平成 24 年度 修士論文

ブラウザを用いた複数ユーザ動画制御共有ツール

電気通信大学 大学院情報システム学研究科
情報システム基盤学専攻
1153016 西尾 信吾

指導教員　末田 欣子　客員准教授
多田 好克　教授
大森 匡　教授

提出日　平成 25 年 1 月 23 日

目次

図目次

表目次

ソースコード目次

第 1 章

序論

近年では，インターネットの普及により画像や音声，動画，ドキュメントなどの様々なデジタルコンテンツを共有できるようになった．特に，動画の共有は YouTube[3] や，ニコニコ動画 [5] などの動画投稿サイトの台頭により，様々な人々と動画を共有することが一般的になった．しかし，これはリアルタイムに同じ動画を視聴するのではなく，単に違う時刻に同じ動画を見ているだけである．リアルタイムに同じ動画を見る形式として，USTREAM[7] やニコニコ生放送 [6] などのライブ配信型の動画投稿サイトがあげられる．ライブ配信型の動画では，リアルタイムに同じ動画を見ることができる一方で，視聴している動画に対し，自由な時刻に動画の再生を開始することや，自由に再生停止や再生位置変更などの操作を行うことはできない．

デジタルコンテンツの共有をリアルタイムに行う研究やサービスが幾つか行われている．専用のソフトウェアのダウンロードやインストールを必要としているものや利用者の負担を削減した一般的な Web ブラウザを利用したリアルタイムのデジタルコンテンツの共有システムが提案されている．

本稿では，一般的な Web ブラウザを利用して，遠隔利用者間でリアルタイムに動画を共有し，動画に対し全ての利用者が再生停止や再生位置の変更を可能にすることで，利用者のコミュニケーションを支援する動画制御共有システムの提案を行う．

本システムでは，JavaScript を用いた実装により「リアルタイム操作の達成」，サーバ側での逐次処理による「複数の利用者による操作を可能にする」，一般的な Web ブラウザのみによる実現で「利用者の負担を低減する」，利用者間で変更される度に動画の再生時刻を同期させることで「同期のずれを許容範囲に抑える」の 4 つ「」で示した要求条件を実現する設計と実装を行った．

そして，本システムの利用者が増加した際に，共有サーバで本システムがどの位のメモリ，CPU を利用するかの測定や，本システムがどの位の利用人数まで耐えうるかの負荷実験を行った．また，視聴部屋内の利用者の一人のネットワークに遅延を掛けた場合に，どれほど動画の再生共有に影響が出るのかの測定を行う遅延実験による利用者間の動画の同期のずれが大幅なネットワーク遅延がない限り，2 秒以内に抑えられるということを幾つかの利用例に基づき示した．

本論文の構成を以下に示す．第 2 章で背景を述べ，第 3 章で関連研究について述べる．第 4 章で本システムの提案方式を説明し，第 5 章で設計を行う．第 6 章で本システムの実装を説明し，第 7 章で実験と評価を説明する．そして，第 8 章で結論と今後の課題について述べしめる．

第 2 章

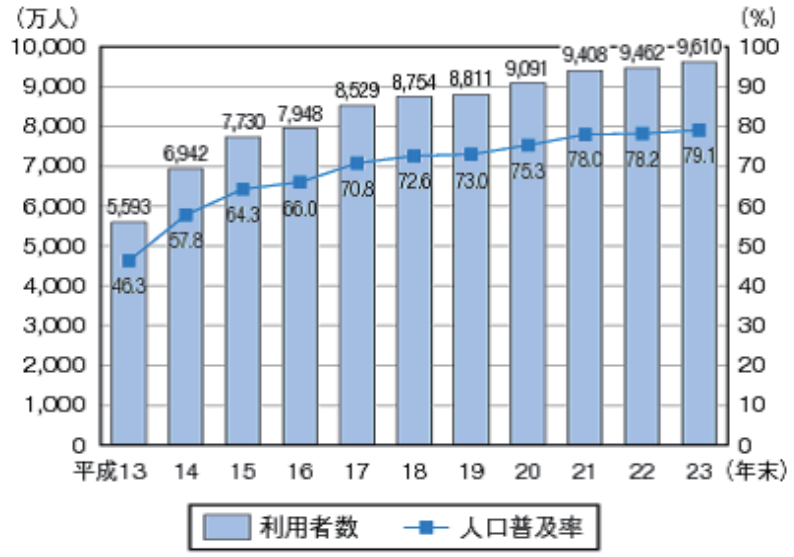
背景

この章では，インターネットや情報通信機器の普及に伴うコンテンツ共有の普及について述べる．そして本研究の目的を述べ，既存のシステムの問題点を挙げる．最後に本システムの利用例を説明する．

