への変換が計算によって可能なため，同期信号などを用いて，同時刻に撮影された複数の映像を，編集作業上で同期させる用途などへの応用力がある．そのため，個々のフレーム識別方法にはタイムコードを用いる．

4.2.3 UDP/RTP の利用

SDDV フォーマットはネットワークを介してデータ送信を行った場合，映像データそのものが使用する帯域に加え，IP 通信を行うための各種ヘッダを含めると約 32Mbps になる．4.1.3節で述べたように，現在の計算機環境では TCP を用いて SDDV の映像データを大量にバッファリングすることは困難である．また，クライアントの再生制御要求の送信から，その反映までの即時性を要求するため，UDP/RTP を使用する．

4.2.4 実時間処理の利用

4.1.3節で，単位時間あたりのデータ量が多い映像フォーマットを用いた True VoD システムを，汎用マルチタスク OS 上で実装するのは困難であることを述べた．SDDV の使用は，上記の場合に当てはまる．そのため，実時間処理を用いてサーバ・クライアント間のデータ入出力同期を実現することを提案する．実時間処理を用いることで，Dead Line を厳守すべきタスクを，他のタスクからの割り込みから保護することで実時間性を保証する．個々の DV ストリーム生成タスクを実時間処理を用いて実装することで，安定したデータ供給を実現できる．

マルチタスク OS 上で,Dead Line を持つタスクが必要な時間に, 必要な資源を確保することで, 計算機・ネットワークの資源は最大限に有効活用され, 結果としてユーザの計算機環境の安定化につながる．特に, 高品質映像フォーマットを用いたストリーミングでは, サーバ・クライアント上の計算機資源, ネットワーク上の資源を多量に消費するため実時間処理への考慮は不

可欠な要素である.

4.2.5 まとめ:本研究が提案する True VoD システムにおける要素技術の組合わせ

本章では，WMT，および DVBS の映像配信形態と VoD 要素技術の組み合わせと，本研究が理想とする映像配信形態と VoD 要素技術の組合わせの差違について議論をした．そこから導き出された問題点を，時間粒度の細かい映像の制御に関する問題と，大容量映像データを多数転送する際の問題点についてまとめた.

本研究の目的である高品質映像の時間粒度の細かな制御と，パフォーマンスが計算機資源の有効活用を両立する VoD システム実現に対して，映像配信形態と具体的な要素技術の組合わせを表 4.2にまとめる.

表 4.2: 本研究の提案する VoD システムが活用する要素技術

配信形態	映像フォーマット	フレーム探索方法	転送プロトコル
True VoD	SDDV(フレーム内圧縮フォーマット)	タイムコード	UDP/RTP

表 4.2と表 2.5を比較することで，本研究が考える True VoD に最適な要素技術の組み合わせを実現していることがわかる.

次章では，これらのアプローチを基に，計算機資源の有効活用のために実時間処理を用いた True VoD に必要な要件と機能を洗いだし，本研究が構築する True VoD システムの設計について述べる.

第5章　True DV VoDシステムの設計

本章では，4.2節のアプローチで述べた実時間処理を利用したVoDシステムと，DVストリーミング中の再生制御を実現するための設計を行う．設計に際しての要件，必要な機能を洗いだし，設計を行う．

5.1　設計要件

1.2節で述べた要求と，4.2節で述べたアプローチを基に，DVを用いたTrue VoDを設計する．その際に必要となる要件と，それを満たすための機能を以下にまとめる．

- 2.2章で述べたTrue VoDで頻繁に利用される再生制御コマンドの実装
  - サーバ・クライアント間における再生制御要求の送受信
  - 任意のDVフレームの特定と読み込み
  - 任意のDVデータ送信レート選択
- 安定したデータ入力・送信
  - DVデータ読み込み，DVデータ送信の1サイクル中の計算機資源占有
  - 複数映像ストリーム生成時の実時間処理
- 情報家電上での実装の考慮
  - 汎用的な組み込みOSを備えたハードウェア機器を意識した設計と実装環境

5.2　設計概要

本節では前節の要件と，機能を実装するために行った設計の概要を述べる．図5.1に設計の概略図を示す．本システムは大きく分けて送信部と受信部によって構成される．送信部はDVデータ読み込み・送信部とそのタイミング制御部，フレーム制御部によって構成され，受信部はDVデータ受信・再生部，再生制御要求部によって構成される．以下に各構成部分の詳細を述べる．

5.3　送信部

送信部は，二次記憶装置上のDVデータ読み込み部と，IPネットワークに対してのデータ送信部，読み込み・送信のタイミング制御部, フレーム制御部によって構成される．送信部はリアルタスクとして実行され，タイミング制御部によってタスク管理を行う．

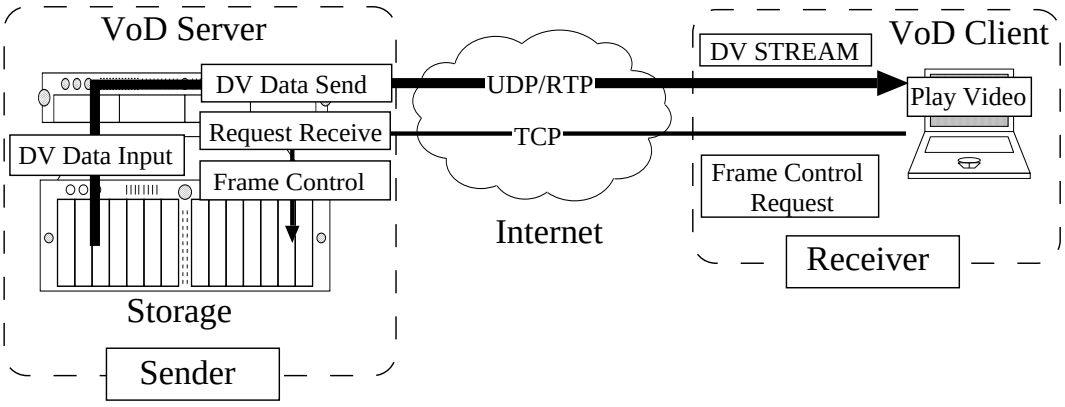

図 5.1: 設計概略図

DV データ読み込み部分と，IP ネットワークに対するデータ送信部分については，通常のファイル入出力，UDP/RTP を用いたデータ送信と同様なため詳細については言及しない．

5.3.1 DV データ読み込み・送信のタイミング制御

3章で述べた通り，サーバの DV データ送信ビットレートと，DV 再生ビットレートが等しく無ければクライアントの受信バッファは枯渇，もしくは超過する．そのため，1 つの DV フレームの読み込み・送信に費やす時間を，1 つの DV フレーム再生時間未満で実行し，そのサイクルを DV の再生フレームレートと同じ間隔で行う必要がある．また，複数の DV データ読み込み・送信タスク，およびその他のタスクが協調して実行できなくてはならない．図 5.2に複数の DV データ読み込み・送信タスクがのタイミング制御を示す．

2 つの DV データ読み込み・送信タスク (STREAMING TASK)A,B と，その他のタスク (Other TASK)C，ハードウェアインタラプトによって発生する TASK D がある．Dead Line を持つ実時間性を要求する処理である DV データ読み込み・送信は，他のタスクである TASK C や，ハードウェアインタラプトによって非同期に発生する TASK D よりも優先し，Dead Line を厳守する．また，必要な資源を連続して占有することによって，可能な限り迅速に DV データ読み込み・送信の 1 周期に費やす時間の短縮をする。これによって，他の実時間性を要求する処理の Dead Line 厳守の補助を図る．

