修士論文　　2002 年度 (平成 14 年度)

自律メディア同期機構の実現

慶應義塾大学 政策・メディア研究科

氏名：小川 浩司

修士論文要旨 2002 年度 (平成 14 年度)

自律メディア同期機構の実現

本研究では，複数のメディアを自由に組み合わせて配信を行なう環境の構築を目指し，メディアの形式，配信方法に依存せずに，動的にメディア間の同期関係が定義できるメディア間同期モデルの提案を行なった．

実時間配信におけるメディア間同期関係の動的な定義には，コンテンツを構成する多様なメディアに対して，同期処理の基準となる同期点を定義し，試聴状況のフィードバックを行なう必要がある．

本研究が提案するメディア間同期モデルは，1) メディアの抽象化，2) 自律的な時間軸の管理，3) 同期情報の記述の 3 つの特徴を有する．

本モデルでは，個々のメディアをメディアオブジェクトとして抽象化し，メディアの配信処理，表示処理とメディア間同期処理を切り分けることで，メディアの多様化に対応可能とする．

メディア間同期処理は，基準時間軸上に指定される表示時刻を基にメディアオブジェクトを操作することで行なう．基準時間軸は，メディアの再生状況に応じて自律的に管理することで，受信環境の違いによる処理時間の差異を吸収する．

また，メディアオブジェクトの定義と基準時間軸に基づき，メディアオブジェクトの操作指示，試聴状況のフィードバック情報の記述方法を定義した．同期情報を記述することで，動的な時間関係の定義を行なう手法を示す．

本研究の提案する設計に基づいて，講義中継を対象とした自律メディア同期機構 SMP（Synchronized Media Point）の実装を行なった．メディアオブジェクト抽象化の実現方法として，C++言語の継承クラスによる実装方法を示す．また，COM 技術を用いて既存のアプリケーションの利用を行なうことにより，多様なメディアへの拡張性を確保した．SMP の実装では，Windows Media Video と PowerPoint スライドショーのメディア間同期を実現した．

SMP を用いて，本研究が提案するメディア間同期モデルについて評価を行なった．評価結果より，SMP は 100$msec$ でのメディア間同期を実現可能であることがわかった．

本機構は，メディアの形式，配信方法に関わらず，動的にメディア間の時間関係を定義し，定義された時間関係にしたがった同期を実現する．また，フィードバック情報によって，送信者による複数の受信環境における試聴状況の把握を可能とする．

以上より，本論文で提案したモデルは，あらゆるメディアを自由に組み合わせて配信を行なえる環境を実現できることを示した．

キーワード
1．実時間配信，2．同期，3. 複合メディア，4. インターネット

慶應義塾大学大学院 政策・メディア研究科

小川 浩司

Abstract of Master's Thesis

Academic Year 2002

Implementation of autonomy inter-media synchronizing system

For the purpose of establishing the environment where multiple media are flexibly combined and distributed on the Internet, this paper proposes the inter-media synchronizing model which defines the time relations of media independently from the way of distribution.

To define the time relationship dynamically, it is necessary to fix the synchronizing point of multiple media composing one content. It is also important to get feedback information from each receiver to check the playing status. This proposed model has the following 3 characteristics; 1) abstraction of media, 2) autonomous timeline management, 3) description of synchronizing information

This model is flexible to diversification of media, because each media is abstracted to the media object and synchronization process is distinguished from distribution process and playing process.

The inter-media synchronization process is performed by controlling the media objects based on the playing time defined in the base timelines. The base timelines are managed autonomously by each receiver so that the inequality of process time has no influence to the whole system.

It is also defined in this paper how to describe the operations for media object and the feedback information. By describing all the information required for synchronization, this paper shows how to manage the dynamic time relations of each media.

Based on the proposed model, the autonomous synchronizing system for the real-time lectures, SMP(Synchronized Media Point), has been implemented. As the way of media abstraction, the implementation with C++ inheritance class was adopted. The COM technology was also used so that this system can be applicable to any existing applications. SMP achieved the synchronization between the PowerPoint slide show and the Windows Media Video.

We evaluated the proposed model based on SMP and found that SMP can achieve the synchronization with the accuracy required for the lectures.

This result indicated that this model can achieve the environment for distribution with flexible combination of various media.

Keywords :
1. realtime-distiribution, 2. synchronization, 3. multimedia, 4. Internet,

Keio University Graduate School of Media and Governance

Koji Ogawa

目 次

図 目 次

表 目 次

第1章　序章

本章では本研究の背景として，複数のメディアから構成されるコンテンツの配信の現状について検討を行ない，本研究の目的について述べる．

1.1　複数のメディアから構成されるコンテンツの増加

インターネットを用いたコミュニケーションにおいて，利用されるメディアの多様化が起こっている．従来は，インターネットを用いて配信されるコンテンツは文字列や静止画像が中心であった．しかし，近年では動画や音声，或いは複数のメディアを組み合わせたコンテンツが多く見られる．

これらの背景として以下の 2 点が挙げられる．

- インターネットの広帯域化とユーザ層の拡大
- 計算機環境及び技術の向上

動画情報や音声情報は，文字情報や静止画像情報と比較して多くの帯域を必要とする．インターネットバックボーンは WDM（Wavelength Division Multiplex），Ethernet 等の技術の向上により 10$Gbps$ 帯域の利用が可能となった．

家庭環境では，CATV（Cable TeleVision），xDSL（x Digital Subscriber Line）や FTTH（Fiber To The Home）といった基盤技術の普及に伴い，10$Mbps$ 以上の帯域が利用可能となり，また常時接続サービスの開始によって，家庭環境において動画や音声コンテンツの閲覧に必要となるネットワーク環境が整いつつある．

一般に動画情報や音声情報は情報量が多いため圧縮して配信する．圧縮形式に依存するものの，情報の圧縮及び解凍処理には多くの計算機資源を必要とする．また，動画や音声の再生処理は連続的に行なわれる必要があるため実時間処理が要求されるが，圧縮技術・ストリーミング技術，CPU（Central Processing Unit）の性能等の向上によってインターネットを用いた動画・音声の配信が可能となった．

上記の技術的背景に伴い，複数のメディアから構成されるコンテンツの配信に対する要求が増加している．HTML（Hyper Text Markup Language）[1] は，複数のメディアを組み合わせたコンテンツの制作手段の 1 つである．HTML を用いることにより，文字情報や画像情報のみならず，動画情報や音声情報を埋め込んだ HTML コンテンツ制作が可能となる．

複数のメディアから構成されるコンテンツ配信の応用例として遠隔講義が挙げられる．近年，多くの大学・企業で遠隔教育環境の利用あるいは構築に関する取り組みが行なわれている．

著者が参加している WIDE Project School of Internet Working Group（SOI）[2] は，インターネット基盤を利用した大学環境の構築を行なうプロジェクトであり，実際に大学で行なわれている講義を収録し，講義映像及び講義資料を蓄積して配信するとともに，リアルタイム受講環境の構築を行なっている．リアルタイム受講環境では，講義映像及び講義資料の他に遠隔受講者からの質問

をはじめとする文字情報も同時に配信される．使用される講義資料も多様であり，講義は講師の話と講師が使用する資料という複数のメディアから構成されるコンテンツであると言える．

さらに，講義資料や質問等の情報は，講義の該当箇所で適切に表示されることが重要となることから，講義はコンテンツを構成するメディアに，表示を行なう時刻という時間的関係が定義されたコンテンツと言える．

このような複数メディアの利用は，コンテンツにおける表現の幅を大きく広げ，遠隔講義のみならず，さまざまな分野で用いられている．

1.2 複数のメディアから構成されるコンテンツ配信の現状

前述したように，複数のメディアを利用したコンテンツには時間的関係の定義が必要となるものが多くあるが，HTML を用いてメディアの時間的関係を定義するのは困難である．このような要求に対して，SMIL（Synchronized Multimedia Integration Language）[3] のような複数のメディアを用いた表現を行なうための技術も提案され，注目されている．SMIL を用いることでメディアの時間的関係を記述できる．また，Macromedia 社の Flash[4] のようなソフトウェアを用いることで，メディアの時間的関係を定義したコンテンツを作製できる．Flash で作製したコンテンツはプラグインによって Web ブラウザ上で表示できる．

このようにメディアの時間的関係を静的に定義できる技術或いは環境は多い．しかし，動的にメディアの時間的関係を定義できる環境は少ない．

動的な時間的関係の定義は，前述の遠隔講義環境のような実時間配信を行なう場合に必要となる．講義では，講師の操作によって資料の表示が行なわれるため，静的な時間関係の定義はできない．

動的なメディア間の時間的関係の定義を可能とする既存の配信機構の多くでは，利用できるメディアが制限されてしまうことが多い．その主な要因としては，システムのメディアの形式あるいは伝送方法への非対応にある．

さらに，これまであまり重要性が認識されてこなかったが，実時間配信における各受信者の試聴確認は，円滑な双方向コミュニケーションを実現するために必要不可欠である．送信側は，遠隔地の受講者が時間的関係にしたがって情報を受けとっていることを確認しながら進行できなければ，情報を充分に伝えられない．送信側で受信者の試聴状態を確認し，待機あるいはそのまま進めるといった判断が可能な環境の構築が必要である．

以上のように複数メディアを使用したコンテンツの配信環境には未だ多くの課題がある．

1.3 研究の目的

本研究では，多様なメディアを組み合わせたコンテンツの柔軟な配信環境の構築を目指し，複数メディアから構成されるコンテンツの配信において，動的なメディア間の時間関係の定義及び送信側による試聴状況の確認方法のモデル化を行なう．さらに，本研究で提案するモデルに基づき，メディアの種類及び配信方法に依存しない実時間配信機構の実現を行なう．

1.4 本論文の構成

本論文は 7 章で構成される．第 2章では本研究が対象とするメディア及び複数のメディアから構成される複合メディアの定義を行なう．複合メディアの特性及び配信手法について検討し，複数メ

ディアから構成されるコンテンツの表示における同期処理の必要性について述べる．第 3章では，メディア配信において必要となる同期処理について分析し，同期処理に影響を与える要素の検討を行なう．また，同期処理において必要となる同期点について述べる．第 4章では本研究の要求事項を明らかにし，動的な時間関係の定義を可能とするメディア配信機構モデルの提案を行う．第 5章で本モデルに基づいて行なったプロトタイプの実装について述べ，第 6章で本実装に関する評価を行なう．最後に結論として本研究のまとめと今後の課題について述べる．

第2章　複合メディア

本章では，本研究が対象とするメディア及び複合メディアの定義を行なう．また，複合メディアの配信について述べ，伝送体系，配信形態の検討を行なう．

2.1　メディア

メディア（媒体）は．コミュニケーションの際に情報を伝えるための媒体・手段を意味する．本研究では，人間の知覚で認知される情報をデジタル化したものをメディアと定義し，本研究の対象とする．本研究の対象とするメディアの種類を表 2.1に示す．

メディアは，符号・圧縮処理によりアナログ情報からデジタル情報に変換され，復号・解凍処理によって表示される．メディアのデジタル情報形式は表 2.1に示すように多岐にわたる．

表 2.1: メディアの種類

種類	形式
文字情報	ASCII，UTF8，JIS などの形式で表される
画像情報	BMP，JPEG，PNG などの形式で表される
音声情報	PCM，WAV，MP3 などの形式で表される
動画情報	MPEG4，H.263 などの形式で表される

メディアは，時間に依存する連続性メディアと依存しない非連続性メディアに分類できる．時間に依存するメディアは時間的連続性のあるメディアであり，動画情報や音声情報が挙げられる．動画の場合，情報はフレーム単位の静止画像の集合であり，形式によってフレームの表示される間隔が規定されている．

連続性メディアの表示に不連続性が生じると人間の感覚にとって不自然なものとなり，内容理解の妨げとなる．しかし時間に依存するメディアは，多少の情報の欠落や不連続性が生じても視聴者が前後の関係から内容を推測できるので，全体の流れを把握できるという性質がある [5]．

時間に依存しないメディアには文字情報や画像情報がある．これらのメディアは時間軸と関係なく表示できる．

メディアの表示と時間軸の関係を図 2.1に示す．図 2.1より，時間軸に対して連続メディアは線，非連続メディアは点と考えられる．また，非連続メディアは再生時間が限りなく 0 に近い連続メディアであるとも言える．

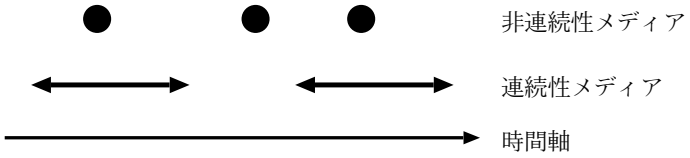

図 2.1: 時間軸とメディアの表示

2.2 メディアの配信

インターネットを用いたメディアの配信処理の過程を図 2.2に示す．本研究の対象とするメディアは全てこの過程を経て配信されると考える．

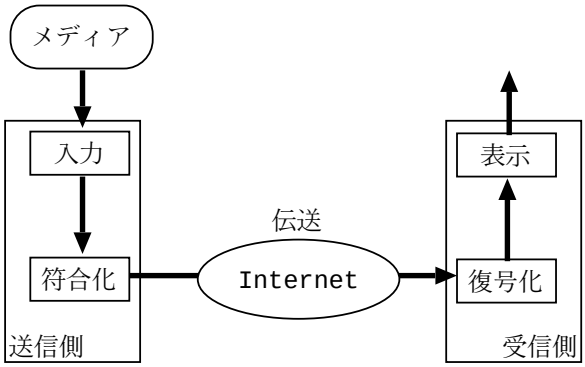

図 2.2: インターネットを用いたメディア配信

図 2.2の処理は以下の 5 つの行程に分類される．

1. 情報の入力
   キーボードやキャプチャーボードのような入力デバイスから文字情報，動画情報が入力される．
2. 符号化処理
   情報を配信するための形式に符号化・圧縮される．
3. 情報の伝送
   インターネットを介して情報が伝送される．
4. 復号化処理
   情報を表示するための形式に復号化・解凍される．
5. 表示処理
   情報を出力用デバイスに渡すことで，表示を行なう．

2.2.1 実時間配信と蓄積型配信

メディアの配信形態には実時間配信と蓄積型配信がある．

蓄積型配信では，情報は符号化された時点でサーバなどのストレージに蓄積される．受信者はストレージから情報を受信し試聴する．web サーバを介した HTML の閲覧は蓄積型配信にあたる．

実時間配信は入力された情報を蓄積することなく，受信者に配信する．Real System[6] の Broadcast 機能等が実時間配信にあたる．連続性メディアを扱うシステムの多くが実時間配信を実現している．

蓄積型配信では情報をストレージに蓄積するまでの処理と受信者の試聴までの処理を，時間的にあるいはシステムとして分けてとらえることができるが，実時間配信では送信側の入力から受信側での表示まで一連のシステムとして機能する必要がある．

2.2.2 中継点を介する伝送

既存のメディア配信システムには図 2.3に示す中継点を介するモデルと，図 2.4に示す中継点を介さないモデルがある．配信サーバやキャッシュサーバ，ファイアウォール等の機能を持ったノードが中継点として挙げられる．

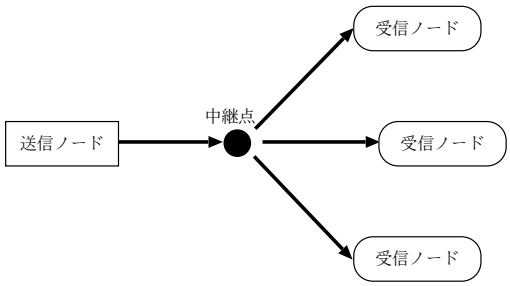

図 2.3: 中継点を介する配信

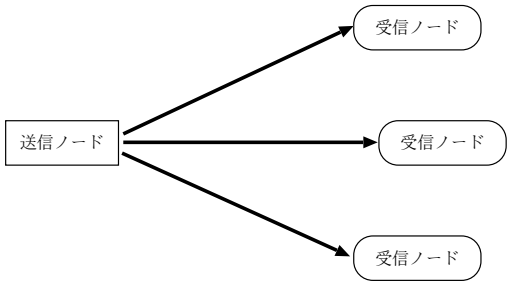

図 2.4: 中継点を介さない配信

配信サーバやキャッシュサーバを中継点として設置し，中継点を介した配信を行なうことにより，配信処理の分散化が可能となり，サービスに規模性を持たせることができる．しかし，送信ノードから中継点への情報伝送及び中継点における処理に要する時間によって，配信の実時間性は下がる場合が多い．

マルチキャスト経路制御プロトコル PIM（Protocol Independent Multicast）[8] のランデブーポイントも配信における中継点と言える．ランデブーポイントは，最適なマルチキャスト経路木を構築するために定められる．

2.3 メディアの伝送とプロトコル体系

インターネットで利用される代表的な伝送プロトコルにはTCP（Transmission Control Protocol）[10] と UDP （User Datagram Protocol）[9] がある．メディアはその特性，あるいは配信サービスの要求に応じたプロトコルで伝送される．

TCP はコネクション型の接続で，送信側と受信側の間に仮想回線を設定し情報の伝送を行なう．TCP は以下の 3 つの特徴を有する．

- パケットの再送処理
  伝送の途中でパケットの損失が起きた場合，失われた情報の再送が行なわれる．
- パケットの到着順序の保証
  受信側にパケットが到着する間に，パケットの順序が入れ替わった場合は，受信側で送信時の順序に並べ替えられる．
- パケットの輻輳制御
  受信側の受信能力とネットワークの状態に応じてフロー制御を行ない，回線の輻輳を回避する．

UDP はコネクションレス型の接続で，情報はデータグラムに分割されて伝送される．UDP にはパケット再送などの機能を備えていないため，情報の到達性は保証されない．

動画や音声のような連続性メディアは UDP で伝送され，文字情報などの非連続メディアは TCP で伝送されるのが一般的である．

時間に依存する連続性メディアが UDP で伝送される理由として 4 つが挙げられる．

- TCP によるパケットの再送処理や輻輳制御はいずれもパケットの到着時間に影響を与え，時間に依存した表示が困難となる．
- 2.1節で述べた連続性メディアの性質から，多少の情報欠落が許容できる．
- パケットの到着順序の保証は受信側のアプリケーションによる対応が可能である．到着順序を保証する手法の 1 つとして RTP（Realtime Transport Protocol）[11] の利用が挙げられる．
- 1 対多のコミュニケーションを実現する方法の 1 つである Multicast は，TCP よりも UDP が適している [12].