2.1 コンテンツ共有

図 2.1 に示すように，23 年度末には日本人の 6 歳以上の約 80% がインターネットを利用するまでに普及してきた [1]．インターネットの利用方法は，図 2.2 に示すように様々なものがある．そしてそれらの利用方法では，ブロードバンドはナローバンドと比べ，音楽・音声，映像，ゲームなどのデジタルコンテンツの入手・視聴や動画投稿サイトの利用の割合が増加している．それは，ネットワークのブロードバンド化に伴い，これらの様々なデジタルコンテンツに対してアクセスが容易な環境になったからだと考えることが出来る [2]．そして，さらなる PC(Personal Computer) の性能の向上や，インターネット回線の高速化・大容量化に伴い，ゲームなどの Web アプリケーションの様な多くの帯域幅を要求するデジタルコンテンツを共有できるようになる．

それらのデジタルコンテンツの共有の中で，動画共有サイトを利用した動画の共有が重要なものとなってきている．図 2.3 に示すように，10 代〜60 代の全体で約 40% の人々が月に数回以上動画共有サイトを利用している [2]．さらに，10 代では約 70% もの人が月に数回以上動画共有サイトを利用している．そして，月に数回以上動画共有サイトを利用している人の半数が趣味・娯楽としてのインターネットを重視しており，動画共有サイトの利用が重要な娯楽の一つとなってきている．



※ 調査対象年齢は６歳以上。
※ インターネット利用者数（推計）は、６歳以上で、調査対象年の１年間に、インターネットを利用したことがある者を対象として行った本調査の結果からの推計値。インターネット接続機器については、パソコン、携帯電話・PHS、スマートフォン、タブレット端末、ゲーム機等あらゆるものを含み（当該機器を所有しているか否かは問わない。）、利用目的等についても、個人的な利用、仕事上の利用、学校での利用等あらゆるものを含む。
※ インターネット利用者数は、６歳以上の推計人口（国勢調査結果及び生命表等を用いて推計）に本調査で得られた６歳以上のインターネット利用率を乗じて算出。
※ 無回答については除いて算出している。

図 2.1 我が国のインターネット利用者数及び人口普及率の推移 (文献 [1] より抜粋)

10ポイント以上の差

機能・サービス	ブロードバンド	ナローバンド	全体	ブロードバンドとナローバンドの差
商品・サービスの購入・取引	52.5%	36.8%	45.1%	15.7ポイント
デジタルコンテンツ（音楽・音声、映像、ゲームソフト等）の入手・聴取	28.3%	16.3%	23.6%	12.0ポイント
動画投稿サイトの利用	28.4%	17.0%	23.6%	11.4ポイント
地図情報提供サービス（有料・無料を問わない。乗換案内、ルート検索サービスも含む）	34.8%	26.2%	30.2%	8.6ポイント
インターネットオークション	17.7%	10.9%	14.8%	6.8ポイント
オンラインゲーム（ネットゲーム）への参加	9.7%	5.3%	8.0%	4.4ポイント
ソーシャルネットワーキングサービス（SNS）への参加	7.1%	3.4%	5.8%	3.7ポイント
電子ファイルの交換・ダウンロード（P2P、FTPなど）	9.3%	5.7%	7.8%	3.6ポイント
マイクロブログの閲覧・投稿	4.5%	3.7%	3.8%	0.8ポイント

※ 複数回答あり

図 2.2 ナローバンド・ブロードバンド別、利用した機能・サービスと目的・用途の比較 (文献 [2] より抜粋)