5.3.2 フレーム制御部

フレーム制御部では，クライアントからの再生制御要求を受信し，要求に応じて DV フレーム読み込みを制御する.2.1.2節で述べた通り，フレーム内圧縮フォーマットは，個々のフレーム情報のみで映像の復号が可能で，個々のフレームはタイムコードによって識別することができる．図 5.3にフレーム制御の流れを示す．DV データ読み込みを行う際に，SUBCODE フレームを抽出し，現在読み込み・送信中の DV データのタイムコードを格納する．再生制御要求中のタイムコードや N フレーム前後などの要求をタイムコードに変換し，格納されているタイムコードと照合することで，次に読み込むべき DV フレームを導き出す．

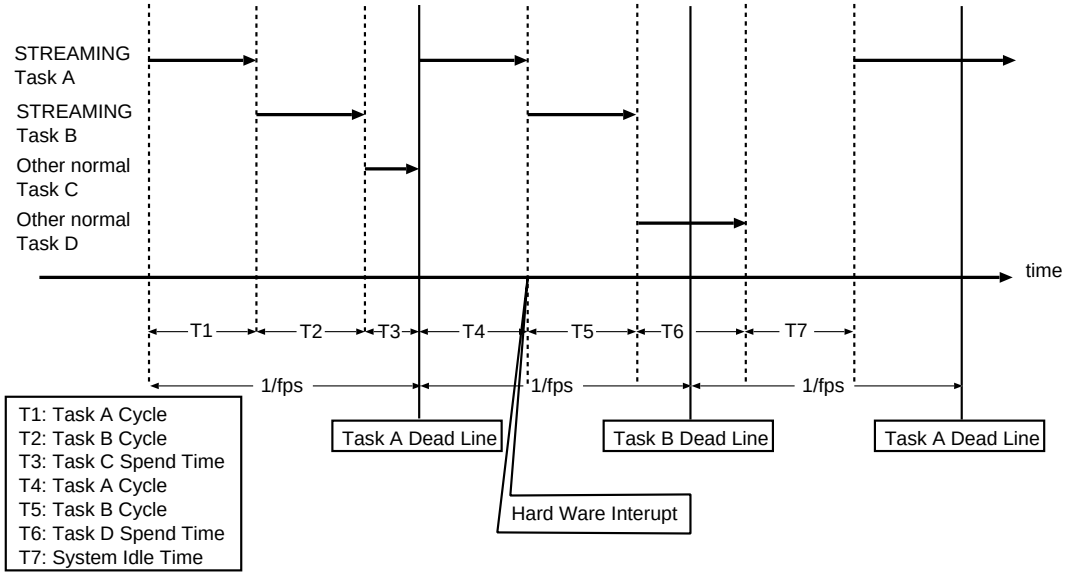

図 5.2: DV データ読み込み・送信タスクのタイミング制御

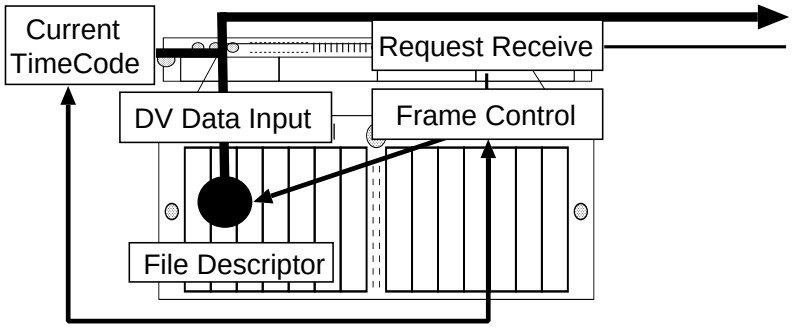

図 5.3: フレーム制御の流れ

5.4 受信部

受信部は DV データ受信・再生部と，再生制御要求部によって構成される．DV データ受信・再生部によってディスプレイや IEEE1394 を備えた AV 機器に対して出力を行い，再生制御要求部によってユーザからの再生制御を受付，サーバに対して送信する．ユーザは現在再生されているフレームの情報として，タイムコードを参照し，再生制御の際の参考データとする．DV 受信・再生部には DVTS のクライアント (xdvshow・dvrecv) を利用するため，その部分についての詳細は深く言及しない．

5.4.1 再生制御要求部

図 5.4にユーザからの再生制御要求の流れを示す．再生制御要求部では，ユーザからの再生制御コマンドの受付を行い，サーバへ送信する．その際，その要求はフレーム操作を行う場合にタイムコードへと変換される．フレーム制御の要求を一様にタイムコードで扱うことによっ

て，サーバ側でのフレーム制御処理を簡略化する．また，フレーム制御要求は，パケットロスによって制御要求が消滅してはならないため，TCP を用いる．

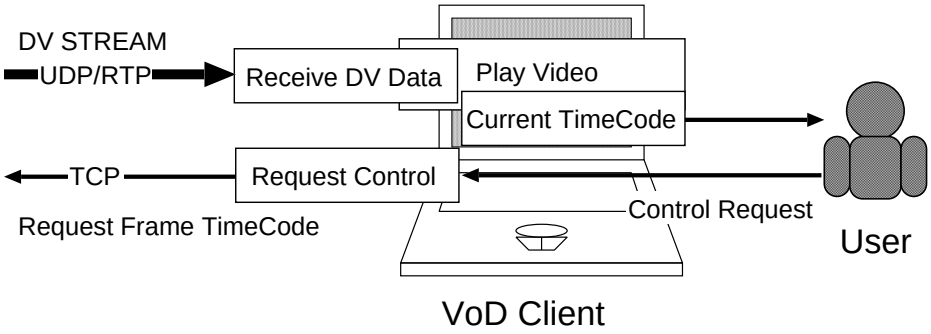

図 5.4: 再生制御要求の流れ

5.5 まとめ:設計

本章では，4章でまとめたアプローチと具体的に使用する要素技術を基に，本システムが必要とする要件と機能を洗いだし，設計を行った．設計に際して，サーバ側の送信部とクライアント側の受信部に，必要な構成要素とその挙動を定義した．

次章では，これらの構成要素の具体的な実装について述べる．

第6章　実時間処理を用いたTrue DV VoDの実装

本章では4章と5章で行ったアプローチと設計を基に行った実装について述べる．

6.1　実装環境の選択

本節では，本研究が用いるVoDの要素技術と実装環境について述べる．具体的には，再生制御に適した映像フォーマット，ハードウェア機器に組み込むことを考慮したリアルタイムOSの選択，パフォーマンスの高い映像配信に適した実時間処理の選択について述べる．

6.1.1　実時間処理とリアルタイムOS

実時間処理を可能にしているOSをリアルタイムOSと言う．リアルタイムOSは汎用マルチタスクOSと比較して，実時間処理を中心として以下の特徴を備えている．

- 厳密な時間管理に基づくタスク管理
- 粒度の細かい優先度に基づくタスク管理 (Priority制御)
- 任意タスクのRound Robin制御
- プログラム中の特定ルーチン, もしくは特定タスクのDead Lineの死守 (Periodic制御)

リアルタイムOSの設計には大きく分けて二つのアプローチがある．図6.1に二つのアプローチの概略を示す．リアルタイムOSの核となるkernelを作成した上で，汎用API・デバイスドライバ・アプリケーションを追加していくアプローチと，汎用マルチタスクOSにリアルタイム処理を追加するアプローチである．前者は開発コストが大変高く，サポートしているCPUアーキテクチャ・デバイスドライバ・汎用APIが不足している．後者は前者と比較してタイミング制御の時間粒度が荒い傾向にあるものの, 普及している汎用マルチタスクOSの持つ汎用API・デバイスドライバ・ライブラリ・アプリケーションを最大限に活用できるメリットがある．