Windows Media Technology や Real System 等の既存のアプリケーションには連続性メディアを TCP を用いて伝送するものもある．TCP を用いて伝送を行なう場合，アプリケーションが伝送するネットワークの使用可能帯域，パケット損失率といった情報を検知・予測し，これらの情報を基にメディアを表示するために必要なバッファリング量を計算することで連続性を確保する．TCP を用いる場合，欠落のないメディア配信を行なえるが，再送や輻輳制御に応じたバッファリング処理が必要となるため，実時間性が低下する．

2.4 複合メディア

複合メディアとは，複数のメディアの集合である．図 2.5に示すように，複合メディアの構成要素に複合メディアを使用することもできる．例えば，映画というコンテンツは動画，音声及び文字情報（字幕情報）の 3 つのメディアから構成される複合メディアである．

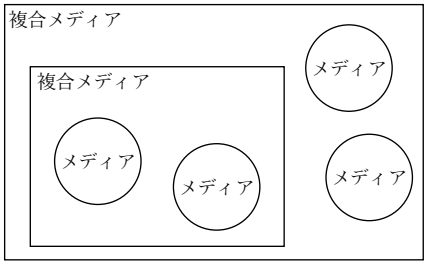

図 2.5: 複合メディア

複合メディアでは，メディア間に時間的あるいは空間的な相対関係が定義されており，複合メディアを表示する際にはメディア間に定義された関係にしたがって表示を行なう必要がある．

2.4.1 メディアの持つ相互関係

メディア間の相互関係が定義されていない，あるいは定義された関係にしたがって表示されない複数メディアは複合メディアではない．

複合メディアにおけるメディア間の相互関係には時間的関係と空間的関係がある．

時間的関係

メディア間に定められる時間的関係とは，メディア間における時間への依存である．例えば，動画と音声からなるメディアコンテンツでは，動画と音声の時間的関係を考慮して再生されなければ，動画の内容と音声があっていないといった問題が起こり，コンテンツとして成立しない．

時間に依存するメディアは個々の時間への依存性をくずすことなく，メディア間の時間的関係に従う必要がある．

空間的関係

メディア間に定められる空間的関係とはメディアのレイアウトである．メディアコンテンツは，見る側にとって理解しやすいレイアウトで提供されなければならない．例えば映画では字幕情報が表示されることが多いが，字幕情報は動画を見る際に妨げにならない場所でかつ読みやすい場所に配置される必要がある．

2.4.2 複合メディアの配信

複合メディアのインターネットを用いた配信形態には，単一ソースとしての配信と複数ソースとしての配信の 2 種類がある．

単一ソースとしての配信

単一ソースとしての配信を図 2.6に示す．

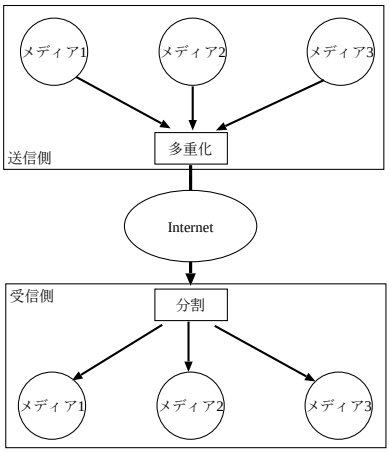

図 2.6: 単一ソースとしての配信

単一ソースとしての配信では，複合メディアを構成する複数のメディアは 1 つのソースとして配信を行なう．受信側ではソースをメディア毎に分解して表示する．

文字情報と動画情報からなる複合メディアを配信する場合，単一ソースとしての配信方法として以下の 2 つが挙げられる．

1. 情報の合成
   文字情報を画像情報に変換し，動画情報のフレームと合成して配信する．

2. 情報の多重化
   1 つのデータストリームに動画情報のパケットと文字情報のパケットを相互に送信する．

個々のメディアは送信側で合成・多重化されるため，単一ソースとしての配信では，メディアの入力あるいは符号化の際にメディア間の相互関係は満たされる必要がある．

情報の合成の例として映画の字幕情報が挙げられる．映画では字幕情報は動画情報に埋め込まれて，1 本のフィルムとして上映される．この方法でメディア間の同期関係は維持できる．しかし，計算機上では画像情報として扱われ，文字情報として扱えないという欠点がある．

情報の多重化では，文字情報は文字情報として配信できる．しかし，メディアの特性に適した伝送形態，プロトコルを選択することはできない．

単一ソースとしての配信は動画情報と音声情報に対して使われることが多い．Microsoft の AVI（Audio Video Interleave format）[13] フォーマットでは RIFF（Resource Interchange File Format）チャンクという単位で情報の多重化を行なう．動画情報と音声情報はチャンク毎に独立して格納される．タイムスタンプを各チャンクに付加することで表示アプリケーションは時間的関係を維持した表示が可能となる．

動画情報と音声情報はいずれも連続性メディアであり，そのメディアとしての特性は類似している．複合メディアを構成するメディアの特性に類似性があり，且つ全てのメディアが 1 台の送信ノードから配信される場合，単一ソースとしての配信は有効である．

既存の多くの複合メディアを実時間配信するアプリケーションは単一ソースとしての配信を行なっている．

複数ソースとしての配信

複数ソースとしての配信を図 2.7に示す.

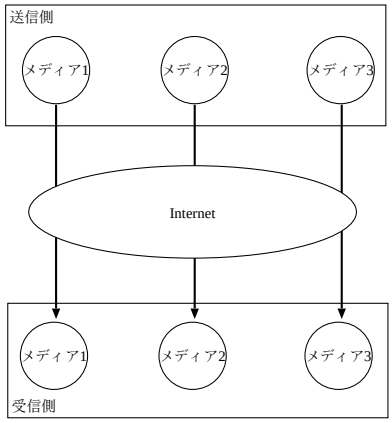

図 2.7: 複数ソースとしての配信

複数ソースとしての配信では，複合メディアを構成するメディアを個々に配信する．受信側は受信した複数のメディアを複合メディアとして構成し，表示を行なう.

1.2節で述べた HTML，SMIL や HTML+TIME[14] は，画像や文字，動画の情報をサーバから個別に受信して表示を行なうため，複数ソースとしての配信を実現していると言える.

複数ソースの配信では，メディア間の相互関係は受信側で再現される必要がある．時間的関係の構成については，ネットワーク上のジッタや復号化処理時間を考慮する必要があるため，単一ソースによる配信の方が容易であると考えられる.

しかし，単一ソースとしての配信は複数ソースの配信と比較して，以下に挙げる 3 つの欠点がある.

- 送信側の分散化
  配信を行なうノードが多重化を行なう必要があるため，伝送処理の分散化を行なえない．これによって，処理の負荷が送信ノードに集中してしまう.

- メディアに応じた伝送手段の選択
  単一ソースとしての配信では，伝送手段がメディアの特性に関わらず統一されるため，文字情報は TCP を用いて伝送し，動画情報は UDP で伝送するといった伝送手段の選択ができない．複数ソースとしての配信では，プロトコルのみならず，メディア毎に適切な QOS（Quality Of Service）を適用できる.

- 複合メディアを構成するメディアの選択
  例えば，映画を配信する際に字幕情報の言語を数種類用意する場合，単一ソースの配信では，全ての字幕情報に共通の動画情報を符号化する必要がある．複数ソースの配信を用いれば，受信側で受信する字幕情報を選択することが可能となり，送信側の処理コスト及び運用コストを削減できるとともに柔軟な拡張が可能となる.

以上の理由により，多様なあるいは柔軟な複合メディアコンテンツの配信には，複数ソースとしての配信が適していると考える.

第3章　複合メディアの同期

本章では複合メディアにおける同期の定義を行ない，同期処理が必要となる要因について検討を行なう．さらに，同期処理の基礎となるメディア間の時間的相互関係の記述と同期点について述べる．

3.1　複合メディアにおける同期

複合メディアにおける同期とは，複合メディアを構成するメディア間の時間的関係に基づいた表示処理を行なうことである．複合メディアの同期は，メディア内同期（intra-media synchronization）とメディア間同期（inter-media synchronization）の 2 種類に分類できる．

3.1.1　メディア内同期

メディア内同期は，連続性メディアにおいて，不連続のない表示処理を行なうための同期である．よって，メディア内同期処理は複合メディアを構成する個々の連続性メディア毎に行なわれるが，全ての連続性メディアのメディア内同期が実現されなければ，複合メディアとして情報を正確に伝えられないため，複合メディアの表示処理においても行なう必要がある．

図 3.1にメディア内同期処理の例を示す．例えば $20fps$（frames per second）の映像では，フレームはメディア内同期処理によって，$50msec$（milliseconds）毎に表示される．

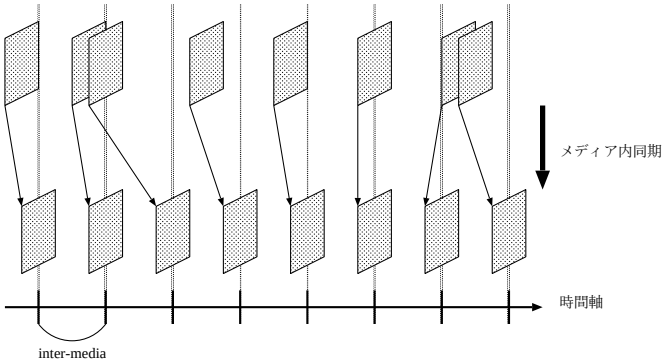

図 3.1: メディア内同期

メディア内同期に要求される精度は，メディアの形式によって異なるため，メディア毎に考慮する必要がある．また，メディアの種類によって許容される遅延・不連続の範囲も異なる．動画情報と音声情報では，音声情報の方が連続性が高いため，不連続性に対する許容範囲は狭い．

3.1.2 メディア間同期

メディア間同期は，複合メディアを構成するメディア間に定義される時間的関係に基づいた表示を行なうための同期である．メディア間同期は絶対的同期と相対的同期の 2 つに分類できる．

絶対的同期

絶対的同期ではメディア間の時間的関係では，メディアの開始時点はある 1 つの基準となる時間軸に基づいて決定される．

例えば基準となる時間軸を実時間として，地点 A と地点 B から実時間配信を受信した場合，地点 A で起きた a1 という事象の 1 分後に地点 B で b1 という事象が起きたとすると，図 3.2に示すように，b1 は a1 の 1 分後に表示されなければならない．

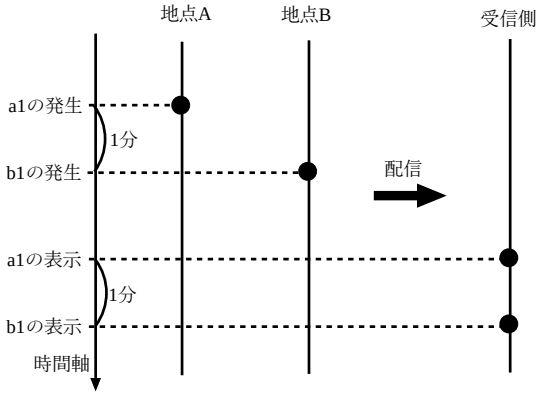

図 3.2: 複合メディアの絶対的同期

相対的同期

相対的同期ではメディア間の順序関係，開始点あるいは終点からの相対時間によってメディア間の時間的関係を定める．

SMIL では，*PAR* タグや *SEQ* タグを用いることでメディア間の順次及び同時関係を定める．また *video* タグや *audio* タグ等の *begin* 引数によって表示開始までの相対時間を指定できる．このように，メディア間の相対時間で時間関係を定めているため，SMIL による同期は相対的同期である．

同期の精度

メディア間同期に要求される精度は，複合メディアを構成するメディアの形式，内容によって異なる．

映画の動画と字幕情報を同期させる場合は，シーンの切り替わりに合わせて字幕情報を表示させる．シーンの切り替わりと字幕情報の表示が 1 秒程度ずれたとしても，試聴者は内容を理解できる．しかし，吹替え情報を同期させる場合は，動画情報と音声情報のリップシンクロが要求されるため，要求される同期精度はより高いものとなる．[15].

3.2 同期に影響を与える要素

メディアの配信では，メディアが伝送されるネットワーク環境及びメディアの復号等の処理環境が，受信者毎或は時間によって異なるため，情報の受信間隔や表示までに要する時間は変化する．よって，メディア同期処理はこれらの動的に変化する要素を考慮しなければならない．

考慮すべき要素はその要因と発生箇所から表 3.1の 3 つに分類できる．

表 3.1: 同期に影響を与える要素

要素	説明
符号化時間	データの符号化・圧縮処理に要する時間
伝送時間	ネットワーク伝送に要する時間
復号化時間	データの復号化解凍処理に要する時間

メディア A とメディア B からなる復号メディアを，複数ソースとして配信する例を図 3.3に示す．送信側で同時刻に入力されたメディア A とメディア B は，メディア処理，ネットワーク伝送に要する処理時間が異なるため，受信側では表示時刻にギャップが生じ，同時に表示されない．

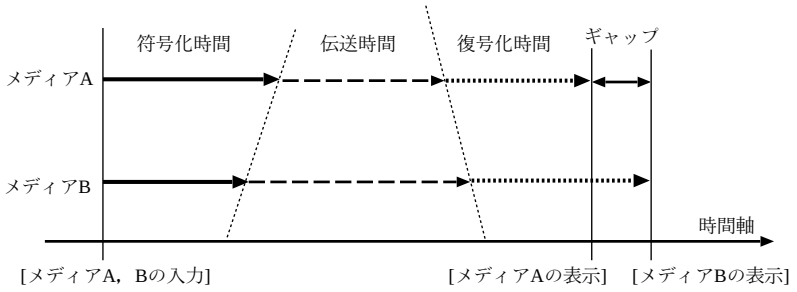

図 3.3: メディア配信のギャップ

符号化処理時間，復号化処理時間はメディア処理遅延の影響を受け，伝送処理時間は伝送遅延の影響を受ける．これらの処理時間は環境の変化に応じて動的に変化する．

3.2.1 メディア処理遅延

メディア処理遅延はメディア情報の符号化・復号化処理によって発生する遅延である．

計算機の性能への依存

メディア処理時間は計算機の性能によって変化する．符号化・復号化処理時間は以下に挙げる 3 つの要因の影響を受ける．

- CPU の処理能力
  計算機に搭載されている CPU の処理能力によって処理時間が異なる．
- メモリの容量とアクセススピード
  利用できるメモリの容量，アクセススピードによって処理時間が異なる．

- OS（Operating System）のプロセススケジューリング
  プロセスのスケジューリングポリシーの違いによって処理時間が異なる．

上記の要因に加えて，処理ノード内の資源使用状況も要因となる．メディアを処理するプロセスとは別に，多くの資源を使用するプロセスが実行されている場合，メディア処理プロセスが単位時間あたりに使用できる資源が減少するため，処理時間はより長くなる．

メディア形式への依存

メディア情報の形式によって，符号化・圧縮アルゴリズムは異なり，計算機が必要とする資源の量が変化するため，符号・復号化処理時間は異なる．

また，同じアルゴリズムでもパラメータによって処理時間は変化する．Real Video のエンコードでは，対象とする帯域を $250Kbps$ とした場合と $500Kbps$ とした場合では，$500Kbps$ でエンコードする方がより多くの処理時間を必要とする．

単位時間あたりの情報量への依存

単位時間あたりの情報量は符号化・復号化処理時間に影響を及ぼす．単位時間辺りの情報量は，メディアの種類によって異なり，情報量が多ければ，符号化・復号化処理は処理コストが増加するため，多くの処理時間を要する．

3.2.2 ネットワーク伝送遅延

ネットワーク伝送遅延はネットワーク伝送に要する時間の違いによって発生する遅延である．

ネットワーク環境への依存

ネットワーク伝送に要する時間はネットワークの状態に影響を受ける．以下に挙げる 5 つの要素が組合わさることによって，ネットワーク上における遅延は不均一なものとなる．

- 伝送メディアの速度
- ネットワークの帯域
- ネットワークの混雑状況
- ルータの性能
- 経路の変化

このため情報の伝送時間には不均一性が生じ，これをジッタと呼ぶ．ジッタの特徴として，遅延時間は急激に変化することは稀であり，ほとんどの変化は少しずつ移行し，ある範囲に収束することが知られている [5].