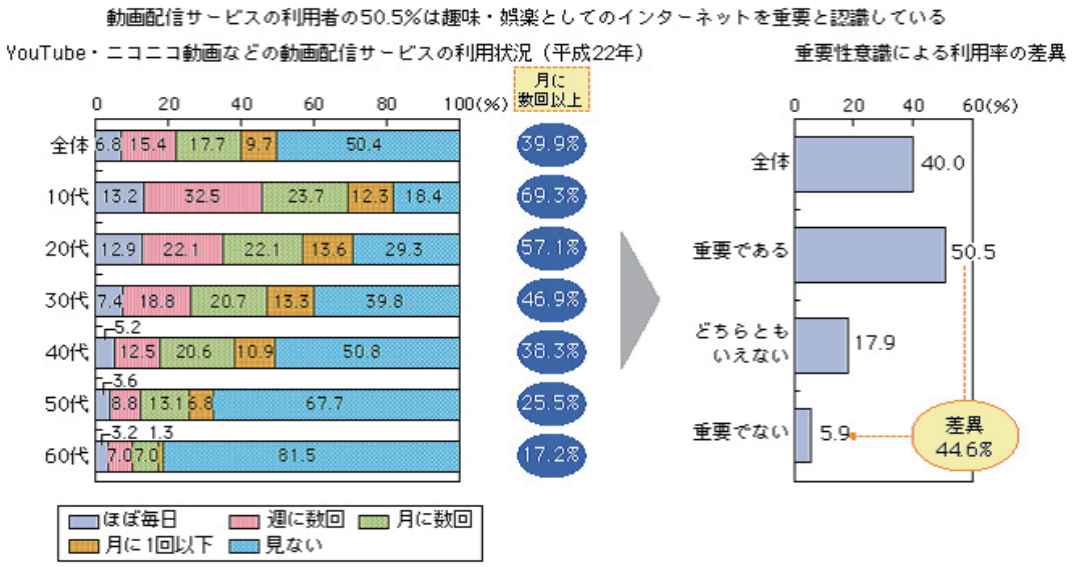


図 2.3　動画配信サービスの利用状況と、趣味・娯楽としての重要性意識による利用率の差異 (文献 [2] より抜粋)

2.2　目的

本稿では，動画を用いたコミュニケーションを円滑に行うため，動画を自由に再生停止や再生位置変更ができ，なおかつ，多くの利用者が同一の動画の同じ再生箇所を共有し，視聴できるようなシステムを提案し，実現する．リアルタイムに動画の再生時刻を共有すること (以下，再生共有) は，スポーツの動画などを視聴している時の，得点の入る瞬間などを繰り返し視聴しながら議論を交わすことなどを， 遠隔地にいながら可能にする．このシステムを用いることで使用者の一人が再生位置の変更を行なっても即座に他の使用者の動画に反映されることでリアルタイムに同じ再生箇所を視聴しながらコミュニケーションを行うことが出来る．

以下では，まず既存のシステムを概観し，その問題点を考える．そして，提案システムの利用例を示す．

2.3　投稿型動画共有サイト

Web2.0 の発祥年である 2005 年に登場した，Google が提供している動画投稿サイトの YouTube[3] により，動画をサイトに投稿することで，遠くに離れた様々な人々と同じ動画を共有できるようになった．YouTube は世界で月 8 億人に閲覧されており，また日本

国内でも図 2.4 に示すように，平成 23 年 2 月には 28,986 千人が利用している世界を代表するインターネットサービスである [2, 4].