6.1.2　実時間処理の選択

実時間処理には複数のスケジューリング方法があることは6.1.1節で述べた通りである．その中で本研究は優先度付きPeriodic制御を用いる．優先度付きPeriodic制御を用いることで，再生制御要求を受け付けるタスクを，最優先度タスクに割り当て，DVデータ読み込み・送信タスクを次点の優先度に割り当てることで本研究の要求を満たすことができる．

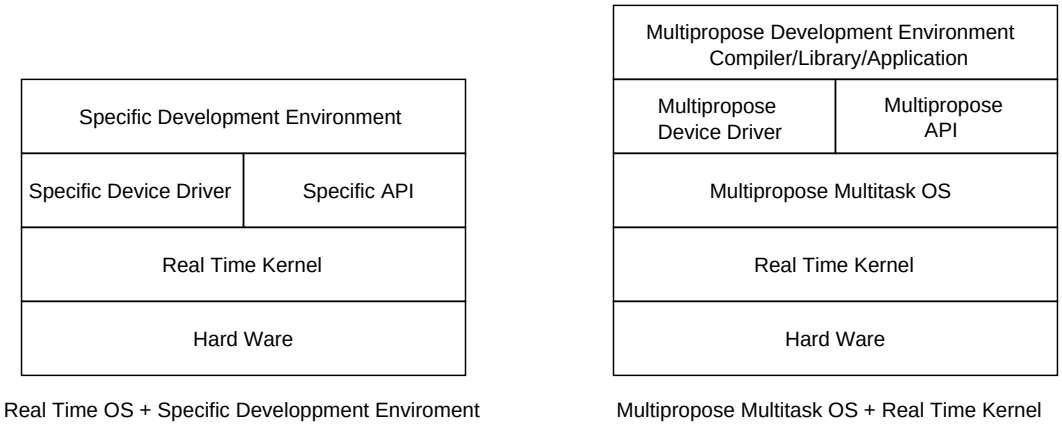

図 6.1: リアルタイム OS 作成の二つのアプローチ

Round Robin 制御は再生制御要求の受信後，次のタスクが任意のフレーム探索タスクになる保証が無い．そのため再生制御要求が反映されるまでの即時性に欠る．Priority 制御の場合は，他の DV 送信タスクの Dead Line を保証しない．

6.1.3 リアルタイム OS の選択

本研究は，5.1節に列挙した要求を満たすリアルタイム OS として，組み込み型 OS 上での実装実績や，実時間処理を付加した実装の候補の多さから Linux をターゲットにした．Linux に実時間処理機能を付加しているシステムの中から、本研究では、FSMLab RTLinux[21]，ムービングアイ ART-Linux[22] に着目した．以下に RTLinux と ART-Linux の特徴を述べる．

- RTLinux
  RTLinux はハードウェアの上位レイヤーとして RT Kernel を実装している．RT Kernel は Linux Kernel を最低優先度タスクとしてとらえ，Linux Kernel は RT Linux をハードウェアとして認識する．そのため，RT Kernel はリアルタイムタスクとリアルタイムタスクを同種のタスクとして制御することを可能にしている．また，Linux Kernel が生成するプロセスからリアルタイムタスクへの通信方式として RT FIFO を定義することで，エンドユーザへのリアルタイムタスク制御を実現している．この設計を図 6.2に示す．斜線の部分が RTLinux によって拡張された機能を示している．なお，RTLinux 上でのリアルタイムタスク生成は，Linux Loadable Module と同様の開発手順が適用される．

  RTLinux は以下の利点を持つ．

  – n(ナノ) 秒単位でのタスク管理が指定可能
  – リアルタイムタスクは Linux Loadable Module と同様の作成手順である
  – RT FIFO を用いて Linux ユーザプロセスからリアルタイムタスクにアクセスできる

  また，RTLinux は以下の欠点を持つ．

  – Linux Loadable Module と同様の作成手順なため，LibC API などの汎用 API が使用できない

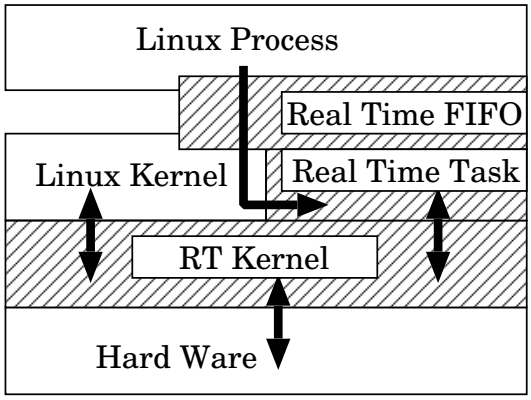

図 6.2: RTLinux のデザイン

  - 優先度逆転抑止機構を備えていない
  - Linux Kernel が保護タスクでないため，他のリアルタイムタスクとの間でデッドロックなどでクラッシュし，制御不能に陥る場合がある
  - リアルタイムタスクと通常の Linux プロセスが別構造として記述されているため、優先度逆転を抑止できない

- ART-Linux
  ART-Linux は RTLinux とは異なり，実時間処理を Linux Kernel を改変することで実現している．そのため，以下の利点を持っている．

  - Linux アプリケーションのバイナリ互換
    これまでの Linux アプリケーションをも実時間処理に適用できる
  - Linux デバイスドライバのソース互換

  しかし,Linux Kernel を基調にしているため,Linux が管理可能な時間粒度よりも細かな時間管理を行うことができない．そのため m(マイクロ) 秒単位でのタスク管理が ART-Linux の限界である．ART-Linux は, 割り込みハンドラの周期実行, 非同期な割り込み処理の時間制限をもうけることによって, デッドラインの保証を行っている．デバイスドライバからカーネルに対する割り込みハンドラの登録要求を受け, 周期的に再会されるリアルタスクを生成し, タスク内で割り込みハンドラを実行する．そのため, 優先度継承を行うことが可能で,RTLinux と異なり，優先度逆転を抑止することができる．これらの機能は非リアルタイムタスクへの優先度継承機能を持つリアルタイムロックによって実装されている．リアルタイムタスクの優先度制御において，実行中のリアルタイムタスクと休眠中のリアルタイムタスクが相互にリンクし，依存関係のあるリアルタイムタスクへのタスクスイッチングの際に優先度の継承を行うことで，優先度逆転を抑止している．図 6.3にそのデータ構造を示す．TASK A から TASK C へとタスクスイッチングされた場合に，TASK A の優先度を継承し，プログラマが望む優先度付けが意図しない方向へずれることを抑止している．

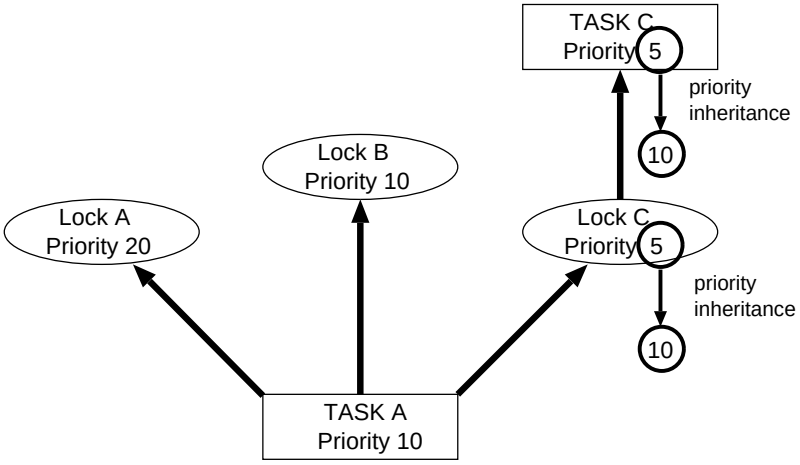

図 6.3: ART-Linux の優先度継承

本システムでは優先度付き Periodic 制御を用いたリアルタイムタスクが複数生成される．そのため，以下の二点の理由より，本研究では、ART-Linux を採用した．