伝送形態への依存

2.2.2項で述べた中継点を介する配信では，送信ノードから中継点までの伝送時間と中継点から受信ノードまでの伝送時間を考慮しなければならない．しかし，中継点が設置されている場合，受信ノードが，送信ノードと中継点間の伝送時間を取得することは困難である．

単位時間あたりの情報量への依存

ネットワーク伝送において単位時間あたりの情報量が増えれば，ネットワーク伝送処理に要するコストは増大する．途中経路のルータのパケット転送処理量は増加し，ネットワークの帯域が低い場合はキューイング処理が必要となるため，より多くの処理時間が必要となる．

さらに，単位時間あたりの情報量の増加によって，伝送を行なうネットワークの使用率が上がるので，パケットロスやジッタの発生する可能性が高くなる．

3.3 同期点

同期点とは，メディア間の同期をとるための基準となる時間情報である．メディア間同期を行なうためには，メディア間の時間的相互関係が同期点を基に記述される必要がある．複数ソースとしての配信では，記述された相互関係も受信側に伝送される必要がある．受信側はこの記述された相互関係に基づいて複合メディアの再構成を行なう．

3.3.1 時間的相互関係の記述

相互関係の記述については，多くのモデルが提案されている．文献 [16] では，ある 2 つのインターバルの相関関係を 13 種類の時間的関係に分類し表している．文献 [17] では，空インターバルを導入することで，2 つのインターバル間の全ての相互関係が図 3.4に示す順次関係（meets）と同時関係（equals）の 2 つの概念で表せることが示されている．

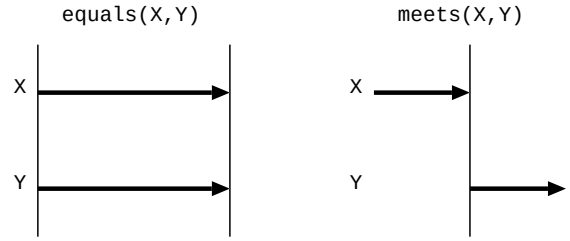

図 3.4: インターバル間の相互関係

同時関係 $equals(X,Y)$ はメディア X とメディア Y について，X の始点 X_{start} と Y_{start} の始点 Y_{start} が同時であり，且つ X の終点 X_{finish} と Y の終点 Y_{finish} が同時であることを示す．順次関係 $meets(X,Y)$ は X_{finish} と Y_{start} が同時であることを示す．インターバル間の相互関係における同期点はインターバルの始点と終点となる．

今日，広く利用されている記述方式としては SMIL や HTML+TIME が挙げられる．SMIL で規定されている *PAR* タグや *SEQ* タグの動作は *equals* と *meets* の組合せとして解釈できる．

再生時間の異なる動画メディア X と動画メディア Y の再生を同時に開始するということは以下のように表せる．X の再生時間を X_{length}，Y の再生時間を Y_{length} とし，X_{length} は Y_{length} よりも短いとする．

1. 再生時間が $(Y_{length} - X_{length})$ となる空インターバル $e1$ を定義する．
2. X と $e1$ が順次関係 $meets(X, e1)$ をとる複合メディア Z を定義する．
3. Z と Y との間で同時関係 $equals(Z, Y)$ をとる．

SMIL では図 3.5のように表される．SMIL では，空インターバルは明示的に記述されない．

```
<SMIL>
 <PAR>
  <VIDEO SRC="x">
  <VIDEO SRC="y">
 </PAR>
</SMIL>
```

図 3.5: SMIL で記述した x と y の同時再生

インターバルを基にしたメディア間同期処理機構として FLIPS[18] が挙げられる．FLIPS では，メディアの同期点として *IDLE*，*READY*，*INPROCESS*，*FINISHED*，*COMPLETE* の 5 点を定義し，この同期点に基づいてメディア間同期処理を行なう．

3.3.2 同期点を基にした相互関係

実時間に配信されるメディアや，ユーザとの対話によって再生時間が決定するメディアは終点が未知なため，インターバルを基にした相互関係の記述方法ではメディア間の相互関係を記述できない．

文献 [19] ではインターバル中の任意の点に同期点を設け，その同期点の関係により時間的関係を表現する手法が提案されている．

インターバル X と Y の同期点は以下のように定義される．

$$X = \{X_{start}, X_1, X_2, ..., X_n, X_{finish}\}$$
$$Y = \{Y_{start}, Y_1, Y_2, ..., Y_m, Y_{finish}\}$$

X_n，Y_m は同期点であり，X_n は X_{start} からの相対時間で表現される．また，文献 [19] では，任意の同期点を設けることでメディア間の時間的関係は同期点の順次関係のみで表せることを示している．

あるメディアの任意の同期点はそのメディアの始点からの相対時間で表されるため，このメディアを扱うシステムはメディアの終点を把握する必要がない．よって，この手法は再生時間が未知のメディアに適用することができる．

3.3.3 動的な同期点の生成

RPT(Remote Presentation Tool)[20] は，遠隔地間で PowerPoint[21] のスライドショー機能の動作を同期するアプリケーションである．

RPT の概要を図 3.6に示す．RPT は発信者のスライドショーに対する操作を監視し，操作内容を RPT メッセージとして受信者に伝える．RPT メッセージを受信すると，メッセージに記された内容にしたがって PowerPoint の操作を行なう．

RPT では，スライドの表示処理を同期点とし，同期点の間隔を動的に受信者に伝えることで，PowerPoint 操作の同期を実現する．ユーザとの対話によって動作するメディアでは，そのメディアの特性に合わせた同期点の生成が必要である．

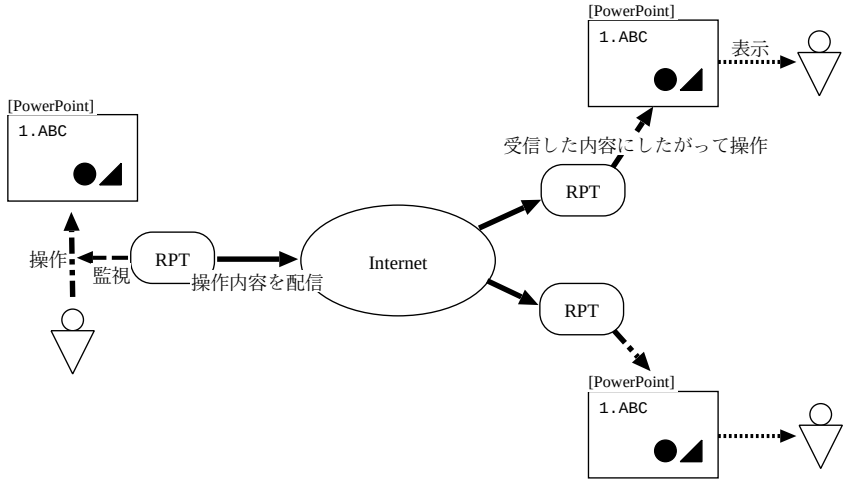

図 3.6: RPT の概要

第4章　自律メディア同期機構の設計

本章では，メディア同期機構の要求事項を明らかにした上でその設計について述べ，自律メディア同期モデルの提案を行なう．

4.1　要求事項

本研究では，複合メディアを構成するメディアを制限することなく，蓄積型配信及び，実時間配信を可能とする複合メディアコンテンツ配信環境の実現を目指し，下記の 2 項目を要求事項として複合メディア同期機構の設計を行なう．

- 多様なメディアへの対応の実現
- 動的な時間的相互関係の定義の実現

さまざまなメディアを自由に組み合わせたコンテンツ配信環境を実現するためには，配信環境は，複合メディアコンテンツを構成するメディアの形式や配信方法を制限してはならない．

しかし，2章，3章で述べたように，メディアの特性は，種類，形式，伝送形態によって多岐にわたる．また，同期処理において考慮するべき要因もメディア毎に異なる．さらに，メディアの符号化，復号化や表示処理に要する処理時間は，受信環境毎に異なり，環境の変化に応じて動的に変化する．メディア間の時間的相互関係を実現するには，多様なメディアに対する処理時間の差異を吸収する機構が必要である．

また，複合メディアコンテンツの実時間配信には，3.3.3項で述べたように，動的にメディア間の時間的相互関係を定義する機能が必要である．さらに，1.2節で述べたように，実時間配信では，送信者が各受信者の試聴状況を把握できる機能が必要である．

4.2　関連研究

本節では，本研究の関連研究について検討を行なう．

4.2.1　NSync

NSync[22] は，連続性メディア，非連続性メディアから構成される複合メディアコンテンツを対象とした同期機構であり，*Synchronization definition language* と *Run-time presentation management system* の 2 つのコンポーネントから構成される．

Synchronization definition language は，メディア間の時間的関係の記述を行なう言語である．*When* 節によって同期点の条件記述を行ない，条件が真だった場合は記述される動作の処理を行な

う．条件には各メディアの時間軸に対する加減演算，論理積，論理和，否定の論理演算が記述できる．

Synchronization definition language で記述されたメディア間同期関係は，*Run-time presentation management system* によって複合メディアコンテンツの表示と並列して処理され，メディア間の同期処理が行なわれる．

NSync のメディア間同期処理は個々のメディアに依存した時間軸にしたがって行なわれる．Nsycn では個々のメディアの時間軸にしたがった同期処理を行なうことによって，同期点のメディア内への定義を実現する．

しかし，ユーザの操作等によって，動的に時間的相互関係を記述するには，言語仕様が複雑であること，メディア毎に時間軸が存在し，同一の事象が複数の方法で記述できることから，送信者，受信者間の時間管理が複雑になると予測でき，NSync の実時間配信への対応は困難である．

4.2.2 Flow Scheduling Framework

FSF（Flow Scheduling Framework）[23] は，コンポーネントモジュール方式のアーキテクチャを持ったメディア同期機構である．FSF のアーキテクチャを図 4.1に示す．

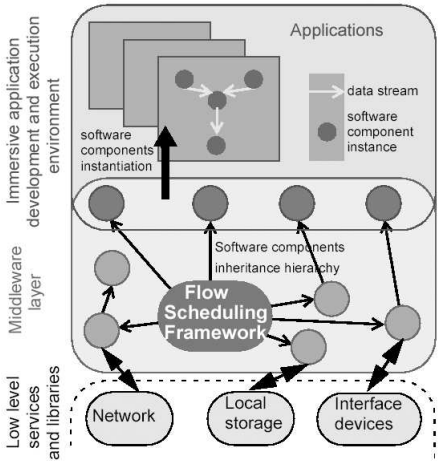

図 4.1: FSF のアーキテクチャ

FSF はミドルウェアレイヤで動作し，メディア情報のデータフローを基準とした基本的な同期機能を提供する．各コンポーネントは *Xcell* と呼ばれるオブジェクトを単位に構成され，メディア情報は *Xcell* 毎に処理される．

FSF で処理するメディア情報には時刻情報が含まれることを前提とし，メディアのデータフローは *Active Stream* と *Passive Stream* に区別される．*Active Stream* はデータフローに固有の揮発的な情報を維持するが，*Passive Stream* では，FSF の処理に必要となる情報のみを維持し，揮発的な情報は維持されない．全ての *Passive Stream* は FSF のメディア間同期処理の基準となる時間情報を共有しており，*Xcell* に *Active Stream* が入力されると，*Passive Stream* の時間情報を基に *Active Stream* 固有の時間情報の変換を行なう．複数の *Active Stream* は変換された時間情報を基に同期処理が行なわれる．

メディア情報の取得は *Low level services and libraries* で行なわれ，カプセル化処理によって，

メディアの形式や配信方法に対する拡張性を確保する．しかし，FSF の手法ではデータフローの形式に対して解釈・変換を行なう必要がある．メディアの形式と配信方法は密接に結び付いている場合が多いため，汎用的なコンポーネントの実装は困難であり，拡張のための開発コストは高く，メディアの多様化を充分に実現しているとは言えない．

4.3 メディア配信処理とメディア間同期処理の切り分け

本研究では，複合メディアを構成する個々のメディアの配信・表示処理とメディア間同期処理を切り分けることで，メディアの形式・配信方法を制限しない配信手法を提案する．また，メディアの形式及び配信方法を自由に選択できるため、本手法によって，メディアの特性あるいはコンテンツの内容に応じた配信方法の利用が可能とする．

4.3.1 メディア間同期処理のアプローチ

本機構は多様なメディア配信に適応するために，複数ソースとしての配信を用いる．2.4.2項で述べたように複数ソースとしての配信を行なうことによって，複合メディアを構成する個々のメディアはその特性に応じた配信方法を選択できる．

メディア間同期処理の手法は，配信プロトコルに依存する方法，メディア形式に依存する方法がある．配信プロトコルに依存する方法の例としては，RTP のソース ID とタイムスタンプを用いた同期方法が挙げられる [24]．メディア形式に依存する方法の例としては，複数の Digital Video ストリームをフレーム毎にバッファリングするメディア間同期方法 [25] や，4.2.2項で述べた FSF が挙げられる．

しかし，メディア間同期処理が配信プロトコルや配信形態に依存する場合，メディアの配信方法がメディア間同期処理によって制限されてしまい，メディアの特性に応じた配信は困難となる．また，配信プロトコルにおける同期処理では，メディアの複合化・表示処理時間に対応できないという問題がある．

メディア間同期処理がメディア形式に依存する場合，メディア間同期処理に必要となる情報が得られない形式ではメディア間同期処理を行なえない．また，新たなメディア形式を利用する際は，4.2.2項で述べたように，そのメディア形式への対応にかかる開発コストは高く，メディアの多様化に対する拡張性に問題がある．

よって，本研究では，メディアの配信方法，メディア形式に依存しないメディア間同期処理が必要と考える．

4.3.2 メディアの抽象化

メディアの形式・配信方法に依存しないメディア間同期処理を実現するために，本研究では，メディアの抽象化を行なう．

メディアを抽象化することで，メディア間同期処理に必要となる，メディアの操作及び情報を明らかにし，必要としない情報や，メディアに対する処理を隠蔽化する．

メディア間の時間的相互関係は個々のメディアの表示開始時点で実現される必要がある．他の時点でメディア間同期を実現したとしても複合メディアコンテンツとして正確に情報を伝えることはできない．

3.2節で述べたように，同期に影響を与える要素としては符号化処理時間，伝送処理時間，復号化処理時間が存在し，それぞれの処理時間について影響を与える要因も異なるが，これらの処理が終了した時点でメディアの表示が可能となるため，これらの処理はメディアの表示を行なうための準備処理として一括できる．