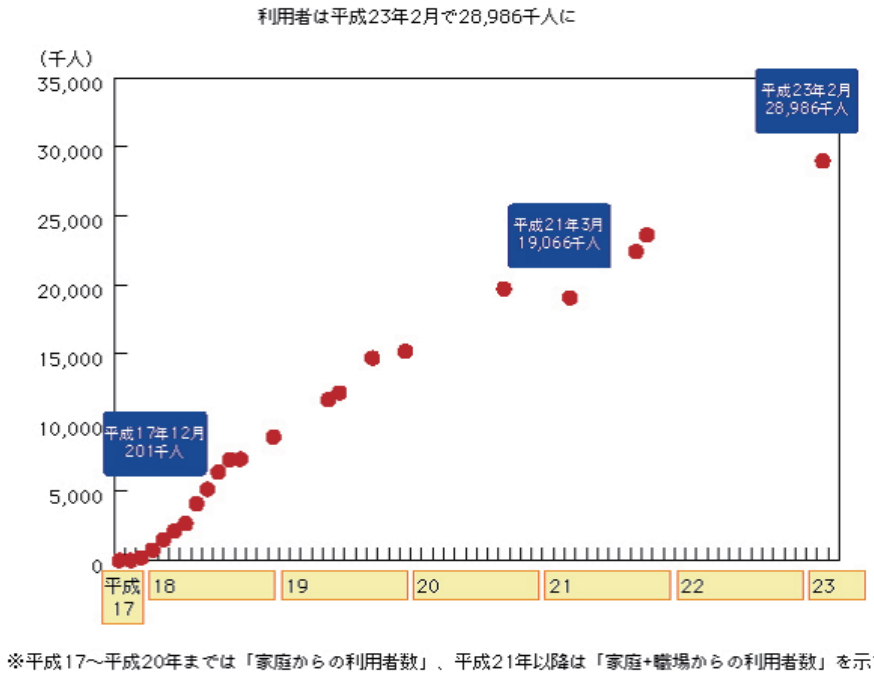


図 2.4　動画配信サービス（YouTube）利用者数の推移 (文献 [2] より抜粋)

YouTube のサイト上や，YouTube の動画が埋め込まれた Web サイトやブログ上で，視聴者は投稿された動画を容易に全世界で視聴することが可能である．そこで視聴者は TV 番組を録画したビデオを視聴するように，視聴している動画に対し自由に再生停止や再生位置の変更を行い，自分の見たいシーンや見逃したシーンを何度も繰り返し視聴を行うことも可能である．また録画されたビデオを視聴する場合とは異なり，投稿された動画に対しての感想や意見などを，動画の下のコメントフォームに投稿することで他の視聴者と，動画に対する議論などのコミュニケーションを行うことが出来る．

また日本国内での同様のサービスとして，ドワンゴが提供しているニコニコ動画 [5] がある．ニコニコ動画では Youtube と同様に動画の閲覧が可能である．YouTube と異なる点として投稿されたコメントが，動画の視聴中に画面上に表示される点である．すべての視聴者がその動画を視聴するたびに，動画の再生位置に応じて，投稿されたコメントが画面上に表示される．

しかし図 2.5 のイメージ図に示すように，これらの動画共有サイトのでの “共有” は，複数の人々が同時に同じ動画を視聴するということではなく，任意の時刻で同じ動画を様々な視聴者が視聴可能であるということである．