- 優先度逆転の抑止
  本研究では，再生制御要求受付タスク，複数の DV ストリーム生成タスクがリアルタイムタスクとなる．それぞれのタスクが計算機資源を DV データの読み込み・送信の周期処理中占有することで，個々のタスクに対するパフォーマンスをハードウェア限界まで近づけ，個々の周期処理時間の短縮・安定化を目指す．そのため，優先度逆転は即処理時間の不安定化を招く．特に二次記憶装置の場合は資源を占有した場合にもデバイスの特徴上不安定性は回避できないため，優先度逆転は起ってはならない事態である．

- 実時間処理・通常処理の入れ替え
  また，ART-Linux はシステムコールによって実時間処理・通常処理の入れ替えが可能である．そのため，フレームレートの変更など，一度 Periodic 制御の Dead Line 設定を変更しなくてはならない場合に対して柔軟に対応できる．

6.1.4 実装環境のまとめ

これまで述べてきた実装環境を表 6.1以下にまとめる．本研究はこの実装環境下で，5章で述べた設計に基づき送信部・受信部の実装を行った．

表 6.1: 開発環境

OS/Destribution	Linux Kernel 2.4.20 based ART patch Vine 2.6.1 Release
プログラミング言語	C 言語
実時間処理	優先度付き Periodic 制御
コンパイラ	gcc 2.95.3
映像フォーマット	SDDV フォーマット

6.2 送信部

送信部はパラメータ設定部，二次記憶装置上の DV データ読み込み，および IP ネットワークへのデータ送信部，タイミング制御部，フレーム制御部によって構成される．各部の処理は 5.3節の通りである．

図 6.4に送信部の構成要素の関係図を示する．基本的な処理の流れは，パラメータの設定を行い，実時間処理にプログラムを変化させる．その後ループがスタートし，DV データを読み込みメモリ上にマップする．SUBCODE の場合はタイムコードをデコードし変数に格納する．メモリ上にマップされた DV データを送信し，クライアントからの再生制御の有無を調査する．再生制御要求がある場合は，要求されるタイムコードと，格納されている現在のタイムコードを比較し，そこから導き出される次に送信すべき DV データへと，DV データ読み込みに用いるファイルディスクリプタを介して読み込むポインタを操作し，ループの開始へ戻る．

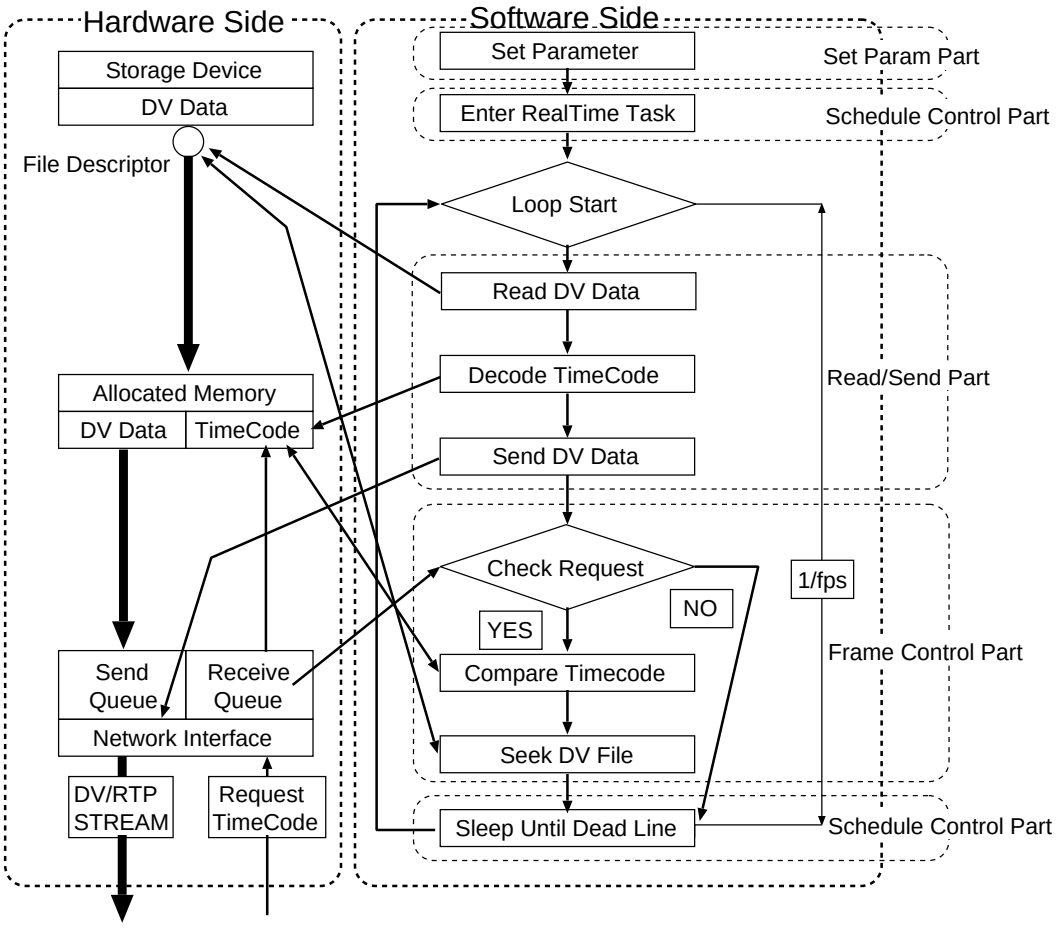

図 6.4: 送信部構成要素の関係図

次節以降に各部の実装について述べる．

6.2.1 パラメータ設定部

本実装では送信部が持つパラメータとして，*dvsend_param* 構造体を定義した．*dvsend_param* 構造体には，読み込みフレーム中の SUBCODE からタイムコードをデコードし，格納する *timecode* 構造体を定義している．図 6.5に *timecode* 構造体と，*dvsend_param* 構造体のメンバを示す．各変数に格納される値は，図 6.5中のコメントの通りである．

```
struct timecode{
  /* TIMECODE PARAM */
  int hh;  /* hour */
  int mm;  /* munute */
  int ss;  /* second */
  int ff;  /* frame */
}

struct dvsend_param {
  /* DV FORMAT PARAM */
  int format;      /* NTSC or PAL */
  double fps;      /* frame/sec: NTSC is 29.97fps. PAL is 25fps */
  struct timecode current_time; /* current reading frame timecode */
  struct timecode request_time; /* request frame timecode */
  int frame_drop;  /* for FastForward and Rewind */

  /* 中略 */

  /* FILE PARAMTERS */
  char *filename;  /* DV data file name */
  int file_fd;     /* file descriptor of DV data file */

  /* READ PARAMTERS */
  int readsize;    /* Frame Size of NTSC or PAL */
  int read_flag;   /* read from IEEE1394 or File */
};
```