また，受信側において，メディア内同期が実現されていることを前提とすると，メディア間同期は表示開始時点における meets 関係のみを満たせばよいことになる．

よって，本研究ではメディアを表 4.1に示す 4 つの状態を持つオブジェクトとして抽象化を行なう．

表 4.1: メディアの状態

名称	説明
停止状態	メディア情報の受信処理が開始されるまでの状態，または メディアの表示が終了した状態
表示準備状態	メディア受信処理の開始から復号化処理の終了までの状態
表示準備完了状態	復号化処理の終了から表示処理の開始までの状態
表示状態	メディア表示処理の開始から終了までの状態

メディアの表示処理を行なうには受信処理，復号化処理が終了している必要があるが，本研究では表示準備完了状態を定義することで，メディア情報の取得・復号化処理と表示処理を区別する．区別することによって，取得・複合化処理を表示処理の直前ではなく，事前に行なうことを可能とする．

メディア間同期処理に必要となるメディア情報

表 4.1に示すメディアの状態に基づき，メディア間同期処理に必要となる情報として，以下に示す 4 つの同期点をを用いる．

- 表示準備開始点
- 表示準備終了点
- 表示開始点
- 表示終了点

メディア間同期処理では，表示予定時刻までにメディアの表示準備処理が完了している必要があり，表示準備終了点によって，表示が可能な状態を判別する．

また，実時間配信時ではユーザによって操作されるメディアについて，時間的相互関係の定義を行なうため，表示開始点，表示終了点が必要となる．

メディア間同期処理に必要となるメディア操作

メディア間同期処理に必要となるメディア操作は以下に挙げる 3 つとする．

- 表示準備開始操作

- 表示開始操作
- 表示停止操作

メディア操作はメディア間同期処理内部で実行され，表示開始操作によって，メディアは即座に表示を開始するものとする．表示開始操作の実行から実際の表示開始までの処理時間を短縮化するために，表示準備開始操作を用いる．

4.3.3 メディアオブジェクト

4.3.2項で述べたメディアの抽象化に基づき，本研究ではメディアを，操作機能 **METHOD** と同期点通知機能 **STATUS** を持つメディアオブジェクトとして定義する．

メディアオブジェクトは **METHOD** 命令によって操作され，メディアの状態が遷移する際に，**STATUS** 通知によって同期点を通知する．よって，**STATUS** 同期点通知は各 **METHOD** 命令に 1 対 1 で対応する．**METHOD** を表 4.2に **STATUS** を表 4.3に示す．

表 4.2: MEDIA METHOD

名称	説明
OPEN	メディア情報の表示準備を行なう． 停止状態から表示準備状態に遷移する．
PLAY	メディア情報の表示を行なう． 表示準備状態から表示状態に遷移する．
STOP	メディア情報の表示を一時的に停止する． 表示状態から表示準備状態に遷移する．
CLOSE	メディア情報の表示を停止する． 表示状態から停止状態に遷移する．

表 4.3: MEDIA STATUS

名称	説明
OPENED	表示準備状態終了点を通知する
PLAYED	表示開始点を通知する
STOPPED	表示状態から表示準備完了状態への遷移を通知する
CLOSED	表示状態から停止状態への遷移を通知する

メディアオブジェクトでは，**METHOD** 命令に 4.3.2項で述べたメディア操作には無い *STOP* 命令を追加した．*STOP* 命令も *CLOSE* 命令もともにメディアの表示を停止するが，停止後のメディアの状態が異なる．*STOP* 命令では，即座に表示が再開できる表示準備完了状態に遷移するが，*CLOSE* 命令では，表示に再度表示準備処理が必要となる停止状態に遷移する．

複合メディアコンテンツでは，1 つのメディアの表示が停止された場合に，時間的相互関係を維持する目的で，他のメディアの表示を停止する必要性が考えられる．再度コンテンツの表示再開時に，表示準備処理の実行を避けるために *STOP* 操作を追加した．しかし，2.1節で述べたように，非連続メディアの表示時間は限りなく 0 に近いので，*STOP* 命令及び *STOPPED* 通知は無効とする．

また，*PLAYED* 状態は受信ノードではメディアの表示された時点，送信ノードでは配信のための符合化処理開始時点に通知されるものとする．

図 4.2にメディアオブジェクトの状態遷移を示す．

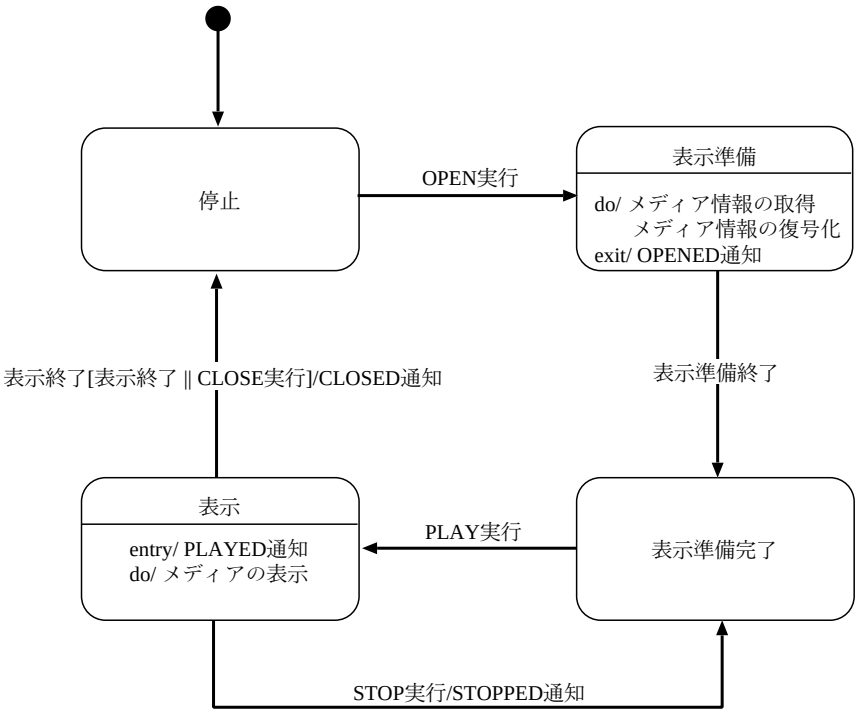

図 4.2: メディアオブジェクトの状態遷移

メディアオブジェクトは最初停止状態にある．*OPEN* 命令によってメディア情報の取得及び復号化処理を行なう．メディア情報の取得，復号化処理が終了すると *OPENED* 通知を発行し，表示準備完了状態に入る，*PLAY* 命令によって，メディアの表示を開始し *PLAYED* 通知を行なう．*STOP* 命令による表示の停止，あるいは障害による表示の一時停止の発生によって，表示準備完了状態に戻り *STOPPED* 通知を行なう．表示状態の際に *CLOSE* 命令によって，*CLOSED* 状態を通知して停止状態に戻る．

本研究では，**STATUS** 通知によってメディアの状態を管理し，**METHOD** 命令によってメディアを操作することで，メディア間同期処理を行なう．

4.4 同期モデル

本節では，メディアオブジェクトを用いたメディア間同期手法について述べる．

3.3.2項で述べたように，インターバルを基にした同期処理では，複合メディアコンテンツの実時間配信は実現できない．これに対し，4.2.1項で述べた NSync では，個々のメディアの時間軸を基に複数の時間軸を管理し，メディア間同期処理を行なう．しかし，NSync の手法では，時間軸の存在しないメディアへの対応が困難である，また，同一の事象が複数の方法で記述できるという問題がある．

よって，本研究では単一の基準時間軸を基にした絶対的同期を行なう．基準時間軸を設定するこ

とで再生時間が未知のメディアが利用可能となり，メディア間の時間的相互関係は一意に定められる．

基準時間軸の精度は実装あるいはコンテンツの要求する精度値に依存するものとする．

4.4.1 基準時間軸と基準メディア

本研究では，基準時間軸を基にしたメディア間同期処理を行なうと述べた．しかし，連続メディアの再生では，バッファリングされた情報量の不足により発生するバッファアンダーフローやメディアモジュールが使用できる CPU リソースの低下等の障害によって，不連続が発生する場合がある．

基準時間軸をノード内のローカル時計とした場合，連続メディアの不連続が発生した場合も基準時間軸は独立して進行してしまうため，メディアの表示に不連続が発生すると，メディア間の時間的相互関係は維持できない．本研究ではこの問題を解決するために基準メディアを設置する．

本研究では，複合メディアを構成するメディアには主従関係があることを前提とし，その主となるメディアを基準メディアとする．主メディアは基準メディアの中心となるメディアであり，例えば映画なら動画，講義ならば講師の音声あるいは映像となる．基準メディアはコンテンツの内容に依って変わるので，ユーザによって指定されるものとする．

基準時間軸の進行には基準メディアの状態が反映される．基準メディアが表示状態に遷移すると基準時間軸の進行が開始される．また，基準メディアがユーザ操作，正常な再生終了あるいは障害によって，表示準備完了状態または停止状態に遷移した場合は基準時間軸の進行を止める．基準メディアが再度表示状態に入ると基準時間軸の進行も再開される．

基準メディアの表示状況に応じて基準時間軸の進行を管理することで，基準メディアと他のメディアとの間の時間的相互関係を維持できる．

4.4.2 基準時間軸に基づいたメディア間同期

本手法では，基準時間軸を基に，メディアオブジェクトの同期点を指定することで，メディア間の時間的相互関係の定義を行なう．具体的には，複合メディアコンテンツを構成する個々のメディアオブジェクトについて，基準時間軸上の操作時刻を指定する．

あるメディアオブジェクトが指定された表示時刻に基準時間が到達すると，そのメディアオブジェクトの *PLAY* 命令を実行する．

表示予定時刻までに，メディアの表示準備処理が完了しなかった場合，あるいは表示状態中の連続性メディアで不連続が生じた場合は，基準メディア及び他の連続性メディアの表示を一時停止し，該当のメディアが正常に処理されるまで待機することで時間関係の維持を行なう．非連続性メディア及び表示が開始されていないメディアについては，4.4.1項で述べたように基準メディアの操作によって基準時間軸の進行も停止するので，メディア間の時間的相互関係は維持される．

本手法により，受信ノード毎に基準時間軸を基にしたメディア間の時間的相互関係が自律管理されるため，個々の受信ノード上でネットワーク環境や計算機環境に依存せずに，基準メディアとその他のメディアとの間の時間的関係が維持できる．

4.5 受信者の試聴状況の把握

1.2節で述べたように，複合メディアコンテンツの実時間配信を実現するには，受信側におけるメディア間同期処理のみでなく，送信側における各受信者の試聴状況を把握できる機能が必要である．試聴状況を確認することによって，配信者は受信者の試聴状況に応じて，動的に複合メディアコンテンツを構成できる．

しかし，前述のように複合メディアを構成するメディアの表示に要する処理時間は各受信者の環境によって異なる．また，本研究の提案するメディア間同期手法では，個々の受信ノードは自律的に基準時間軸を管理するため，受信ノード間の試聴状況の同期をとることはできない．そこで，本研究では各受信ノードが送信ノードに対して処理の完了を伝えるフィードバック情報を送信することで，配信者による各受信者の試聴状況の把握を実現する．

フィードバック機能により，送信者側における受信者間の試聴状況の同期調整が可能となり，複合メディアの実時間配信において，各受信者の試聴状況に合わせた複合メディアコンテンツの動的な構成が実現できる．

4.6 同期情報の記述

本研究が提案するメディア間同期処理は，メディアの形式・配信形態に依存しないため，メディア間の時間的相互関係に関する情報も，個々のメディア情報から独立して受信される必要がある．また，4.5節で述べた試聴状況のフィードバックでは，受信者から送信者に対してフィードバック情報の送信を行なう．

そこで，本研究では，メディア操作を指示するための情報（操作指示情報）と，試聴状況をフォードバックするための情報（フィードバック情報）を同期情報とし，記述方法の定義を行なう．実時間配信時は同期情報は動的に送信ノードによって生成され，受信ノードによって実時間的に処理されるため，記述及び解析が容易である必要がある．

同期情報は操作指示情報、フィードバック情報ともに以下に示す形式をとり，各パラメータは空白によって区切られる．

< 表示時刻情報 > < 同期情報識別子 > < メディア識別子 > [< メディア表示位置情報 >]

表示時刻情報は基準時間軸上の時刻を指定し，操作指示情報の場合はメディアオブジェクトの操作を行なう時刻を表し，フィードバック情報の場合はメディアオブジェクトの操作が完了した時刻を表す．

表示時刻情報には時刻を表す整数値あるいは”-”（ハイフン）が指定される．”-”は操作指示情報の場合にのみ指定することが可能であり，”-”が指定された場合は，その操作は受信と同時に実行されるものとする．”-”は主にメディアオブジェクトに対する表示準備処理の指示に使用する．

同期情報識別子は同期情報の種類を定める要素であり，メディアオブジェクトの **METHOD** 命令及び **STATUS** 同期点通知に基づいて定義される．

操作指示情報として *OPEN*，*PLAY*，*STOP*，*CLOSE*，*MASTER* の 5 つの値が指定できる．*MASTER* はそのメディアを基準メディアとすることを指示する．基準時間軸は基準メディアによって管理されるため，送信ノードと受信ノードは共通の基準メディアを設定する必要がある．他の 4 つの値はメディアオブジェクトの **METHOD** 命令に対応し，メディア操作を指示する．

フィードバック情報には *OPENED*，*PLAYED*，*STOPPED*，*CLOSED* の 4 つの値が指定できる．それぞれ **STATUS** 同期点の通知に対応する．

メディア識別子は同期情報の対象となるメディアを識別する文字列情報とする．メディア識別子を用いて，同期情報の対象となるメディアオブジェクトの識別を行なうため，複合メディアを構成する個々のメディアについて一意のメディア識別子が割り当てられる必要がある．メディア識別子の実装例としては URI（Uniform Resource Identifier）[26] が挙げられる．実時間配信の場合は，メディア識別子は送信ノードと受信ノードとで共通の値を用いる．

メディア表示位置情報は，メディアオブジェクトの **METHOD** 命令を実行する際に必要となる情報を指定するために用いる．メディア表示位置情報は必要ない場合は省略できる．

メディア表示位置情報はメディアオブジェクト毎に定義される．例えば動画情報の場合は再生経過時間あるいはフレーム番号が位置情報として利用できる．また，スライド資料の場合はスライドのページ番号が利用できる．

4.7 自律メディア同期モデル

本研究の提案する自律メディア同期モデルを図 4.3に示す．本機構は複合メディアコンテンツを構成するメディアのメディアオブジェクト群と，メディア間同期モジュール，通信モジュールの 2 つのモジュールから構成される．

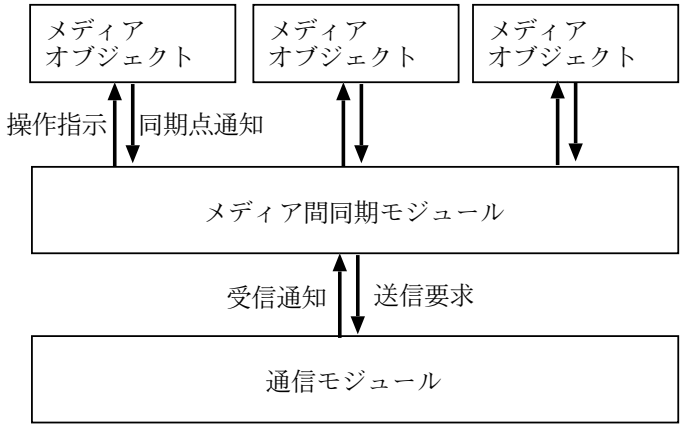

図 4.3: 自律メディア同期モデル

複合メディアコンテンツを構成する個々のメディアは，メディアオブジェクトとして抽象化され，メディア間同期モジュールによって管理される．通信モジュールによって同期情報の取得を行ない，メディア間同期モジュールが操作指示情報にしたがってメディアオブジェクトの操作を行なう．

4.7.1 通信モジュール

通信モジュールは，蓄積型配信の場合は操作指示情報の受信，実時間配信の場合は操作指示情報の受信及びフィードバック情報の送信を行ない，受信した同期情報をメディア間同期モジュールに通知する．

同期情報の取得方法は，蓄積型配信の場合は，同期情報はローカルファイルシステム上のファイル，ネットワーク上のサーバによって蓄積型配信されるファイルが想定できる．また，実時間配信では，送信ノードと受信ノードは独自プロトコルで双方向通信を行なうことを想定する．

このように複合メディアコンテンツの配信形態によって同期情報の取得方法が異なるため，本モデルでは，メディア間同期モジュールと通信モジュールを分けることで，同期情報の取得方法に応じた通信モジュールの選択を実現する．受信者は同期情報の取得方法に応じた通信モジュールを選択することで同期情報の取得を行なう．