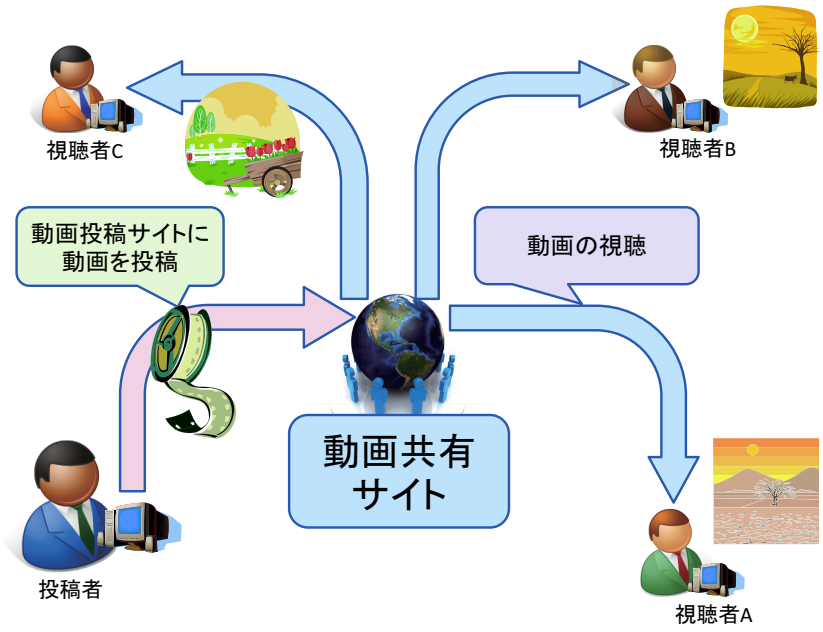


図 2.5　投稿型動画共有サイトのイメージ

2.4　ライブ配信型動画共有サイト

投稿型動画共有サイトと異なり，リアルタイムに複数の視聴者と同じ時刻で同じ場面を見る動画共有サイトの形式として，一般的に TV 放送のようにライブ中継を行う形式 (以下ライブ配信と省略) でのストリーミング放送が挙げられる．ライブ配信型の動画共有サイトは，2007 年の USTREAM[7] を筆頭に出現してきたサービスである．そしてその他にも，同 2007 年のニコニコ動画で提供されているサービスの一つであるニコニコ生放送 [6] や，2011 年の YouTube のサービスの一つである YouTube Live など，様々なライブ配信型の動画共有サイトが存在する．これらのライブ配信型動画共有サイトの登場により，従来はテレビ事業者などの限られた人々しか行えなかったライブ配信が，一般の人でも容易に行うことが可能になった．

図 2.6 のイメージ図に示すように，これらのライブ配信型の動画共有サイトでは，投稿された動画ではなくリアルタイムに放映されている動画の視聴を行う．放送者は動画の放送を行う時刻を予め予定し，その予定した時刻に動画の放送を開始する．視聴者は放送者が予定した時刻以降に，動画共有サイトの放送が行われるページにアクセスを行うことで動画の視聴を行うことが出来る．

また，リアルタイムにライブ中継されているので，投稿されたコメントに対し放送者がリアルタイムに反応することも可能である．

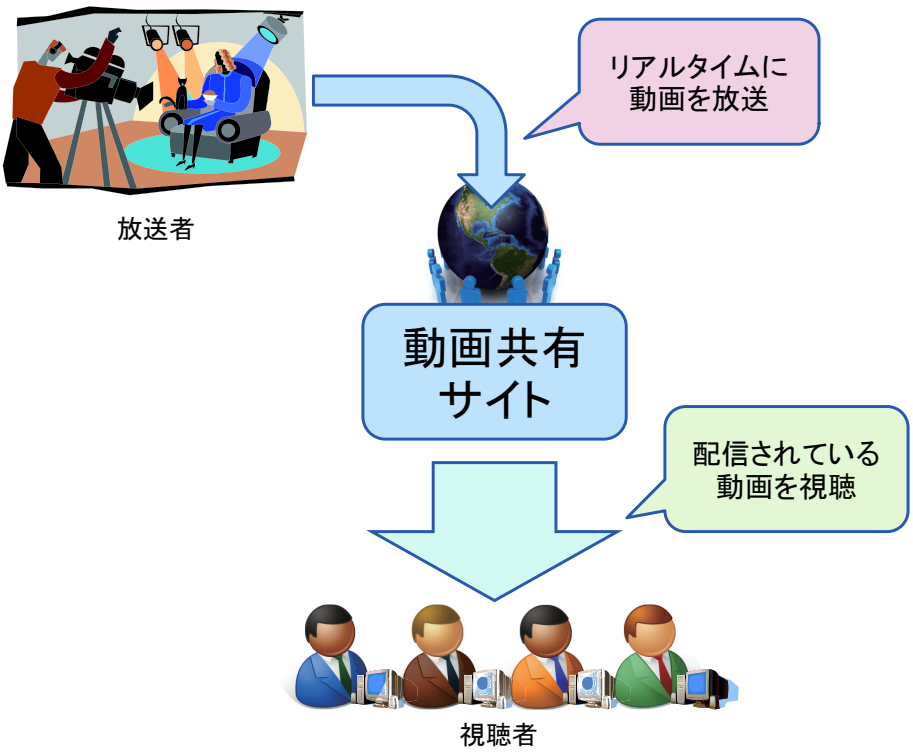


図 2.6　ライブ配信型動画共有サイトのイメージ

2.5　問題点

投稿型動画共有サイトとライブ配信型動画共有サイトでは，以下の様な点が動画を用いたコミュニケーションを行う上で問題となる．

- 投稿型動画共有サイト
  この場合では，複数の人々が同じ時刻に同じ動画を視聴するのではなく，任意の時刻で同じ動画を様々な視聴者が視聴する．よって，以下のような点が問題となる．
  - 視聴者が他の視聴者に対し視聴して欲しい動画の場面を伝えるのに，動画の再生時刻やどのような場面かを相手に伝えなければならない．
  - 伝えられた視聴者自身が，動画プレイヤーを操作し伝えられた再生時刻に合わせなければならない．
  - 再生時刻やどのような場面かの伝達に不備があった場合に，コミュニケーションに齟齬が発生する可能性がある．
- ライブ配信型動画共有サイト
  この場合では，動画がライブ形式で配信されているため，特定の時刻で同時に同じ