図 6.5: timecode 構造体，および dvsend_param 構造体

6.2.2 DV データ読み込み・送信部

本実装では DV データの読み込みを DV フォーマットの放送方式に応じて読み込みサイズを調節する．NTSC の場合は 120,000byte，PAL の場合は 144,000byte で，*dvsend_param* 構造体中の *readsize* に格納する．この値は，それぞれの放送方式に応じた 1 フレームのデータサイズである．1 フレーム分のデータを一度に読み込む理由は，フレーム制御の際に調節が容易にす

るためと，read 関数の呼び出しよって発生する負荷を最小限にするためである．図 6.6に DV フォーマットの放送方式に応じたパラメータ設定を示す．

```
if (dvsend_param->format == DV_FORMAT_PAL) {
  dvsend_param->readsize = 144000;
  dvsend_param->fps = 25;
} else if (dvsend_param->format == DV_FORMAT_NTSC) {
  dvsend_param->readsize = 120000;
  dvsend_param->fps = 29.97;
} else {
  return(-1);
}
```

図 6.6: DV フォーマット放送方式毎のパラメータ設定

なお，*fps* については，6.2.4節で後述する．読み込んだ DV データは，80byte の DIF block 毎に分割され，DIF block ID を基に振り分けを行う．その中で SUBCODE に当たる DIF block に関しては，タイムコーをデコードし *current_time* に格納する．

DV フォーマットでは SUBCODE 中のパックと呼ばれる 5Byte のデータ集合にタイムコードが格納されている．図 6.7に示す通り，SUBCODE 内の 31Byte 目が 0x13 の場合にそこを先頭として 5Byte がパックであり, 後続する 4Byte にタイムコードは記載される．タイムコードはデータがテープに記録された際に記載され、「時間:分:秒:フレーム番号」の順に表される．また，NTSC 方式の場合，Drop Frame Flag によって，30fps と 29.97fps の場合を識別する．

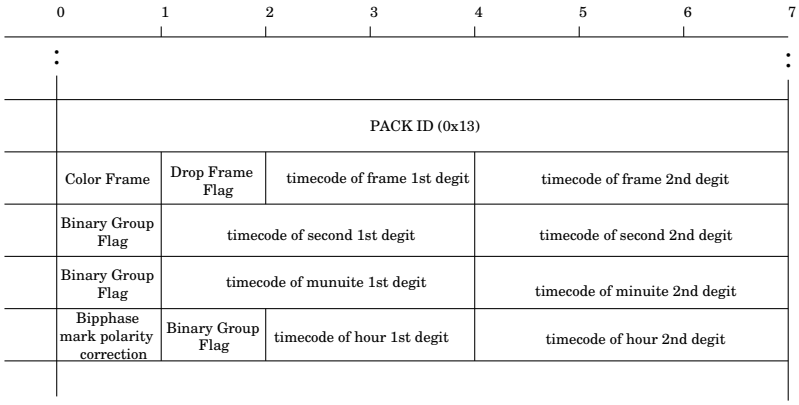

図 6.7: DV TimeCode PAC

6.2.3 フレーム制御部

本実装ではフレーム制御をタイムコードを用いて行う．図 6.8に再生制御要求部によって送信されたタイムコードを受信し，現在読み込み中のタイムコードとの差分をとることによって，*lseek* を行う際の *offset* を導き出す．DV データ読み込み・送信部において，1 フレーム単位で

の読み込みを行っているため，*lseek* を行って，通常の場合と代わりなくファイルの読み込みを行い，クライアントが要求する映像フレームを送信することができる．

```
/* current reading frame */
current_frame = (((dvsend_param->current_time.hh * 3600) +
                  (dvsend_param->current_time.mm * 60) +
                  (dvsend_param->current_time.ss)) *
                   dvsend_param->fps +
                  (dvsend_param->current_time.ff));

/* requested frame */
request_frame = (((dvsend_param->request_time.hh * 3600) +
                  (dvsend_param->request_time.mm * 60) +
                  (dvsend_param->request_time.ss)) *
                   dvsend_param->fps +
                  (dvsend_param->request_time.ff));

/* difference between current frame and requested frame */
offset = (request_frame - current_frame) * 120000;

/* seek requested frame to reading file descriptor */
lseek(dvsend_param->file_fd, offset, SEEK_CUR);
```

図 6.8: フレーム制御処理

6.2.4 タイミング制御部

ART-Linux では，プログラムの実時間処理化，実時間制御を少数のシステムコールで実現している．また，プログラム中で一部の処理のリアルタイムタスク化，非リアルタイム化が自由に行える．図 6.9に実時間処理化，実時間制御，*fps*・*frame_drop* 変数を用いた再生レート変更処理の流れを示す．

タイミング制御部では，6.2.2節で述べた *fps* の値を用いることで，Periodic 制御の Dead Line 設定を行う．これによって，DV データを 1 フレーム毎に読み込み，送信を可能にする．また，*fps* の値を変更し，リアルタイムタスクの Dead Line を再設定することで，すべての DV フレームを，N 倍速で再生することが可能になる．ただし，この処理はクライアントが N 倍速での音声・映像デコード処理が可能な場合にのみ意味をなす．*fps* の値を変更せず，*frame_drop* の値を基に映像データのみを間引くことで，仮想的に N 倍速を実現することが可能である．

```
/* enter RealTime task */
art_enter(ART_PRIO_MAX,                /* Set Maximum Priority */
          ART_TASK_PERIODIC,           /* Task Type is Periodic */
          1000000/dvsend_param->fps);  /* Loop Cycle (micro sec) */

while(1){
  /******************/
  /* Read/Send part */
  /******************/

  if(check_request(request_sd)){
    if (request_is_timecode(recv_pkt)){
      seek_dvfile(recv_pkt);
    }
    else if(request_is_speed(recv_pkt) && framedrop_flag == 1) {
      frame_drop = play_speed;              /* frame dropped */
    }                                       /* N Speed Play  */
    else {
      art_exit();
      art_enter(ART_PRIO_MAX,               /* reconfig Realtime */
                ART_TASK_PERIODIC.          /* Scheduling        */
                (1000000/dvsend_param->fps) * play_speed);
    }
  }
  art_wait();  /* sleep untill Dead Line */
}
art_exit();    /* leave RealTime Task */
```

図 6.9: 実時間処理上での再生レート変更処理

6.3 受信部

受信部には,DVTS のクライアント (xdvshow・dvrecv for windows) に, タイムコードの参照, 再生制御要求処理を付加した．タイムコードの参照は送信部と同様の処理で行い, 再生制御要求処理では，ユーザからの再生制御要求をタイムコードに変換し，TCP を用いてサーバへ送信する．

6.4 まとめ:実装

本章では，設計段階で洗い出された要件を満たす開発環境を決定し，サーバ側の送信部とクライアント側の受信部の具体的な実装について述べた．

次章では，本システムが実現した True VoD システムの機能の評価を行う．また，4.1.3節で述べた，汎用マルチタスク OS と二次記憶装置を利用する際に発生する問題点を基に，本システムが向上させたスケーラビリティについての評価を行う．

第7章　評価

本章では，実装したシステムの評価を行う．評価項目は以下の通りである．

- 定性評価
  本システムが実現したIPネットワーク上における映像再生制御機能と，情報家電への移植性について評価を行う
- 定量評価
  本システムと，通常の汎用マルチタスクOS上でのVoDシステムと比較して，DVデータ入力・ネットワークへの送信に要する時間の短縮，および安定性についての評価を行う