4.7.2 メディア間同期モジュール

メディア間同期モジュールは本モデルの中心となるモジュールであり，メディア間同期処理に必要な情報の管理及びメディア間同期処理を行なう．メディア間同期モジュールの役割は以下の 4 つである．

- 基準時間軸の管理
- メディアオブジェクトの操作及び通知の受信
- メディア操作指示情報の管理
- フィードバック情報の送信

メディア間同期モジュールは，全てのメディアオブジェクトの操作を行ない，全てのメディアオブジェクトからの同期点通知を受けとる．また，通信モジュールが受信した操作指示情報を受けとり，実時間配信時はメディアオブジェクトの同期点通知を基にフィードバック情報を生成し，通信モジュールを介して送信を行なう．

受信した操作指示情報は表示時刻情報を基に管理し，表示時刻に到達したメディアオブジェクトに対してメディア表示位置情報とともに **METHOD** 命令を実行することでメディアの操作を行なう．

基準時間軸は，*MASTER* 操作指示によって指定された基準メディアオブジェクトからの同期点通知情報を基に 4.4.1項で述べた方法にしたがって管理する．

4.7.3 本モデルを用いた実時間配信

実時間配信時のシステムモデルを図 4.4に示す．

実時間配信時は，送信ノードにもメディア間同期モジュールと通信モジュールが必要となり，同期情報は，送信ノード上の通信モジュールと受信ノード上の通信モジュールとの間で交換される．

送信ノードでは，メディアオブジェクトの操作は配信者であるユーザが行なうことを想定する．このため，メディア間同期モジュールは，メディアオブジェクトからの **STATUS** 状態通知を受けるのみとなる．また，受信ノードのメディア間同期モジュールはメディアオブジェクトからの **STATUS** 同期点通知によって，フィードバック情報の生成を行なうのに対して，送信ノードでは操作指示情報の生成を行なう．**METHOD** と **STATUS** 同期点通知は 1 対 1 で対応しているため，通知された **STATUS** 同期点から操作指示情報を生成できる．配信者の操作に応じて操作指示情報を生成することで，実時間配信時のメディア間同期を実現する．

同期情報の表示時刻情報は基準時間軸から取得するが，メディア情報を直接配信するメディアオブジェクトでは，4.3.3項で述べたように，符号化開始時点が *PLAYED* 同期点として通知されるため，送信ノードの基準時間軸は受信ノードの場合と同様の方法で管理される．

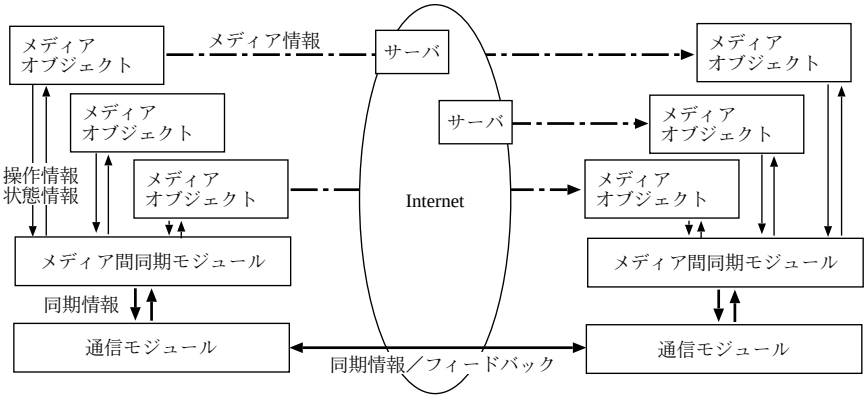

図 4.4: 本モデルを用いた実時間配信

よって，本モデルでは，メディアオブジェクトの **STATUS** 同期点通知によって生成する同期情報の種類を指定することで，受信ノードと送信ノードにおいて共通のメディア間同期モジュール及び通信モジュールが使用できるため，送信専用のモジュールを実装する必要は無い．

4.8 要求される同期精度の実現

本研究で提案する自律メディア間同期モデルによって，さまざまなメディアから構成される複合メディアコンテンツの蓄積型配信及び実時間配信が実現できる．しかし，本モデルはメディア間同期処理手法を示すものであり，その同期精度は実現方法に依存する．本節では，本モデルのメディア間同期精度に影響を与える要素を明らかにし，その機能用件について述べる．

3.1.2項で述べたように，メディア間同期に要求される精度は，複合メディアを構成するメディアによって異なるため，各メディアオブジェクトで要求される同期精度を実現する必要がある．

本モデルでは，メディア間同期は基準時間軸を基に行なうため，メディア間同期精度は基準時間軸の管理精度と，メディアオブジェクトの *PLAY* 操作処理時間によって決定される．

基準時間軸は，メディア同期処理の基準となる時間軸であり，メディアオブジェクトは基準時間にしたがって操作されるため，基準時間軸の管理精度は全てのメディア間同期に影響を与える要素となる．

また，実時間配信時では受信ノード上のメディアオブジェクトは，送信ノード上におけるメディア表示位置情報を維持する必要がある．例えば動画情報で，メディア表示位置情報に経過時間を使用する場合，受信ノードのメディアオブジェクトは，現在表示している動画の表示位置が，送信ノードで何時符号化されたものかを把握できなければならない．

メディアオブジェクトの *PLAY* 操作処理時間は，個々のメディアの同期精度に影響を与える要素である．本モデルでは，基準時間が表示予定時刻に到達した際にメディアオブジェクトの *PLAY* 操作を実行するため，*PLAY* 操作処理時間に多くの時間を要した場合，メディア間同期精度は低くなる．よって，メディアオブジェクトの *PLAY* 操作処理は，要求される精度を実現する範囲内で処理を完了する必要がある．

第5章　自律メディア同期機構SMPの設計と実装

本研究では，自律メディア同期機構のプロトタイプとして，講義の実時間配信を目的としたSMP（Synchronized Media Point）の実装を行なった．本章ではSMPの設計と実装の詳細について述べる．

5.1　SMPの設計

本節では，講義実時間配信SMPの設計について述べる．SMPは，4章で述べた本研究が提案する自律メディア同期モデルに基づいて講義の実時間配信を実現する．

5.1.1　講義を構成するメディアと配信形態

講義は，講師の音声と講義内容に則した資料の表示によって行なわれる．SMPでは，講義を，講師の様子及び音声からなる講義映像と，講義に使用される講義資料から構成される複合メディアと定義する．

講義映像は，時間によって内容が変化する連続性メディアであり，講義の進行によって動的に生成されるため，実時間配信を行なう必要がある．

これに対して，講義資料は講師の操作によって表示される非連続性メディアであり，事前に用意されるため，蓄積型配信が適している．

よって，SMPでは講義映像と講義資料を複数ソースとして配信し，それぞれのメディアに適した配信方法を使用する．

また，講義は1人の講師が複数の受講者に対して行なうのが一般的であるため，SMPは1対多の配信を行なうことで，遠隔地にいる複数の受講者の参加を実現する．

5.1.2　講義映像と講義資料の同期

講義は，講師の話に則して，適切な講義資料が選択・表示されることで構成される．講義映像と講義資料の時間的関係が保たれずに表示される場合，受講者は講義の内容を充分に理解することはできない．よって，講義の実時間配信においても，講義資料は講義映像の該当箇所で適切に表示される必要がある．

講義資料は，講義映像に則して表示されるため，SMPでは講義の経過時間に基づいて講義資料の表示を行なう．

本機構の要求事項は以下の3つである．

- 講義内容が理解可能な程度に講義映像と講義資料が同期される．

- 各受講者が自律的に同期を実現できる．
- 講師が各受講者の試聴状況を確認することができる．

5.1.3 SMP の動作概要

SMP は自律メディア同期モデルに基づいて，講義映像と講義資料のメディア間同期を行なう．図 5.1に SMP の動作概要を示す．

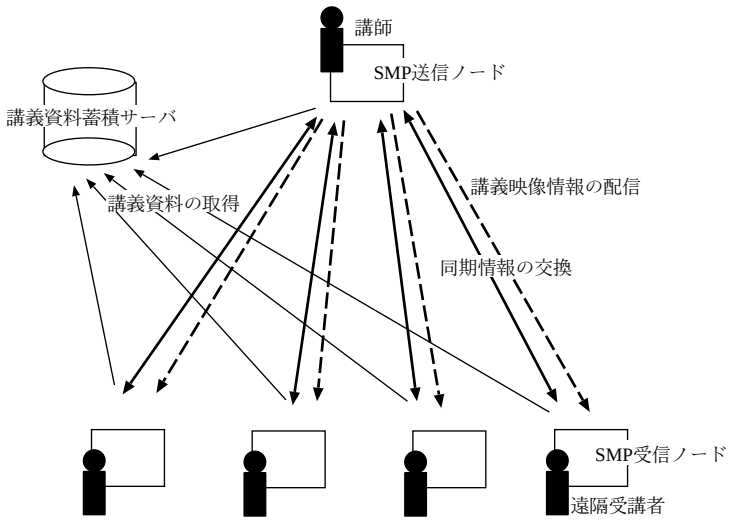

図 5.1: SMP の動作概要

SMP は講師によって操作される SMP 送信ノードと，遠隔受講者に講義コンテンツの表示を行なう SMP 受信ノードから構成される．

5.1.1項で述べたように，講義映像と講義資料はそれぞれ異なる方法で取得される．

講義映像は SMP 送信ノードから各 SMP 受信ノードに実時間配信される．SMP 送信ノードには講義映像を符号化・配信するためのメディアオブジェクトが実装され，SMP 受信ノードには講義映像を復号化・表示するためのメディアオブジェクトが実装される．

講義資料は講義資料蓄積サーバによって配信され，SMP 送信ノード及び各 SMP 受信ノードは講義中継が開始される前に講義資料の取得を行なう．

講義資料については，SMP 送信ノードが講義資料自体を配信する必要が無いため，SMP 送信ノードと SMP 受信ノードで共通のメディアオブジェクトが使用される．送信ノードでは講師の操作によって通知される講義資料の表示同期点を基に，操作指示情報を生成し，各 SMP 受信ノードに送信する．SMP 受信ノードでは，受信した操作指示情報に基づき，講義資料メディアオブジェクトを介して講義資料の表示を行なう．

講義映像を基準メディアとすることで，講義の経過時間を基にした講義資料の表示時刻の定義を実現する．

各受講者の試聴状況は，講義映像及び講義資料の表示状態が変化する度に SMP 送信ノードに送信される．講師はこの試聴状況を確認することにより，各受信者の試聴状況に合わせて講義を進行することができる．

5.2 実装の概要

5.1節で述べた設計を基に，講義の実時間配信を対象とした自律メディア配信機構 SMP の実装を行なった．SMP の動作画面を図 5.2に示す．

図 5.2: SMP の動作画面

講義を構成するメディアとして，講義映像には，Windows Media Video を採用し，講義資料には PowerPoint ファイルを採用し，それぞれのメディアオブジェクトの実装を行なった．

5.1.2で述べたように，SMP では講義の経過時間を基に講義資料の表示を行なうため，基準メディアである Windows Media Video のメディア表示位置情報は，符号化を開始してからの経過時間とした．PowerPoint ファイルは複数のスライドから構成され，それぞれのスライドには PowerPoint ファイル内で一意にスライドを識別できるページ番号が割り当てあられる．本実装では，PowerPoint のメディア位置情報としてスライドのページ番号を用いる．

メディア識別子には各メディアの URL（Uniform Resource Locator）を用いた．各メディアオブジェクトは指定された URL を基にメディア情報の取得を行なう．

本実装は，IBM PC/AT 互換機上で動作する Microsoft Windows XP Proffesional オペレーティングシステム上で，VisualC++ 7.0 言語を用いて行なった．

5.3 メディアオブジェクトの実装

本機構では，講義映像及び講義思慮はメディアオブジェクトとして抽象化される必要があるため，メディアの抽象化を実現する仕組みが必要となる．本実装では C++言語のクラスの継承を用いてメディアの抽象化を実現した．基底抽象クラス SMPMedia と SMPMediaListener を図 5.3と図 5.4 に示す．

各メディアオブジェクトは SMPMedia クラスを基底クラスとして実装される．Initialize 関数及

```
class SMPMedia
{
public:
  SMPMedia();
  virtual ~SMPMedia();

  virtual HRESULT Initialize(HWND hWnd) = 0;
  virtual HRESULT UnInitialize() = 0;
  virtual HRESULT Open(LPCTSTR mediaID, LPVOID arg) = 0;
  virtual HRESULT Play(LPCTSTR mediaID, LPVOID arg) = 0;
  virtual HRESULT Stop(LPCTSTR mediaID, LPVOID arg) = 0;
  virtual HRESULT Close(LPCTSTR mediaID, LPVOID arg) = 0;
  virtual LONG GetCurrentPosition() = 0;

  void AddListener(SMPMediaListener *listener);
  void SetMaster(BOOL fMaster);
  BOOL IsMaster();
private:
  std::list<SMPMediaLitener *> m_Listeners;
  BOOL m_fMaster;
};
```

図 5.3: SMPMedia クラス

び UnInitialize 関数は Windows アプリケーションとして必要となる初期化・破棄処理を行なうために定義した．

SMPMedia クラスの Open，Play，Stop，Close の各関数は抽象関数であり，派生クラス毎に，メディアオブジェクトの **METHOD** 命令の実装を行なう．引数はメディア識別子と LPVOID（VOID 型のポインタ）型を取り，メディア毎に定義されるメディア表示位置情報を渡す．

GetCurrentPosition 関数は現在のメディア表示位置情報を取得する抽象関数である．自律メディア同期モデルでは，基準時間軸は基準メディアのメディア表示位置情報を基に管理されるため，メディア表示位置情報を取得するインターフェースが必要である．

STATUS 同期点通知処理の実装にはオブザーバデザインパターン（Observer Design Pattern）[28] を利用した．SMPMedia オブジェクトから通知を受け取る必要があるオブジェクトは SMPMediaListener クラスを継承し，SMPMedia クラスの AddListener 関数を通してオブジェクトの登録を行なう．登録されたオブジェクトは SMPMedia クラスの m_Listeners メンバ変数に配列として保存される．

SMPMedia オブジェクトは m_Listeners メンバ変数に登録された SMPMediaListener オブジェクトの OnOpend，OnPlayed，OnStopped，OnClosed の各関数を実行することで **STATUS** 同期点の通知処理を行なう．これらの関数は，メディア識別子，メディア表示位置情報と該当のメディアオブジェクトが基準メディアかどうかを表す情報（fMaster 変数）を引数に取る．

```
class SMPMediaListener
{
public:
  virtual HRESULT OnOpend(LPCTSTR mediaID, LPVOID arg, BOOL fMaster) = 0;
  virtual HRESULT OnPlayed(LPCTSTR mediaID, LPVOID arg, BOOL fMaster) = 0;
  virtual HRESULT OnStopped(LPCTSTR mediaID, LPVOID arg, BOOL fMaster) = 0;
  virtual HRESULT OnClosed(LPCTSTR mediaID, LPVOID arg, BOOL fMaster) = 0;
};
```

図 5.4: SMPMediaListener クラス

STATUS 通知関数がメディア識別子を引数に持つのは，複数の SMPMedia オブジェクトから通知を受ける場合に通知を発行したオブジェクトを識別するためである．SetMaster 関数を用いることでメディアオブジェクトが基準メディアであることを指定し，IsMaster 関数によって基準メディアの判別を行なう．

5.3.1 COM を利用した既存のアプリケーションの利用

本実装では，メディアオブジェクトの COM 技術を利用した．COM 技術を用いることで，既存のアプリケーションを利用したメディアオブジェクトを実装できる．

既存のアプリケーションの利用によって，各メディアに対する符号化，復号化等の処理を実装する必要が無いため，メディアの多様化に対する開発コストが削減でき，拡張性を高めることができる．また，既存のアプリケーションの利用によって，講師は普段と同様に PowerPoint を操作することで，同期情報の配信が行なえるため，ユーザの操作性に対する透過性を実現できるという利点が挙げられる．上記の利点により，本実装では COM 技術の利用を採用した．

COM の概要

COM は Microsoft が提唱するオブジェクト指向プログラミングモデルの 1 つであり，OS やアプリケーションを構成する部品（コンポーネント）間のコミュニケーション方法を規定したものである．コンポーネントのインスタンスを作成する側をクライアントコンポーネント，コンポーネントのインターフェースを提供する側をサーバコンポーネントと呼ぶ．