動画をすべての視聴者が視聴する．よって，以下のような点が問題となる．

- 投稿型動画共有サイトの動画と異なり自由に再生停止や再生位置の変更を行うことができない．
- リアルタイムの配信のため，指定された時刻にしか動画を視聴することができない．
- 再度同じ場面を視聴することができない．

2.6 利用例

本システムの利用イメージを，図 2.7 に示す．図 2.7 の左側の場面では，4 人の人が一緒に同じ場所で一つの画面に映る動画を視聴しながら，動画についての会話を行なっている．この動画を視聴している 4 人の各々が，再度視聴したい場面に再生位置の変更操作を行うことや，特定の場面を止めて見たい場合に動画を一時停止したり，視聴を再開するために再生することが可能である．

本システムでは上記の場面を，インターネット上の分散された環境で実現する．上記の場面で，一緒に同じ場所で一つの画面に映る動画を視聴していた利用者は，それぞれが離れた場所でコンピュータを利用している．その各々が一般的な Web ブラウザを利用し本システムにアクセスすることで，各々の Web ブラウザに同じ動画が表示される．そして，各利用者が動画に対し再生停止や再生位置変更の操作を行うと，上記の場面の様にあたかも全ての利用者で同じ動画を見ているように，動画の再生停止の状態や再生位置が変更される．

以下で本システムの利用例として，「娯楽の場面での利用」と「教育の場面での利用」の 2 つの場面を示す．

図 2.7　提案システムの利用イメージ

2.6.1 娯楽の場面での利用

娯楽の場面での利用の例として，図 2.8 のような複数の友人間での野球の動画の視聴を示す．複数の友人たちと視聴開始の時間を決め，インターネット上の本システムにアクセスする．そして，本システム上で野球の動画の視聴する．チャットなどのコミュニケーションツールを用いて利用者間でコミュニケーションを行う．利用者の一人が再生開始を行うことで，全ての利用者で再生が開始される．例えば、野球のコンテンツの場合、ある一人の利用者が他の利用者と，バッターのバッティングフォームをもう一度視聴し，それについてコミュニケーションを行いたい時に再生位置の変更する．そうすると，全ての利用者の動画の再生位置が，その動画の再生位置を変更した利用者の再生位置に移動する．そして，チャットなどのコミュニケーションツールを用いて，その場面についてのコミュニケーションを行う．

また，ある利用者がバッティングフォームを止めて視聴したいときに，一時停止を行うことで他の利用者の動画も一時停止状態に変更される．そしてその場面についてのコミュニケーションが終了した際に，再生ボタンを利用者のうちの一人が押すことで動画の再生が再開される．