7.1　定性評価:本システムが実現した機能

本節では，本システムが実現した映像制御機能と，情報家電への移植性について評価を行う．

7.1.1　映像の再生制御機能

本論文では，2.2節において，True VoD環境でユーザが頻繁に行う映像の再生制御はフレームの操作によって実現できることを述べた．本システムでは，True VoD環境において，DVフォーマットとタイムコードを連携させ，映像ストリーミング再生中のフレーム制御を実現した．満たした機能は以下の通りである．

- 任意のDVフレームの特定と読み込み・送信
- 任意のDVデータ送信レート選択

これらの機能を組み合わせることによって，あらゆるフレーム制御を基にした再生制御が可能となる．また，これらの機能は，フレーム内圧縮を用い，タイムコードを記述フィールドのある映像フォーマットであれば，DVフォーマット以外にも適用することができる．

7.1.2　情報家電への移植性

本システムは、Linuxをベースとしたリアルタイム OSであるART-Linux上で実装を行った．前述の通り、Linuxは6.1.3節で述べた通り、汎用的な組み込みOSを備える情報家電上において広範囲に適用されている．デジタルAV機器の中では，ハードディスクレコーダなどにおいてもLinuxを採用している事例が多く、本システムの情報家電への移植性は高い．

7.2 定量評価:実時間処理の評価

本節では定量評価として，ART-Linux を用いた実時間処理上と通常の Linux Kernel 上の処理に要する時間の計測を行った．具体的には，SDDV フォーマット NTSC 方式の 1 フレーム分のデータである 120,000byte の読み込み時間と，ネットワークインターフェースからのパケット送出時間の揺らぎを計測した．

7.2.1 評価環境

計測を行った計算機のハードウェア環境を表 7.1に示す．

表 7.1: ハードウェア環境

Processor	Pentium III 1GHz, Cache 256KByte
Memory	SDRAM PC133, 256MByte
Network Interface	Intel PRO/1000 MT Desktop Adapter
Storage Device	UltraDMA ATA66, 7200RPM, 40.9GByte Cache 2MByte, Avarage seek time 0,9msec

Intel Pro/1000 MT Desktop Adapter については，64Bit PCI-BUS に対応しているが，実際の使用時は 32Bit PCI-BUS に接続して使用した．

計測に利用したタスクはすべて DV データを 1 フレーム読み込み，送信を繰り返すプログラムで，リアルタイムタスクでは Dead Line を 1/29.97 秒に設定し，通常タスクでは select システムコールを用いて，擬似的に Dead Line 処理を追加したものである．また，計測を行う際に，同一の DV ファイルを複数のタスクを用いた場合，二次記憶装置のキャッシュが大きく関係してくるため，複数のタスクを用いた場合，異なる DV ファイルを読み込みに使用した．なお，時間算出には Pentium Counter を用いた．

7.2.2 DV フレーム読み込み時間の揺らぎ

本節では，DV フォーマット NTSC 方式の映像 1 フレームを読み込むのに要する時間の揺らぎを計測した結果から，汎用マルチタスク OS を用いた場合と，ART-Linux を用いた場合の差異を示す．

計測に用いたタスクの種類は以下の通りである．

1. 優先度最高リアルタイムタスク
   図 7.1，7.2，7.3中 rt * 1
2. 優先度最高リアルタイムタイム + 優先度最高リアルタイムタスク
   図 7.1，7.2，7.3中 rt * 2
3. 非リアルタイムタスク
   図 7.1，7.2，7.3中 no rt * 1
4. 非リアルタイムタスク + 非リアルタイムタスク
   図 7.1，7.2，7.3中 no rt * 2

図 7.1は第 1 項，第 2 項のデータをグラフに示している．図 7.2は第 3 項，第 4 項のデータをグラフに示している．図 7.3は，第 1 項，第 3 項のデータをグラフに示している．7.2で第 3 項のと第 4 項の値が掛け離れ差分がわかりづらいため，参考のために示した．

図 7.1からは，複数の DV データ読み込みタスクが存在した場合でも，互いのタスクが協調してデータの読み込みに必要な時間の揺らぎが小さく，安定したデータ入力を実現していることが読み取れる．

第 1 項，第 2 項のリアルタイムタスクは，DV データ読み込みの開始から終了まで可能な限り計算機資源を占有し，他のタスクの影響を最小限にとどめるよう，タスクスイッチングがなされている．また，それは複数のタスクが立ち上がった場合でもほとんど差がない．

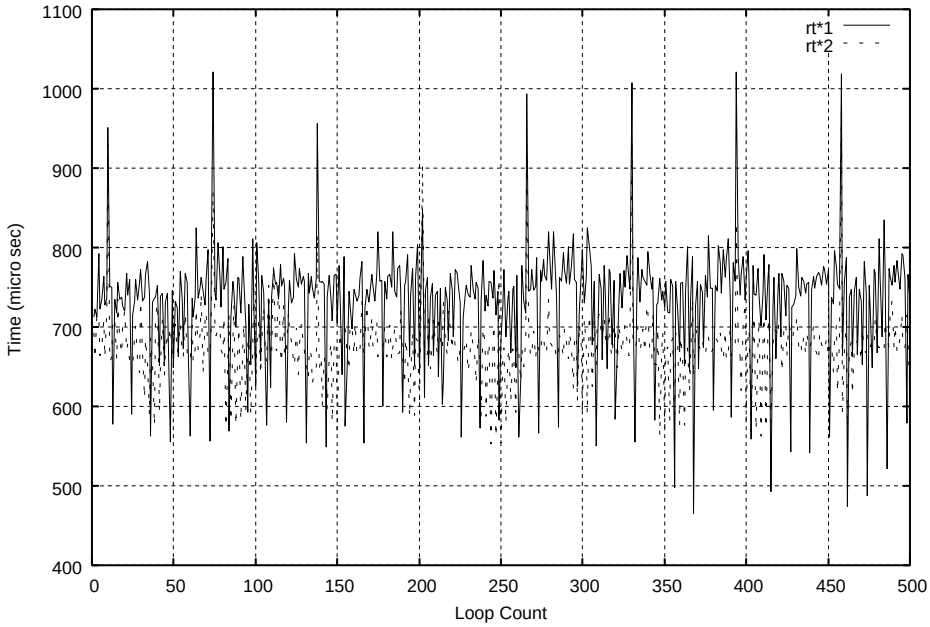

図 7.1: read に要する時間#1

図 7.2からは，複数の非リアルタイムタスクを立ち上げた場合，個々のタスクはタイムシェアリングシステムで管理されるため，頻繁にタスクスイッチングを繰り返し，データ読み込みに必要な時間がリアルタイムタスクと比較して長時間で，大きく揺らいでいることが読み取れる．

NTSC 方式の場合、1 フレームの表示時間は 33,366 マイクロ秒であるため、第 4 項の非リアルタイムタスクは，フレームの入れ替えと言う Dead Line を DV データ読み込みの時点で大幅に越えてしまう場合が発生している．また、DV データ読み込み時間が大きく揺らいでいるため、他のタスクと協調し、計算機資源を有効に活用できていないことを示している．第 3 項は，グラフ中の表示が読み取れないほど第 4 項と比較して DV データ読み込みに必要な時間が短い．そのため，図 7.3で第 3 項を参照する．

図 7.3からは、他に多くの計算機資源を必要とするタスクが無い場合は、一部のはずれ値を覗いて非リアルタイムタスクとリアルタイムタスクの DV データ読み込み時間に差が無いことを示している．