本実装で使用した Windows Media Encoder，Windows Media Player ActiveX コントロール，PowerPoint はいずれも COM コンポーネントとして実装されており，COM 技術を用いることで操作が可能となる．表 5.1に本実装で使用した COM コンポーネントの LIBID （Library Identifier）を示す．

LIBID から各コンポーネントの CLSID（Class Identifier）とコンポーネントを操作するインターフェースの IID（Interface Identifier）を取得し，Win32 API の CoCreateInstance 関数を用いることで，インスタンスの生成を行なう．

本実装において重要となる COM の機能として以下の 2 つが挙げられる．

表 5.1: SMP で使用する COM コンポーネントの LIBID

COM コンポーネント	LIBID
Windows Media Encoder	{632B6060-BBC6-11D2-A329-006097C4E476}
Windows Media Player	{6BF52A50-394A-11d3-B153-00C04F79FAA6}
PowerPoint	{91493440-5A91-11CF-8700-00AA0060263B}

- バイナリ互換性
- 接続可能オブジェクト

バイナリ互換性

COM はオブジェクトの呼び出し規約を規定するのみで，特定の言語との関わりについて規定しないため，プログラミング言語やオペレーティングシステムに依存しない．よって，COM の仕様を満たすことで，言語に関わらずコンポーネント間の双方向通信を行なえる．

COM コンポーネントではオブジェクトとインターフェースが明確に区別されており，利用する場合はライブラリ等のバイナリからインターフェースの取得を行なう．そしてコンポーネント間では取得したインターフェースを介した操作のみが行なわれる．これにより既存の C++等のソースレベルの再利用性のある言語と比べてバイナリレベルでの再利用性を提供していると言える．

例としては Internet Explorer による Word ファイルの表示が挙げられる．Microsoft Word は COM コンポーネントとして開発されており，そのインターフェースが公開されている．Internet Explorer は COM の仕様にしたがって，Word のインスタンスを作成しインターフェースを取得することで Word ファイルの表示を実現している．この場合，Internet Explorer がクライアントコンポーネント，Word がサーバクライアントとなる．

本実装で使用した Windows Media Player，Windows Media Encoder，PowerPoint はいずれも COM コンポーネントとして実装されている．

接続可能オブジェクト

サーバコンポーネントからクライアントコンポーネントへのコミュニケーション手段が実装されているオブジェクトを接続可能オブジェクトと呼ぶ．接続可能オブジェクトの機能を利用することで，クライアントコンポーネントはサーバコンポーネントから非同期に通知を受けとることが可能となる．

接続可能オブジェクトを図 5.5に示す．通知を受けとる必要があるクライアントコンポーネントはシンクオブジェクトを持つ．通知を発行するオブジェクトは接続ポイント（IConnectionPoint）インターフェースを実装し，シンクオブジェクトが接続ポイントに接続することで通知を受けとることが可能となる．

同期処理ではサーバコンポーネントに処理を依頼すると，サーバコンポーネントの処理が終了するまで，クライアントコンポーネントは待機状態に入る．非同期処理ではクライアントコンポーネントは待機状態に入らない．

また，ユーザの操作によるサーバコンポーネントの状態変化の検知を行なう場合，同期処理ではクライアントコンポーネントはサーバコンポーネントに対してポーリング処理を行なう必要があ

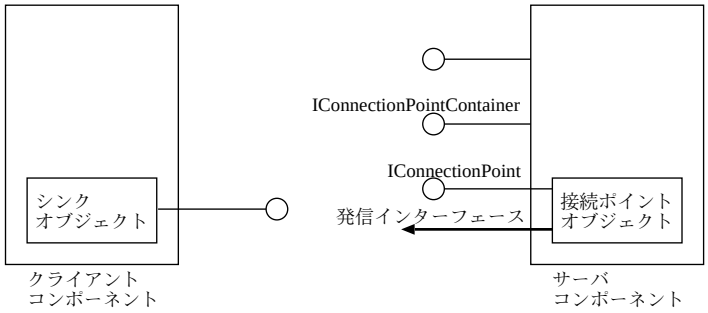

図 5.5: COM の接続可能オブジェクト

るが，非同期処理ではサーバコンポーネントから非同期に通知を受けられるため，アプリケーションは資源を有効に使用できる．

これによってクライアントコンポーネントとサーバクライアント間で双方向コミュニケーションを実現できる．

5.3.2 Windows Media Video の抽象化

5.1.3項で述べたように，Windows Media Video は SMP 送信ノードから実時間配信されるため，SMP 送信ノードと SMP 受信ノードでは，メディアオブジェクトの動作は異なり，SMP 送信ノードのメディアオブジェクトは，Windows Media Video の符号化・配信処理を行ない，SMP 受信ノードでは復号化・表示処理を行なう必要がある．

Windows Media Video の符号化・配信には Windows Media Encoder を，復号化・表示には Windows Media Player ActiveX コントロールを用いて，メディアオブジェクトの実装を行なった．使用した Windows Media Encoder と Windows Media Player ActiveX コントロールのヴァージョンはともに 7.1 である．

SMPMediaWindowsMediaEncoder クラス

SMPMedia クラスを継承する SMPMediaWindowsMediaEncoder クラスをメディアオブジェクトとして定義し，Windows Media Encoder の操作を行なう．

Windows Media Encoder は WMEncoderLib::IWMEncoderApp インターフェースを介して操作を行なう．Initialize 関数によってインスタンスの初期化を行なう．

また，SMPMediaWindowsMediaEncoder オブジェクトはシンクオブジェクトを Windows Media Encoder に接続し，符号化処理の開始同期点を受けとり，同期情報を生成する．

Windows Media Encoder のシンクオブジェクトは WMEncoderLib::_IWMEncoderEvents インターフェースとして公開されており，符号化処理の状態通知は OnArchiveStateChange 関数によって通知される．

OnArchiveStateChange 関数によって通知される状態情報を表 5.2に示す．

符号化処理の開始時に通知される WMENC_ARCHIVE_RUNNING を *PLAYED* 同期点とする．WMENC_ARHIVE_PAUSED，WMENC_ARCHIVED_STOPPED も同様にそれぞれ *STOPPED* 同期点，*CLOSED* 同期点として登録された SMPMediaListener オブジェクトに通知する．Windows

表 5.2: _IWMEncoderEvents オブジェクトによって通知される状態情報

名称	説明
WMENC_ARCHIVE_RUNNING	符号化処理を行なっている
WMENC_ARCHIVE_PAUSED	符号化処理が一時的に停止された
WMENC_ARCHIVE_STOPPED	符号化処理が停止された

Media Encoder では *OPENED* 同期点は最初の *PLAYED* 同期点の直前に発生するものとした．

GetCurrentPosition 関数の実装には WMEncoderLib::IWMEStatistics インターフェースの GetEncodingTime 関数を用いた．GetEncodingTime 関数は *msec* 単位の符号化経過時間を返す．

SMPMediaWindowsMediaPlayer クラス

Windows Media Video の表示を行なうメディアオブジェクトとして SMPMediaWindowsMediaPlayer クラスを定義した．

Windows Media Player ActiveX コントロールを使用するため，SMPMediaWindowsMediaPlayer クラスは SMPMedia クラスと CAxWindow クラスを基底クラスとする多重継承クラスとして実装した．

CAxWindow クラスは ActiveX コントロールを管理するウィンドウを生成・操作するためのクラスであり ATL（Active Template Library）内に実装されている．ATL は Microsoft が提供する COM 及び ActiveX の機能を抽象化したライブラリ群である．

Initialize 関数によって，Window の生成及び Windows Media Player ActiveX コントロールのインスタンスの生成を行なう．

Windows Media Player ActiveX コントロールの操作は WMPOCX::IWMPControl インターフェースを介して行なう．本実装で使用した IWMPControl インターフェースが公開するメソッドを表 5.3に示す．これらのメソッドを呼び出すことで Windows Media Video の再生を操作する．

表 5.3: IWMControl インターフェースによって公開される関数

関数名	説明
play	メディアの再生を開始する．
pause	メディアの再生を一時的に停止する．
stop	メディアの再生を停止する．

STATUS 状態通知は，Windows Media Encoder の場合と同様にシンクオブジェクトを実装し，Windows Media Player ActiveX コントロールに接続することで生成する．

シンクオブジェクトは WMPOCX::_WMPOCXEvents インターフェースとして公開される．メディア再生状態の通知は OnPlayStateChanged 関数によって行なわれる．通知される再生状態を表 5.4に示す．

Windows Media Player ActiveX コントロールの標準設定では，再生するメディアの URL を指定すると自動的に再生が開始される．autoStart プロパティを変更することで自動再生は抑制されるが，play メソッドの発行後にメディア情報の取得・復号化処理が開始されるため，本実装では最

表 5.4: _WMPOCXEvents オブジェクトによって通知される状態情報

状態	説明
Stopped	メディアの再生が停止されている.
Paused	メディアの再生が一時停止されている.
Playing	メディアは再生中である.
ScanForward	メディアは早送り中である.
ScanReverse	メディアは巻戻し中である.
Buffering	サーバからの情報を取得中である.
Waiting	サーバとのセッションの開始を待機中である.
MediaEnded	メディアの再生が完了した.
Transitioning	新しいメディアの再生準備中である.
Ready	メディアの再生を開始する準備ができている.

初に Playing 状態が通知された際に，pause メソッドを実行し，Paused 状態が通知された時点を Opened 同期点とした.

他の同期点については，Playing 状態の通知を *PLAYED* 同期点，Paused 状態を *STOPPED* 同期点，Stopped 状態及び MediaEnded 状態の通知を *CLOSED* 同期点として SMPMediaListener オブジェクトに通知する.

GetCurrentPosition 関数は，WMPOCX::IWMPControl インターフェースの GetcurrentPosition メソッドを実行し，再生経過時間の取得を行なう.

実時間配信時のメディア表示位置情報の伝送

本実装では，Windows Media Video は実時間配信によって配信されるため，4.8節で述べたように，SMPMediaWindowsMediaEncoder オブジェクトは，SMPMediaWindowsMediaPlayer オブジェクトにメディア表示位置情報を送信する必要がある．しかし，Windows Media Technology による実時間配信では，受信ノードが，送信ノードにおける符号化経過時間を知ることはできないため，基準メディアのメディア表示位置情報と基準時間軸は同期しない.

本実装では，Windows Media Technology のスクリプトコマンド機能を使用することで，受信ノードにおける，実時間配信時の Windows Media Video と基準時間軸の同期を実現した．スクリプトコマンド機能を用いることで，Windows Media Encoder の配信するデータストリームに文字情報を埋め込むことができる.

SMPMediaWindowsMediaEncoder オブジェクトは符号化が開始されると，5 秒毎に符号化経過時間を取得し，スクリプトコマンドとして送信する．受信したスクリプトコマンドは WMPOCX::_WMPOCXEvents インターフェースの scriptCommand 関数によって通知されため，SMPMediaWindowsMediaPlayer オブジェクトは，通知された符号化経過時間報を基に，メディア表示位置情報の修正し，基準メディアと基準時間軸を同期する.

5.3.3 PowerPoint の抽象化

PowerPoint ファイルのメディアオブジェクトとして，SMPMediaPowerPoint クラスの実装を行なった．

PowerPoint の操作を行なうメディアオブジェクトとして SMPMedia クラスの派生クラスである SMPMediaPowerPoint を定義した．PowerPoint は PowerPoint::_Application インターフェースを介して行なう．Initialize 関数によってインスタンスの初期化を行なう．SlideShow 機能の操作は PowerPoint::SlideShowView インターフェースを介してを行なう．

本実装で利用した SlideShowView インターフェースの関数を表 5.5に示す．これらの関数を用いて SMPMediaPowerpoint クラスの Play 関数を実装した．

表 5.5: SlideShowView インターフェースが公開する関数

関数名	説明
First	最初のスライドを表示する．
Last	最後のスライドを表示する．
Next	現在表示されているスライドの次のスライドを表示する．
Previous	現在表示されているスライドの前のスライドを表示する．
GotoSlide	指定されたスライドを表示する．

シンクオブジェクトは PowerPoint::EApplication インターフェースとして公開される．PowerPoint::EApplication インターフェースには，SlideShow に関する状態通知メソッドとして，表 5.6に示す 4 つのメソッドが公開されている．

表 5.6: EApplication インターフェースが公開する状態通知関数

関数名	説明
SlideShowBegin	SlideShow が開始されたことを通知する．
SlideShowNextBuild	現在表示されているアニメーション効果の変更を通知する．
SlideShowNextSlide	現在表示されているスライドの変更を通知する．
SlideShowEnd	SlideShow が終了したことを通知する．

SlideShowBegin メソッド，SlideShowEnd メソッドによる通知をそれぞれ *OPENED* 同期点，*ClOSED* 同期点とした．SlideShowNextSlide メソッドによる通知を *PLAYED* 同期点とした．

PowerPoint は 1 つの PowerPoint ファイル内に複数のスライドが含まれる，各スライドはユーザの操作によって表示が行なわれるため，本実装では PowerPoint はスライドの表示処理を行なうメディア内同期点がユーザ操作によって決定される連続性メディアとして実装を行なった．よって，*PLAYED* 同期点は各スライドの表示処理毎に通知される．

5.4 通信モジュールの実装

通信モジュールには，メディア間同期モジュールの指示による同期情報の送信機能，他通信モジュールから送信される同期情報の受信機能及び受信した同期情報のメディア間同期モジュールへの通知機能が必要である．

4.7.1節で述べたように，自律メディア間同期モデルでは，通信モジュールは，同期情報の取得方法に応じてユーザが選択できる必要がある．本実装では，通信モジュールの抽象化を行ない，通信モジュールを選択する仕組みを実装した．

また，SMP では 1 対多の配信モデルで同期情報の交換を行なう必要があることを設計として述べた．これを実現するために本実装では同期情報の交換方法として，IRC（Internet Relay Chat）[29] を用いる通信モジュールの実装を行なった．

5.4.1 通信モジュールの抽象化

配信形態により同期情報の取得方法が異なるため，抽象基底クラス SMPSocket を定義し，複数の同期情報取得方法の選択を可能とした．SMPSocket クラスの定義を図 5.6に示す．

```
class SMPSocket
{
public:
  SMPSocket();
  virtual ~SMPSocket();

  virtual int Connect(LPVOID arg) = 0;
  virtual void Close() = 0;
  virtual int Send(int arglen, LPVOID arg) = 0;

  void AddListener(SMPSocketListener *listener);
private:
  std::list<SMPSocketLitener *> m_Listeners;
};
```

図 5.6: SMPSocket クラス

Connect 関数，Close 関数は同期情報の交換を行なうためのリソースの準備及び解放を行なう関数である．

ネットワークを介して同期情報を受信する場合は Connect 関数内でソケットの生成及び接続を行なう．ファイルから同期情報を取得する場合は Connect 関数内で同期情報が記述されたファイルを開き，読み込み可能な状態にする．

実時間配信を行なう場合，同期情報は送信ノードで動的に生成されるため，通信モジュールは同期情報・フィードバック情報の送受信を非同期に行なう必要がある．SMPSocket クラスでは受信処理を別スレッド上で実行することで，独立して受信処理を行なえる設計を行なった．受信処理は Connect 関数の成功とともに開始され，Close 関数の実行によって停止する．

受信した同期情報のメディア間モジュールへの通知には，オブザーバパターンを利用し．SMPSocket オブジェクトからの通知を受けとるクラスとして SMPSocketListener クラスを定義した．SMPSocketListener クラスを図 5.7に示す．

同期情報あるいはフィードバック情報が受信された場合は OnReceived 関数が呼び出される．

```
class SMPSocketListener
{
public:
  virtual OnConnected(int status, LPVOID arg) = 0;
  virtual OnSent(int status, LPVOID arg) = 0;
  virtual OnReceived(int status, LPVOID arg) = 0;
  virtual OnClosed(int status, LPVOID arg) = 0;
};
```

図 5.7: SMPSocketListener クラス

OnReceived 関数は受信状態（エラーの有無）と受信情報を引数に取る．受信先のソケットが閉じられた場合，あるいはファイルの読み込みで EOF が返された場合は，その受信状態とともに OnClosed 関数が実行される．

5.4.2 IRC を用いた通信モジュール

本実装ではネットワークを介する通信モジュールとして SMPIRCSocket クラスの実装を行なった．SMPIRCSocket クラスは他ノードとの情報交換にを用いる．IRC を用いることで容易に 1 対多の双方向通信基盤を構築できる．