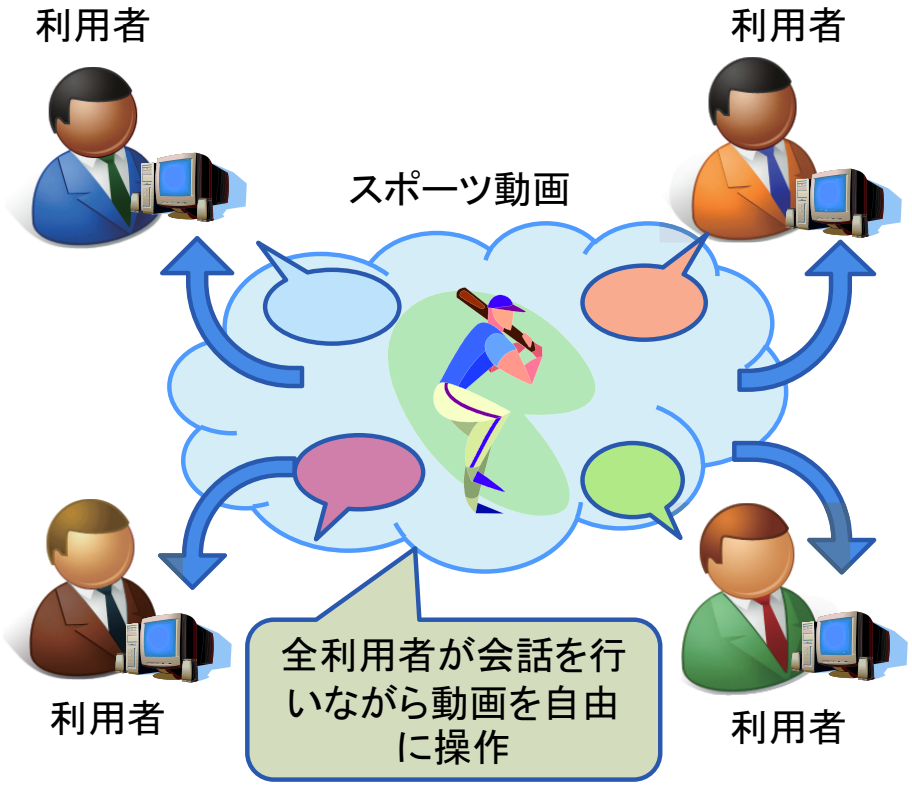


図 2.8　娯楽の場面での利用のイメージ

2.6.2 教育の場面での利用

教育の場面での利用の例として，図 2.9 のような複数の生徒と教師間での教育用の動画の視聴を示す．ある教師が，動画を用いた授業を行うために本システムにアクセスする．そして授業開始の際に生徒たちも本システムにアクセスする．教師は動画を用いた授業を開始するために，動画の再生を行う．そうすると，全ての生徒の動画も同様に再生が行われる．そして，教師は生徒に注視して欲しい場面で動画を停止すると，同様に全ての生徒たちの動画も停止される．強調したい部分を再度再生するときには，教師が再生位置の変更を行うことで全生徒の動画の再生位置を変更する．

また，ある一人の生徒が動画の特定の場面で教師に対し質問を行いたい時に，その生徒が動画の再生位置の変更を行い，教師と他の生徒の再生位置を変えてその場面について質問を行うことが出来る．