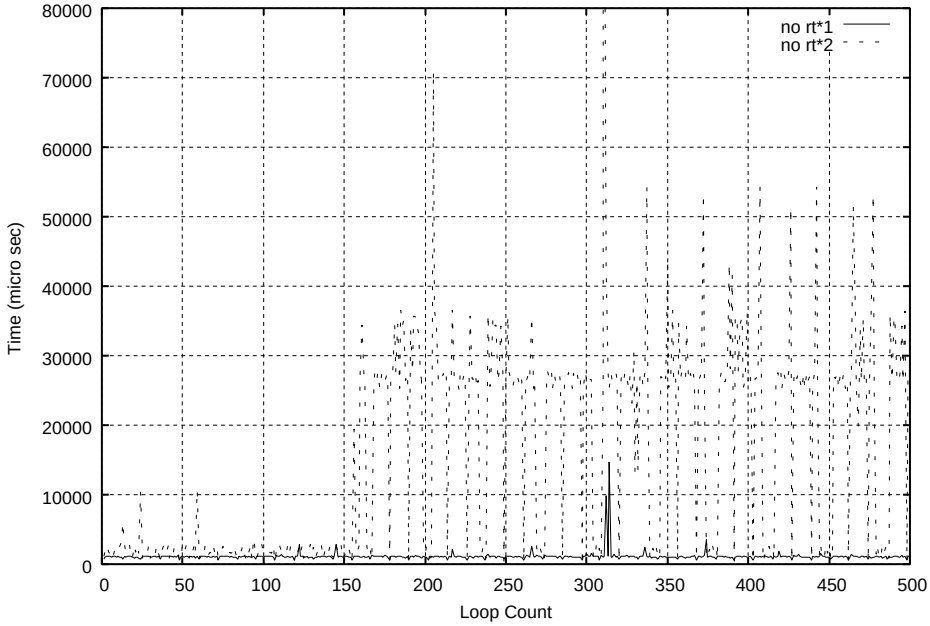

図 7.2: read に要する時間#2

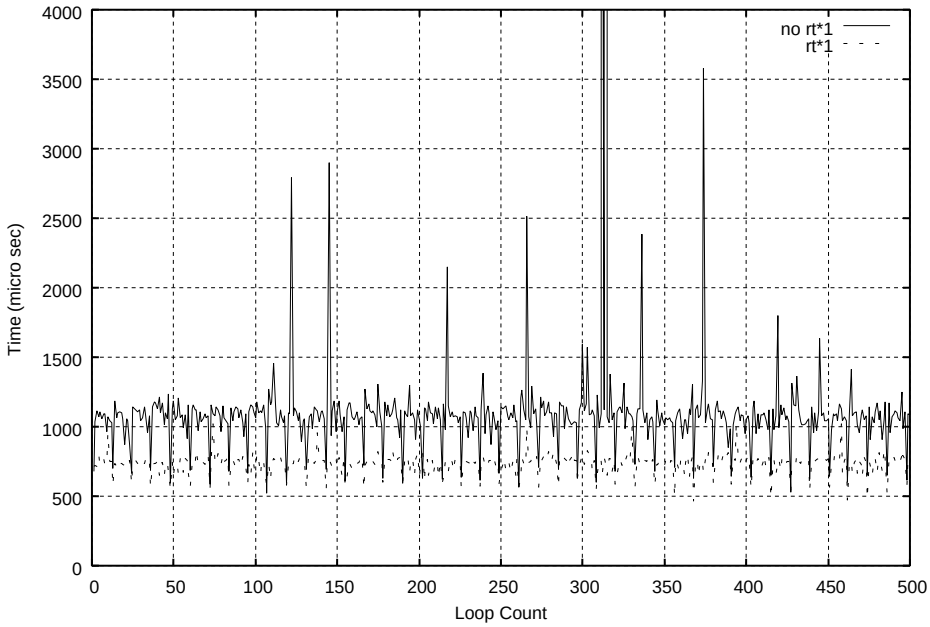

図 7.3: read に要する時間#3

7.2.3 DV フレーム送信時間の揺らぎ

本節では，メモリ上にマップされた DV フォーマット NTSC 方式の映像 1 フレームを，Network Interface から送出するまでに要する時間の揺らぎを計測した結果から，通常の Linux Kernel を用いた場合と，ART-Linux のリアルタイムタスクを用いた場合の差異を示す.

計測に用いたタスクは以下の通りである.

1. 優先度最高のリアルタイムタスク + 優先度最高のリアルタイムタスク
   図 7.4中 prio max * 2

2. 非リアルタイムタスク + 非リアルタイムタスク
   図 7.4中 non real time * 2

図 7.4は，上記の項目から得たデータをグラフに示している.

第 1 項は，リアルタイムタスクの利用によって，DV データ送信中にネットワークインターフェースの資源を最大限に活用していることがわかる．これは，送信処理の開始から終了までに他のタスクからの影響を最小限に押さえ，優先度最高のタスクが最小限のタスクスイッチングで送信処理を終えていることと，タスクスイッチングの回数が毎回一定であることを意味している．そのため，読み込んだ DV データの送信処理終了までの時間の揺らぎはほとんどない.

第 2 項は，ネットワークインターフェースの送信キューに対して，読み込んだ DV データを書き込んでいる最中にタスクスイッチングが発生した場合，データ送信処理終了までに長時間要していることがわかる．逆に，他のタスクからの割り込みによってタスクスイッチングが行われない場合は，第 1 項と同様に短時間で送信処理が終了していることがわかる．そのため，読み込んだ DV データの送信処理終了までの時間が大きく揺らいでいる.

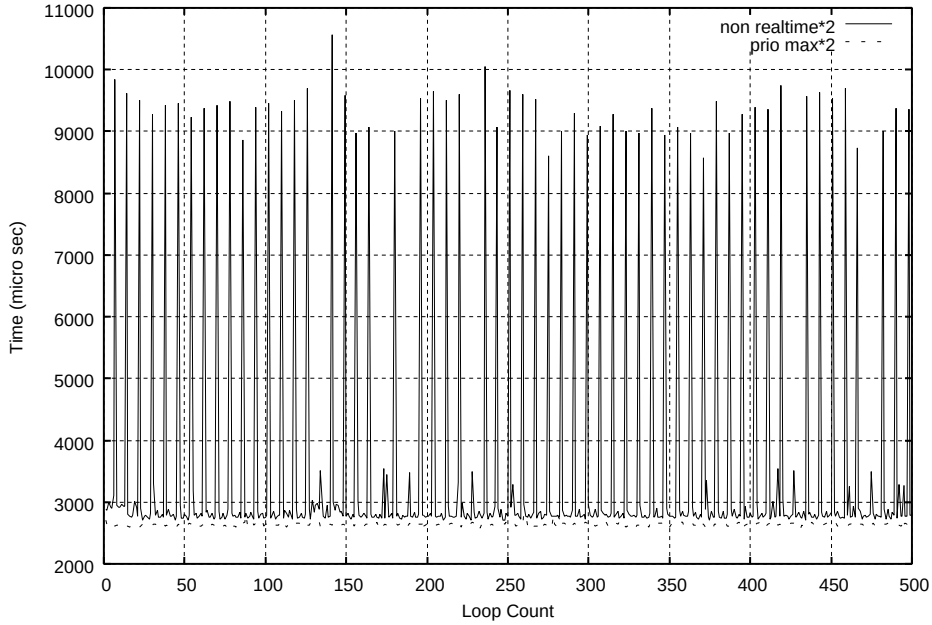

図 7.4: sendto に要する時間

7.2.4 定量評価のまとめ

7.2.2節，および 7.2.3節より，優先度付き Periodic 制御を用いることで，DV データの読み込み・送信に要する時間の平滑化が実現されているのがわかる．これによって，個々のタスクが協調し互いの Dead Line を越えることなくデータ送信を可能にしている．そのため，クライア