SMPIRCSocket クラスでは IRC のクライアントプロトコル [30] を実装した．IRC サーバに接続するために必要となる PASS，USER，NICK コマンドの発行及び IRC サーバとの接続を維持するために必要となる，PING コマンドの受信に対する PONG コマンドの送信は SMPIRCSocket オブジェクト内部で行なわれる．

送信ノードと受信ノードは共通の IRC チャンネルに参加し，同期情報の交換を行なう．同期情報は PRIVMSG コマンドによって以下の形式で送信を行なう．

$$PRIVMSG \ < \text{チャンネル名} > :< \text{同期情報} > CRLF$$

SMPIRCSocket オブジェクトが PRIVMSG コマンドを受信すると，同期情報が正しい形式で含まれていることを確認する．PRIVMSG に同期情報が含まれていた場合は，OnReceive 関数によって SMPSocketListener オブジェクトに同期情報を渡す．

5.5 メディア間同期モジュールの実装

メディア間同期モジュールは以下に挙げる 3 つの機能を提供する．

- 基準時間軸の管理
  基準メディアに基づいた基準時間軸の管理
- 同期情報の生成及び解析

SMPMedia オブジェクトの同期点通知による同期情報の生成及び，SMPSocket オブジェクトによって受信した同期情報の解析．

- メディアオブジェクトの操作及び管理
  メディア識別子によって指定されるメディアオブジェクトの生成と管理及び，操作指示情報に基づいたメディアオブジェクトの操作．

メディア間同期モジュールとして SMPScheduler クラスとして実装した．図 5.8に SMPScheduler クラスの定義を示す．

SMPScheduler クラスは通信モジュールからの通知及び各メディアオブジェクトからの通知を受けとる必要があるため，SMPSocketListener クラスと SMPMediaListener クラスからの多重継承として実装した．

SMP 送信ノード上で動作する SMPScheduler オブジェクトと SMP 受信ノード上で動作する SMPScheduler オブジェクトの動作の違いは，メディアオブジェクトの **STATUS** 同期点通知によって生成する同期情報の種類のみである．そこで，本実装では，SMPScheduler クラスのコンストラクタに BOOL 型の引数をとり，同期点通知による動作を指定可能とすることで，SMP 送信ノードと SMP 受信ノードで共通の SMPScheduler オブジェクトを使用した．

5.5.1 基準時間軸の管理

4.8節で述べたように，基準時間軸の管理精度は自律メディア同期機構のメディア間同期精度に影響を与える要素であり，高い精度で管理される必要がある．

本実装では，基準時間軸の管理には WIN32 API の timeSetEvent 関数を使用した．timeSetEvent 関数は定期的に引数に指定するコールバック関数の実行を行なう関数である．timerKillEvent 関数によってコールバック関数の実行を停止する．本実装では，コールバック関数の呼び出し周期に，timeSetEvent 関数で指定できる精度の最大値である $1msec$ を指定した．

SMPScheudler オブジェクトは，基準時間軸上の時刻を管理するための LONG 型の変数 m_Counter を持ち，コールバック関数はこの変数を 1 増加する．さらに，現在の基準時刻を表示予定時刻とした操作情報の有無を m_ScheduledTasks 変数から検索し，該当する情報があった場合は操作指示の実行を行なう．通信モジュールが受信した操作指示情報で指定される表示時刻情報が，基準時間軸の現在時刻よりも以前のものであった場合は，即座に操作指示の実行を行なう．

timeSetEvent 関数は基準メディアオブジェクトの OnPlayed 関数によって実行される．timeSetEvent 関数実行時に基準メディアオブジェクトの GetCurrentPosition 関数を実行し，m_Counter 変数の補正を行なう．また，OnStopped 関数あるいは OnClosed 関数によって timeKillEvent 関数が実行される．

5.5.2 メディアオブジェクトの管理

SMPScheduler オブジェクトでは，全てのメディアオブジェクトの生成及び管理を行なう．メディアオブジェクトは，受信した操作指示情報によって動的に生成されるため，SMPScheduler オブジェクトはメディア識別子に応じてメディアオブジェクトを選択する必要がある．

本実装では，メディア識別子として指定される URL の拡張子を用いてメディアオブジェクトの選択を行なう．メディア識別子の拡張子が”.ppt”の場合は SMPMediaPowerPoint オブジェクトを，”.wmv”あるいは”.mpg”の場合は SMPMediaWindowsMediaPlayer オブジェクトを選択する．

```
class SMPScheduler :
    public SMPMediaListener,
    public SMPSocketListener
{
public:
  SMPScheduler(BOOL fSender = FALSE);
  virtual ~SMPScheduler();

  HRESULT Initialize(HWND hWnd);
  HRESULT UnInitialize();

  void SetSocketObject(SMPSocket *pSocket);
  void AddMediaObject(LPCTSTR mid, SMPMedia *pObject);
  void RemoveMediaObject(LPCTSTR mid);

  // SMPSocketListener クラスの関数
  void OnConnected(int status, LPVOID arg) {}
  void OnSent(int status, int arglen, LPVOID arg) {}
  void OnReceived(int status, int arglen, LPVOID arg);
  void OnDisconnected(int status, LPVOID arg) {}

  // SMPMediaListener クラスの関数
  void OnOpened(LPCTSTR mid, LPVOID arg, BOOL fMaster);
  void OnClosed(LPCTSTR mid, LPVOID arg, BOOL fMaster);
  void OnPlayed(LPCTSTR mid, LPVOID arg, BOOL fMaster);
  void OnStopped(LPCTSTR mid, LPVOID arg, BOOL fMaster);

private:
  SMPMedia* m_MasterMediaObject;
  std::map<std::string, SMPMedia *> m_MediaObjects;
  std::map<LONG, std::string> m_ScheduledTasks;
  LONG m_Counter;
  BOOL m_fSender;
};
```

図 5.8: SMPScheduler クラス

第6章　評価と考察

本章では，5章で述べた自律メディア同期機構 SMP を用いて，本研究が提案するメディア配信機構モデルについて評価を行ない，考察を行なう．

6.1　評価実験の概要

評価を行なうために SMP が動作する送信ノードと受信ノードを使用した．送信ノードと受信ノードはいずれも PC/AT 互換機であり，構成を表 6.1に示す．

表 6.1: 評価実験に使用した計算機の構成

名称 / 項目	送信ノード	受信ノード
CPU	Intel PentiumIII 600MHz	Intel PentiumIII 1GHz
メモリ	640MB	1024MB
OS	Windows XP Professional	Windows XP Professional
NIC	100Mbps Ethernet	100Mbps Ethernet

送信ノードには Windows Media Encoder と PowerPoint が，受信ノードには Windows Media Player と PowerPoint がインストールされている．両ノードはいずれもインターネットに接続している．送信ノードと受信ノードの通信に必要となる IRC サーバは既存のサーバを利用した．

評価実験では本研究の提案するモデルの実用性の検証を目的とし，以下に挙げる 3 つの項目について SMP を用いた評価及び考察を行なった．

1. 基準時間軸の精度
2. メディアオブジェクトの操作処理時間
3. メディア間同期処理の精度

6.2　基準時間軸の精度

メディア間同期モジュールによって管理される基準時間軸は，メディア間同期処理の基準となる時間軸である．よって，基準時間軸の精度は本実装におけるメディア間同期精度に影響を与える要素となる．

本節では基準時間軸管理の精度について評価を行なう．

6.2.1 基準時間軸の進行の精度

5.5.1項で述べたように SMP では基準時間軸の管理に WIN32 API の timeSetEvent 関数を使用する．timeSetEvents 関数は，指定した時間間隔でコールバック関数の実行を行ない，基準時間はコールバック関数によって進行する．SMP ではコールバック関数の実行間隔を $1msec$ に指定した．

SMP の基準時間の値と実際の経過時間を比較することで，基準時間軸の進行の精度を評価を行なった．

実験 1-1 として，受信ノードの SMP 上で SMPScheduler::m_Counter 変数によって管理される基準時間と経過時間の値の比較を行なった．

経過時間の取得には WIN32 API の timeGetTime 関数を用いた．timeGetTime 関数は返り値として OS が起動してからの経過時間を $msec$ 単位で返す．実際には，timeSetEvents 関数を実行する際に開始時刻を取得し，コールバック関数では timeGetTime 関数の返す時間と開始時刻の差分を経過時間として記録した．基準メディアである Windows Media Video の再生を開始してから 5 秒後に PowerPoint の表示を操作指示によって行なった．

実験 1-1 の結果を図 6.1に示す．

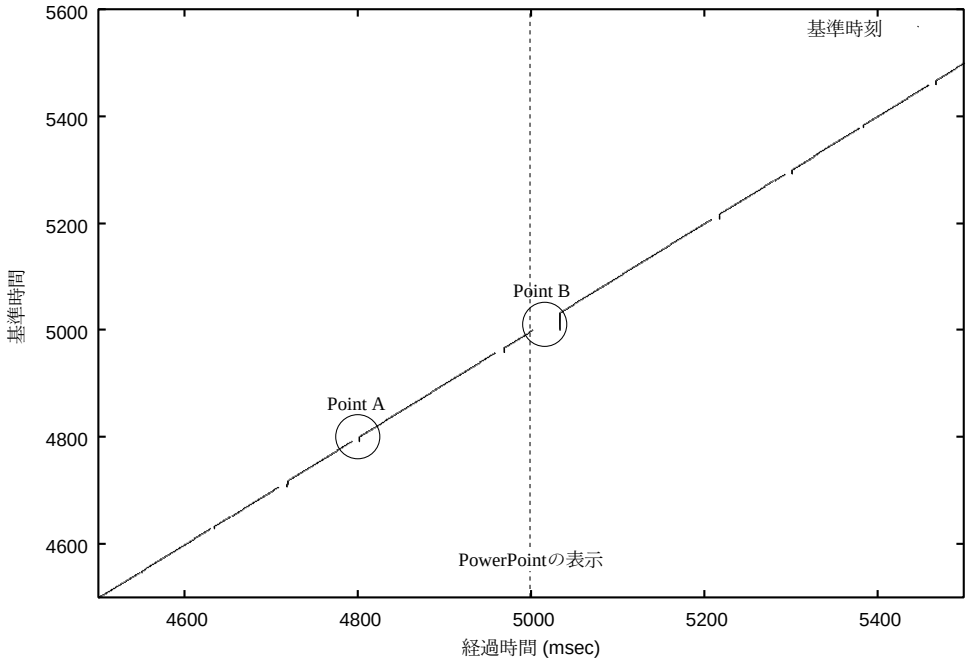

図 6.1: 実験 1-1 の結果:基準時刻と経過時間

図 6.1では，横軸は timeGetTime 関数によって取得される経過時間を表し，縦軸はある経過時間 t における基準時刻の値を表す．

図 6.1の *Point A* に見られる不連続が不定期に発生していることが確認できる．*Point A* では $9msec$ の不連続が発生していた．また，SMPMediaPowerPoint オブジェクトに表示指示を出した $5000msec$ 付近（図 6.1の *Point B*）では，基準時間軸に $32msec$ 分の不連続が生じた．

しかし，不連続が発生した場合は，ある時点で連続的に基準時間の増加が行なわれ，基準時間の概形は経過時間に対して 1 つの直線を描く．

6.2.2 基準時間軸管理の精度

基準時間軸は基準メディアの再生状況に応じて進行することで，基準メディアと他のメディアの時間的相互関係を維持する．

基準メディアである Windows Media Video の再生経過時間と基準時間を比較することで，基準時間軸の管理に関する精度の評価を行なった．

実験 1-2 として，受信ノードの SMP 上で SMPScheduler::m_Counter 変数によって管理される基準時刻と，基準メディアの再生経過時間の比較を行なった．再生経過時間の取得には SMPMediaWindowsMediaPlayer オブジェクトの GetCurrentPosition 関数を使用した．基準メディアの再生を開始してから，Pause 関数を実行した後に再度 Play 関数の実行を行なった．

図 6.2に 1 度基準メディアの再生を停止し，再生が再開された後の結果を示す．

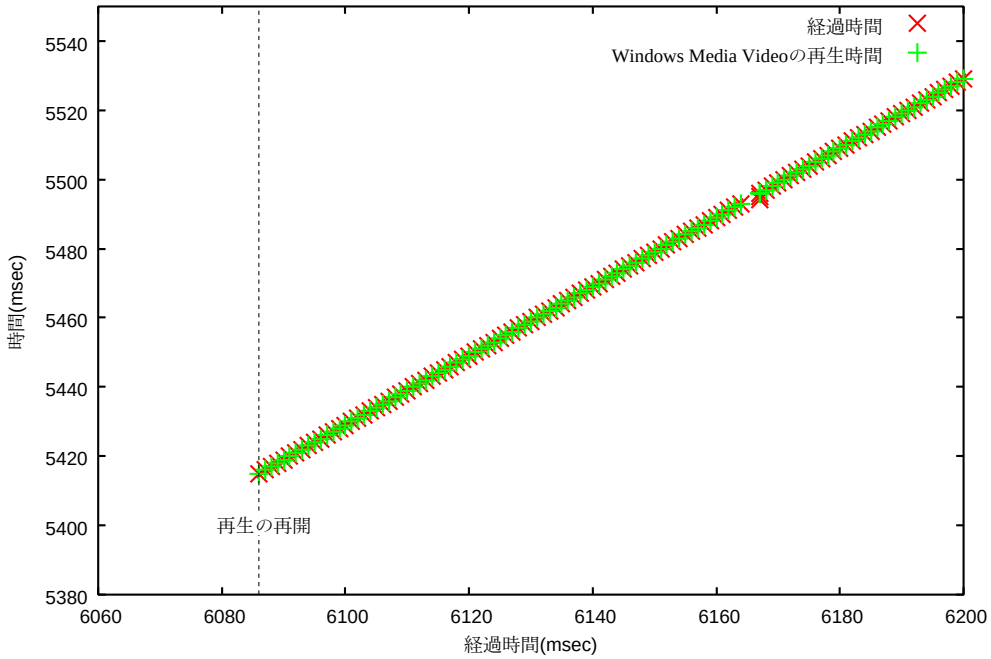

図 6.2: 実験 1-2 の結果:基準時刻と基準メディアの再生時間

横軸は図 6.1と同様の OS の起動経過時間を基にした経過時間を示す．縦軸はある経過時間 t における基準時刻と基準メディアの再生経過時間を表す．

図 6.2より，再生が再開された時点から，2 値は同一線上に描かれていることから，一度 PAUSE 操作によって停止し，再生を再開した場合でも基準時間と再生経過時間は等しくなることがわかった．

6.2.3 基準時間軸の精度に関する考察

実験 1-1 の結果より，基準時間軸の進行では，不定期に $10msec$ 程度の不連続が生じているがわかった．これは，OS のプロセススケジューリングによる誤差だと考えられる．よって，本実装では基準時間軸を $10msec$ 以下の精度では管理できないことがわかった．

また，メディアオブジェクトの操作を行なった際に $30msec$ 程度の不連続が生じた．これはメディアオブジェクトの操作による負荷の影響であることが予想できる．操作処理を行なう際に不連

続が発生するため，30$msec$ 以内に実行される連続的なメディアオブジェクトの操作における同期の実現は困難であることがわかった．

実験 1-2 の結果より基準時間軸は，基準メディアの停止によって進行が停止し，再生開始により進行が再開されていることがわかる．また，基準時間軸として基準メディアの再生経過時間と等しい値が維持されていることから，基準時間軸の管理には正確に基準メディアの再生状況が反映されていると言える．

ユーザの操作によって，30$msec$ 以内に連続的な同期点を生成するのは困難であることが予想できるため，SMP の基準時間軸は 10$msec$ 程度の精度が期待できることがわかった．

6.3 メディアオブジェクトの操作処理時間

メディア間同期処理は，操作指示情報によって指定される表示時刻にメディアオブジェクトを操作することで行なわれる．よって，$PLAY$ 命令を発行してから表示されるまでに要する処理時間はメディア間同期処理の精度に影響を与える要素と言える．

実験 2 として，受信ノードの SMPMediaPowerpoint オブジェクトと SMPMediaWindowsMediaPlayer オブジェクトについて，Play 関数を実行してから OnPlayed 関数が実行されるまでの時間の計測を行なった．メディアオブジェクトの操作は，送信ノードから IRC サーバを介して，操作指示情報を送信することで行ない，連続的に 100 個の操作指示情報を送信した．