ントで視聴する映像は乱れることなく再生でき，タスク数が増加してもハードウェアの性能の限界近くまでパフォーマンスを引き出せる．そのため，ハードウェアの高性能化と共に対応できるクライアントの数も最大限に増加させることが可能である．

第8章　結論

本章では，本研究の成果をまとめ，今後の展望を述べる．

8.1　まとめ

本研究は，高品質映像フォーマットとしてSDDVを用いたTrue VoDシステムの設計と実装を行った．同時に，リアルタイムOSを用いることで，汎用マルチタスクOS上での実装よりも計算機資源を有効活用可能なVoDシステムを構築した．

現在主流なTrue VoDに近い映像配信形態であるWMTは，低帯域なデータリンクを対象としている．WMTは，フレーム間圧縮フォーマットであるWMVとTCPを用いるため，正確な映像制御ができない問題を抱えている．また，広帯域データリンクを想定し，SDDVフォーマットを用いたQuasi VoDの映像配信形態であるDVBSは，大容量映像データであるSDDVを用いているため，二次記憶装置，ネットワークインターフェースに大きな負荷をかけ，多数のクライアントに映像配信することができない問題を抱えている．これらのVoDシステムでは，映像配信形態に適した映像フォーマットと映像転送技術の組合わせを実現することができず，今後の映像配信サービスに要求される，高品質な映像フォーマットを用いた時間粒度の細かい再生制御が実現できなかった．

これらの問題に対して，本システムは，現在の映像配信サービスに対する要求である，高品質映像の配信，時間粒度の細かい映像の操作性，映像の操作要求から反映までの即時性を，SDDV，タイムコード，UDP/RTPを使用することで，クライアントの要求に適した映像フォーマットとトランスポートプロトコルを用いて実現した．また，SDDVを用いた細かな映像操作性を実現することで発生する，ストリーミングセッションの増加に対して，複数のストリーム生成タスクが存在する場合でも，実時間処理を用いることで計算機資源を有効活用し，映像データの入力・送信処理の安定化を実現した．その結果，一定資源当りで対応できるクライアント数を増加する．

また、本システムはリアルタイムOSにLinuxをベースとしたART-Linuxを採用したため、情報家電への高い移植性を実現した．

8.2　今後の展望

本研究の今後の展望として，以下の事項を挙げる．

- より安定したデータ読み込み速度の二次記憶装置の利用と評価
  今回の実装ではIDE接続のハードディスクを用いたため，セクタによる読み込み速度の差が大きく，データ読み込み時間の揺らぎが大きかった．SCSI接続や，シリアルATA接続のハードディスク，RAID0・RAID5で構成された大規模ストレージを用いることで

データ読み込み速度の向上と安定化を実現できる．これらのデバイスを積極的に使用し，本システムがより安定して動作する環境の検証を行う．

- DVTS への応用
  リアルタイム OS には，今回利用したタスク管理方法以外にも有効なタスク管理機能を持っている．IEEE1394 をデータ入力デバイスとして活用する DVTS では，アイソクロナス転送によってデータ入力周期が一定なため，優先度付き Round Robin 制御を用いて，複数の DVTS プロセスを起動した際の安定した処理を実現できる．

- 情報家電へのフィードバック
  今回の実装で実時間処理を採用することで，VoD システムのパフォーマンスが向上することを示した．比較的計算機処理能力の低い情報家電上に本研究の成果を適用し，高画質，かつ高精度な映像制御を可能にする VoD システムを情報家電の分野にも普及させる取り組みを行う．

謝辞

本研究を進めるにあたり，ご指導をいただきました，慶應義塾大学環境情報学部教授である村井純博士，徳田英幸博士，同学部助教授の中村修博士，楠元博之博士, 同大学環境情報学部専任講師の南政樹氏, 重近範行博士に感謝致します.

絶えずご指導とご助言を頂きました独立行政法人通信総合研究所の杉浦一徳博士, 慶應義塾大学院政策・メディア研究科の入野仁志氏，同 SFC 研究所の小川浩司氏に感謝致します.

毎日の大学の往復など，公私において大変お世話になった工藤紀篤氏に感謝致します．すばらしい OSXserver の絵と，卒論中に彼女をつくろうとして論文執筆中に勇気をくれた久松剛氏に感謝致します．一緒に窓際プログラミングをしてくれた三島和宏氏に感謝致します．また同じ卒論生として一緒にやってきた rg-00 の仲間に感謝します.

忙しい時期に，細かな作業に手を貸してくれた小椋康平氏，千代佑氏，松園和久氏をはじめとした後輩達に感謝致します．そして日々の活動の場として，遊びも研究も常に本気な STREAM KG，並びに DVTS Project の皆様に感謝致します．また，DVTS の共同開発において映像分野の専門的な指摘を頂いた三好学氏をはじめとした日本ビクター開発メンバーの皆様に感謝致します.

参考文献

[1] HD DIGITAL VCR CONFERENCE. Specifications of Consumer-Use Digital VCRs PART1 General Specifications of Consumer-Use Digital VCRs. *Specifications of Consumer-Use Digital VCRs using 6.3mm magnetic tape*, pages 1–61, Dec 1994.

[2] MPEG-2 Generic coding of moving pictures and associated audio information. *http://mpeg.telecomitalialab.com/standards/mpeg-2/mpeg-2.htm*, Oct 2000.

[3] Akmichi Ogawa. DVTS(Digital Video Transport Sysytem). *http://www.sfc.wide.ad.jp/DVTS/*.

[4] 入野 仁志. MPEG2-TS over IP の設計と実装. **卒業論文**, Feb 2003.

[5] デジタルシネマコンソーシアム. *http://dcc.imgl.sfc.keio.ac.jp*.

[6] DVTS コンソーシアム. *http://www.dvts.jp*.

[7] Advanced Television Systems Committee. *Guide To DTV Standards*, 2002.

[8] Microsoft. *http://www.microsoft.com*.

[9] RealNetworks. *http://www.realnetworks.com*.

[10] MPEG-4 Industry Forum. *MPEG-4 The Media Standard*, July 2002.

[11] Information technology -Generic coding of moving picture and associated audio information : Systems. *ISO/IEC 13818-1:2000*, 2000.

[12] Harindra Rajapakshe. Interactivity in Public Media. May 1995.

[13] J. Postel. RFC793 Transmission Control Protocol. *http://www.ietf.org/rfc/rfc0793.txt*, pages 1–85, September 1981.

[14] J. Postel. RFC768 User Datagram Protocol. *http://www.ietf.org/rfc/rfc0768.txt*, pages 1–3, August 1980.

[15] Audio-Video Transport Working Group. RFC1889 RTP: A Transport Protocol for Real-Time Applications. *http://www.ietf.org/rfc/rfc1889.txt*, pages 1–75, January 1996.

[16] 菅沢　延彦. IP マルチキャストを用いた放送型 VoD 機構の実現. **修士論文**, 2002.

[17] Microsoft Developer Network. *http://msdn.microsoft.com*.

[18] M. Handley and V. Jacobson. *SDP: Session Description Protocol*, April 1998. RFC 2327.

[19] M. Handley, C. Perkins, and E. Whelan. *Session Announcement Protocol*, October 2000. RFC 2974.

[20] Institute of Electrical and Electronics Engineeers, Inc. IEEE Std 1394-1995 High Performance Serial Bus. Aug 1996.

[21] FSMLabs RTLinux. *http://www.fsmlabs.com/*.

[22] ムービングアイ ART-Linux. *http://www.movingeye.co.jp/*.