実験 2 の結果を図 6.3に示す．図 6.3は横軸に処理時間を $msec$ 単位で表し，縦軸はある処理時間 t で処理が終了した回数を表す．

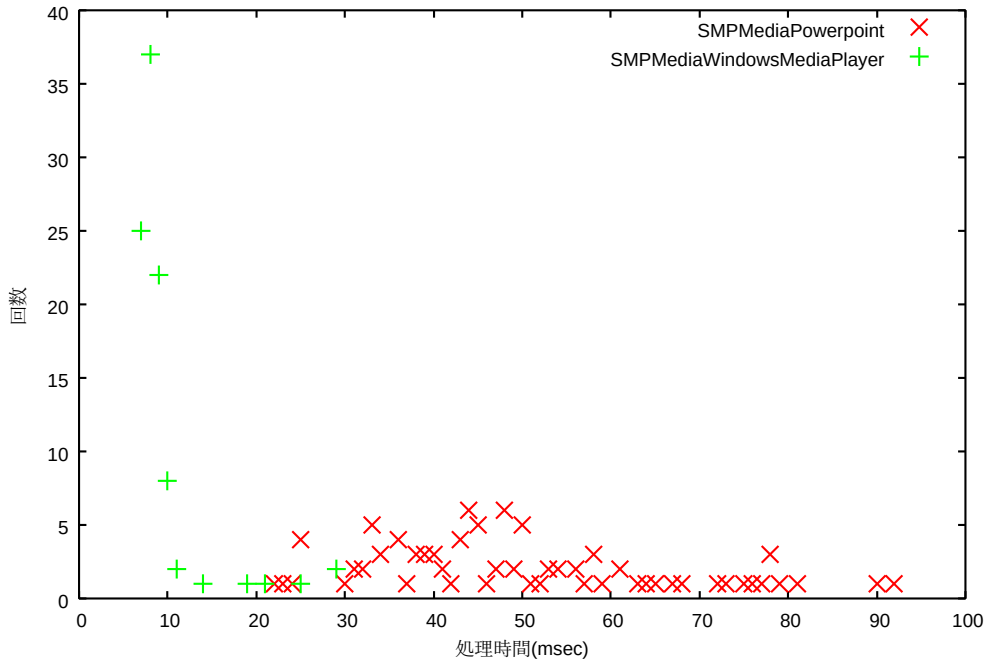

図 6.3: 実験 2 の結果:メディアオブジェクトの PLAY 命令処理時間

図 6.3より SMPMeidaWindowsMediaPlayer オブジェクトの $PLAY$ 処理はほぼ 10$msec$ 前後で完了していることがわかる．しかし，SMPMediaPowerpoint オブジェクトでは $PLAY$ 処理時間に

はばらつきが確認できる．

SMPMediaPowerpoint オブジェクトの表示処理にばらつきが見られるのは表示するスライドに含まれる情報量によって，PowerPoint が必要とする処理が変化するためと考える．

SMPMediaPowerpoint オブジェクト，SMPMediaWindowsMediaPlayer オブジェクトの各 *PLAY* 命令処理時間の最小値・最大値・平均値と平均偏差を表 6.2 に示す．

表 6.2: メディアオブジェクトの PLAY 命令処理時間

処理時間 (msec) / オブジェクト	最小値	最大値	平均値	平均偏差
SMPMediaPowerpoint	22	92	48.02	11.84
SMPMediaWindowsMediaPlayer	7	29	9.08	1.87

PowerPoint の表示処理には平均 50$msec$ 程度の処理時間が必要であり，Windows Media Video の場合には平均 10$msec$ 程度の処理時間が必要となることがわかる．

本研究の提案するモデルでは，メディア間同期精度は個々のメディアオブジェクトの性能に依存するため，メディア間に要求される精度をメディアオブジェクト毎に切り分けることができる．また，要求されるメディア間同期精度はメディアの種類・用途によって異なるため，各メディアオブジェクトの性能として，要求される精度を実現する必要がある．

6.4 メディア間同期精度に関する考察

評価のまとめとして，SMP のメディア間同期精度について考察を行なう．

実験 1-1，実験 1-2 及び実験 2 の結果より，基準メディアは約 10$msec$ の精度で管理され，PowerPoint の表示処理には Play 関数の実行から 20$msec$ から 90$msec$ の処理時間を要する．よって，今回実装を行なった SMP では，約 100$msec$ の精度で Windows Media Video と PowerPoint の表示を同期できることがわかった．100$msec$ という同期精度は，講義中継における資料の同期に要求される精度を実現できる精度であると考える．

また，5.3.2項で述べたように，本機構では基準メディアのメディア表示位置情報は，Windows Media Video のデータストリームに埋め込まれるため，配信形態に関わらず同様の精度が期待できる．

以上の結果より，本研究で提案する自律メディア同期モデルは，メディアの形式，配信方法に依存せずに複合メディアコンテンツの実時間配信環境に構築に対して有効であると言える．

第7章　研究のまとめと今後の課題

本章ではまとめとして本研究の成果を明らかにし，今後解決すべき課題について述べる．

7.1　結論

本研究では，さまざまなメディアを自由に組み合わせて配信を行なう環境の実現を目指し，メディアの種類，形式及び配信方法に依存せずに，動的にメディア間の時間的相互関係を定義できる自律メディア間同期処理モデルの提案を行なった．

実時間配信における動的なメディア間の時間的相互関係の定義には，複合メディアを構成する多様なメディアに対して，同期処理の基準となる同期点の定義が可能であることと，受信側からの試聴状況のフィードバックが必要であることを述べ，同期情報の記述方法を提案した．

また，複合メディアについて，構成するメディアの特性，配信処理について検討し，メディア間同期処理の必要性を明らかにした．

メディア毎に配信形態及びメディア内同期処理が異なること，処理時間が受信者の環境毎に異なることに着目した．前者の問題に対してはメディア間同期処理と各メディアの配信及びメディア内同期処理を切り分けること手法を提案した．後者に関しては受信ノードがメディア間同期処理の基準時間軸を自律的に管理する手法を提案した．

メディア間同期処理の切り分けを行なうために，メディアの抽象化を行ないメディアオブジェクトを定義した．メディアオブジェクトを用いることで，メディアの種類・配信形態に依存せずに，メディアの表示操作及びメディア間同期処理に必要となる情報の取得が可能になることを述べ，メディアオブジェクトの定義によって，複合メディアコンテンツを構成するメディアの多様化に対応した．

また，メディアオブジェクトの定義に基づき，メディアの操作指示及び試聴状況を伝える同期情報の記述方法を定義した．

基準時間軸の管理では，複合メディアを構成するメディアの 1 つを基準メディアとし，基準メディアの再生状況に応じた基準時間軸の管理を行なう手法を示した．本モデルによって，受信者毎に異なるメディアの再生状況の差異を吸収できると述べた．

本モデルに基づいて自律メディア同期機構 SMP の実装を行なった．SMP は講義中継を対象としたメディア間同期機構であり，PowerPoint スライドショーと Windows Media Video のメディア間同期処理を行なう．本実装では，メディアオブジェクトの抽象化の実現方法として，C++言語の継承クラスによる実装方法を示した．また，COM 技術を用いて既存のアプリケーションの利用を行なうことにより，多様なメディアへの拡張性を確保した．

評価の結果，基準メディアは約 $10msec$ の精度で管理され，講義資料の処理時間を考慮しても，約 $100msec$ の精度で Windows Media Video と PowerPoint の表示を同期できることがわかった．本実装により，本機構がメディアの種類，配信方法に依存せずに動的に同期点を生成でき，また，生成された同期点にしたがってメディアが表示できることを示した．

本機構は，メディアの形式，配信方法に関わらず，動的にメディア間の時間的関係を定義し，定義された時間関係にしたがった同期を実現し，また，フィードバック情報によって，送信者による複数の受信環境における試聴状況の把握を可能とする，複合メディアコンテンツの実時間配信環境を実現した．

よって，本論文で提案した手法を用いれば，さまざまなメディアを自由に組み合わせて配信する環境を実現できると言える．

7.2 今後の課題

本研究の今後の課題について述べる．

7.2.1 多対多の配信形態への対応

本研究では，多対多の配信モデルについて検討を行なわなかった．例えば，遠隔会議では複数の参加者が意見を述べ，資料の提示を行なう．

このような，遠隔会議において，議論を円滑に進め，且つ試聴状況の同期を行なうには，本研究で提案したモデルの拡張及び遠隔会議の進行方法について検討を行なう必要があると考える．

前者については，受信者の環境に応じたメディアの選択，同期情報生成のチャンネル化が挙げられる．

受信者の環境に応じたメディアを選択することで，遅延を減少させることができる．RealSystem や Windows Media Technology では，複数の帯域を対象としたコンテンツを用意することでクライアントがネットワーク環境に応じてコンテンツの選択を行なう機構が実装されている．同一のメディア識別子から，受信環境に応じたメディアを選択できる機構を実現することで，システムの枠を越えたメディアの選択が可能になると考える．

また，遠隔会議では全ての参加者が受信者であり，送信者となるため，操作指示情報を生成するチャンネルとフィードバック情報を生成するチャンネルを区別する仕組みが必要となる．

後者の遠隔会議の進行方法については，例えば会議の発表と議論の時間を明確に区別し，それぞれの用途にあった基準メディアの切替えを行なう手法の提案を検討している．

7.2.2 空間的関係の解決

2.4.1項で複合，メディアを構成するメディア間の関係には時間的相互関係と空間的相互関係があることを述べたが，本研究では空間的相互関係について解決を行なわなかった．

動的にメディア間の空間的相互関係の定義を可能にするには，出力デバイスに対する (x, y, z) 座標点，幅，高さなどの空間的関係の抽象化が必要である．

また，位置情報の操作は受信者のデバイス環境に依存するため，この差異について検討を行なう必要がある．

7.2.3 拡張性に関する課題

SMP ではメディアオブジェクト及び通信モジュールの実装に C++言語のクラスの継承を用いることで，メディアの種類・通信方法に対するソースレベルの拡張性を確保した．

しかし，ソースレベルの拡張性ではメディアオブジェクトを追加する度に実行ファイルのコンパイル処理が必要となるため，拡張性として不十分と考える．よって，バイナリレベルの拡張性を確保する必要がある．

バイナリレベルの拡張性を得るために，メディアオブジェクト及び通信モジュールをライブラリあるいはプラグインとして実装し，SMP の実行ファイルとの分離を行ない，動的に必要な機能が SMP に組み込まれる機能を実装する必要がある．

また，動的にメディアオブジェクトが組み込まれる場合，受信環境毎に使用できるメディアオブジェクトの構成が異なることを予測し，メディア識別子と対象のメディアを操作するメディアオブジェクトとの対応を動的に管理する方法について検討を行なう．

謝辞

本研究を進めるにあたり，ご指導頂きました慶應義塾大学環境情報学部教授の村井純博士，稲蔭正彦氏，徳田英幸博士，同学部助教授の楠本博之博士，中村修博士，慶應義塾大学大学院政策・メディア研究科助教授の大川恵子博士，同大学環境情報学部専任講師の重近範行氏，南政樹氏に感謝します．

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氏，慶應義塾大学大学院政策・メディア研究科後期博士課程の村上陽子氏に感謝します．

本研究に取り組む契機を与えて下さいました WIDE Project SOI WG の諸氏，実装の折にご助言頂きました株式会社 KeelNetworks の諸氏に感謝します．そして，本研究を進めていく上でさまざまな励ましとご助言をして下さった，慶應義塾大学大学院政策・メディア研究科後期博士課程の小川晃通氏，今泉英明氏，頴原桂二郎氏，同大学環境情報学部の入野仁志氏，工藤紀篤氏，同大学総合政策学部の堀場勝広氏，ならびに同大学の徳田・村井・楠本・中村・南研究室の諸氏に感謝します．

また，本稿を書き進めるにあたり，STREAM KG の諸氏には絶えず励まして頂きました．ここに深い感謝の念を表します．

参考文献

[1] "HTML 4.01 Specification", W3C Recommendation, 24, December, 1999

[2] Keiichi Kawai, Keiko Okawa, Jun Murai, "Practical Experiences of Higher Education on the Internet -Cases from the School of Internet-", Proc. of ICCC'99, Tokyo, Sep, 1999

[3] "Synchronized Multimedia Integration Language (SMIL 2.0)", W3C Recommendation, 7, August, 2001

[4] Macromedia Flash MX, http://www.macromedia.com/software/flash/

[5] クスタルト ウィドヨ、杉浦 一徳、村井 純、『広域ネットワークにおけるマルチメディアの同期機構に関する研究』、情報処理学会マルチメディア通信・分散処理研究会、1997 年 11 月

[6] RealNetworks, http://www.realnetworks.com/

[7] Windows Media Technology, http://www.microsoft.com/windowsmedia/

[8] D.Estrin, D.Farinacci, A.Helmy, D.Thaler, S.Deering, M.Handley, V.Jacobson, C.Liu, P.Sharma, L.Wei"Protocol Independent Multicast-Sparse Mode(PIM-SM):Protocol Specification", RFC2362, June, 1998

[9] J. Postel, "User Datagram Protocol", RFC768, Aug 1980

[10] J. Postel, "Transmission Control Protocol", RFC793, Sep 1981

[11] H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", RFC1889, January 1996

[12] D. Bryant, P. Brittain, "APAN Implementer's Work Shop Closed Document DLSw v2.0 Enhancements", RFC2166, June 1997

[13] OpenDML AVI M-JPEG File Format Subcommittee, "OpenDML AVI File Format Extensions", Febuary, 1996

[14] "Timed Interactive Multimedia Extensions for HTML (HTML + TIME) Extending SMIL into the Web Browser", W3C Submission Request, 18, Sep, 1998

[15] Miguel Correia, Paulo Pinto, "Low-Level Multimedia Synchronization Algorithms on Broadband Networks", ACM Multimedia, 1995

[16] Allen J., "Maintaining Knowledge about Temporal Intervals", Comm. ACM, Vol.26, No.11, pp.832-843, 1983

[17] Masunaga Y., "An Object-Oriented Approach to Temporal Multimedia Data Modeling",Proceedings of the IEICE Transactions on Information and Systems, Vol.E78-D, No.11, pp.1477-1487, November, 1995

[18] J. Schnepf, J. A. Konstan, and D. H.-C. Du. "Doing FLIPS: FLexible Interactive Presentation Synchronization," IEEE Jounal on Selected Areas in Communications, vol. 14, no. 1, pp. 114125, 1996. http://citeseer.nj.nec.com/schnepf96doing.html

[19] 端山 貴也，清木 康,『協調型同期方式によるメディア間同期とその実現方式』，情報処理学会論文誌，Vol.39, No.3，1998 年 5 月

[20] 小川 浩司，櫻井 智明，大川 恵子，村井 純,『インターネットを利用したリアルタイム中継における資料共有システムの設計と実装』，第 61 回全国大会講演論文集，2000 年 10 月

[21] Microsoft PowerPoint, http://www.microsoft.com/office/powerpoint/default.asp

[22] Bailey, J. A. Konstan, R. Cooley, and M. Dejong. "Nsync - A Toolkit for Building Interactive Multimedia Presentations", Proc. of ACM Multimedia'98, Bristol, England, pp. 257-266. 1998. http://citeseer.nj.nec.com/bailey98nsync.html

[23] Alexandre R.J.Francois，Gerard G. Medioni，"A Modular Middleware Flow Scheduling Framework"，ACM Multimedia Conference，2000

[24] T. L. Disz, I. Judson, R. Olson, R. Stevens, "The Argonne voyager multimedia server", presented at IEEE Computer Society 6th High Performance Distributed Computing, Portland, OR, 1997

[25] Koji Ogawa, Kazunori Sugiura, Masahiko Inakage, Osamu Nakamura, Jun Murai, "The Implementation of Remote Shooting System for Digital Cinema", International Conference on Telecommunications 2002, June, 2002

[26] T.Berners-Lee，"Uniform Resource Identifiers (URI):Generic Syntax"，RFC2396，1998

[27] Microsoft Corporation and Digital Equipment Corporation, "The Component Object Model Specification Draft Version 0.9", 24, Oct, 1995

[28] E.Gamma, R.Helm, R.Johnson, J.Vlissides, "Design Patterns Elements of Reuseable ObjectOriented Software", Addison Wesley, 1994

[29] C.Kalt, "Internet Relay Chat: Architecture", RFC2810, April, 2000

[30] C.Kalt, "Internet Relay Chat: Client Protocol", RFC2812, April, 2000

[31] Dabid L. Mills, "Network Time Protocol (Version 3) Specification, Implementation and Analysis", RFC1305, March, 1992