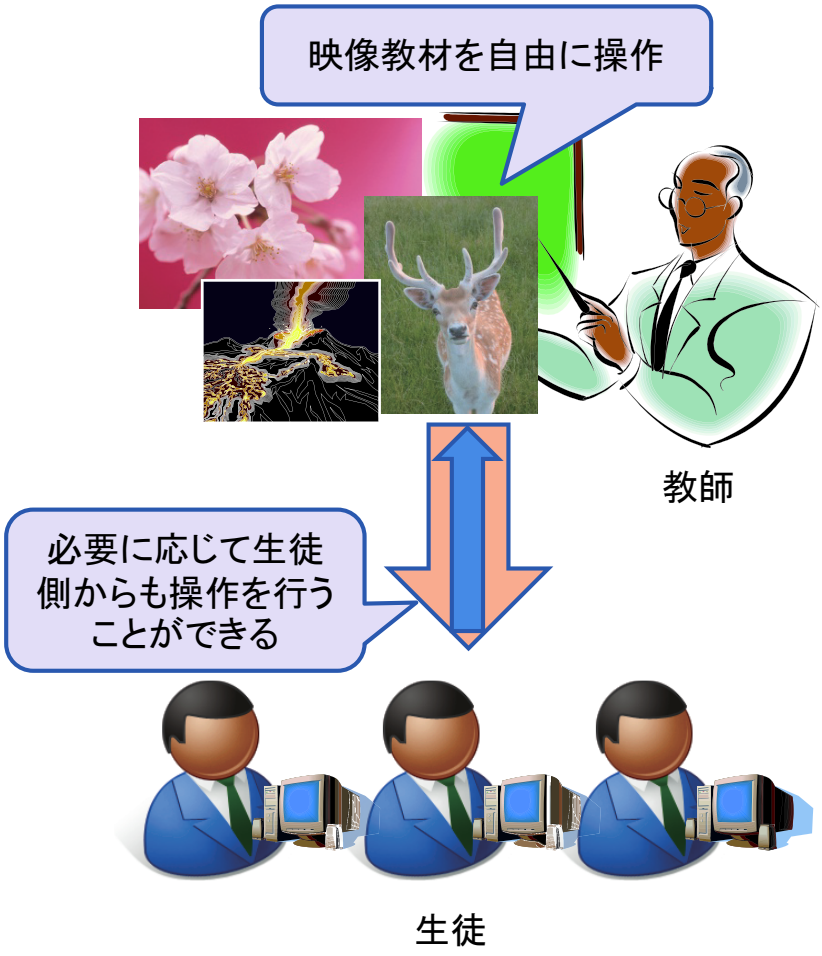


図 2.9 教育の場面での利用のイメージ

第 3 章

関連研究およびサービス

この章では，本研究に関連する既存研究やサービスについて述べる．まず，様々なデジタルコンテンツの共有を行う，専用ソフトウェアを用いた共有と Web ブラウザのみを用いた共有に関連した既存研究及びサービスについて述べる．そして，動画の再生共有を行う既存研究について述べ，これらの関連研究より課題を抽出する．

3.1 専用ソフトウェアを用いたデジタルコンテンツのリアルタイム共有

専用ソフトを用いたデジタルコンテンツのリアルタイム共有を行うサービスや研究は以前から数多く存在する．これらのアプリケーションを利用するために，利用者は専用のソフトウェアのダウンロードやインストールを必要とする．またこのようなネットワークを介するようなアプリケーションは，特別な通信設定を要する場合がある．

代表的なサービス及び幾つかの研究とその内容を以下に示す．

3.1.1 Skype

Skype[8] は，2003 年に登場した，2012 年 6 月時点で世界で 2 億 5000 万人もの人が利用するインターネット通話サービスである [9]．このサービスを利用することで，このアプリケーションの使用者同士でインターネット回線を用いた音声通話が可能になる．そして，2006 年にはビデオ通話が可能になり，その後に図 3.1 のようなグループでのビデオ通話が可能になった．現在でも上記の音声，ビデオ通話の機能以外に，IM(Instant Messaging)，リモートデスクトップ，ゲームなどの共有を実現している．

3.1.2 ReConMUC

Pedro Alves and Paulo Ferreira らの研究 [11] は，マルチタブマルチユーザーチャットのための，メッセージ伝播のプロトコルである ReConMUC (Relaxed Consistency Multi-user chat) を提案し，そのプロトコルに対応したマルチユーザチャットアプリケー

図 3.1　Skype のグループビデオ通話 ([8] より抜粋)

ションを開発した．「楽観的レプリケーション」と「パブリッシャサブスクライブモデル」という考え方を使用し，それらを使用したチャットサーバから送信されるメッセージにチャットトピックごとのフィルタリングを掛けることで，使用帯域幅の削減に成功している．「楽観的レプリケーション」は，分散したデータ管理でデータを複製する際に，途中経過で異なっていても最終的にデータ一致すれば良いという考え方である．「パブリッシャサブスクライブモデル」は、送信者であるパブリッシャが送信するメッセージごとにトピックを割り当てメッセージを全体に送信する．受信者であるサブスクライバは各々がトピックを指定し，その指定したトピックのみをパブリッシャから受信するという考え方である．

3.1.3　IVCST

Qiru Zhou and Dong Liu らの研究 [20] は，ライブビデオストリーミング上にフリーハンドスケッチを行うためのレイヤーを追加し提供をおこなうシステムである IVCST(interactive visual content sharing and telestration) を提案し，開発した．このシステムでは，クライアントサイドでオーバレイを追加するので，放映元の映像ソースに対し手を加えず，エンコーディングやデコーディングなどの操作を行わない．その上，映

像ソースの映像フォーマットに依存しない．このシステムは，遠隔医療，バーチャルオフィス，遠隔教育，娯楽，ソーシャルネットワーキングなどの場面での使用が想定されている．

3.1.4 Antwave

井上恭輔, 小野琢也, 山下桂司, 野田昌弘, 岡田正, 寺元貴幸らの研究である Antwave[12] は，本来単独で行う Web ブラウジングを他の利用者との繋がりを利用して行う独自の Web ブラウザである．Antwave では，グリッドブラウジングを言うブラウジングスタイルを提唱している．グリッドブラウジングとは，ブラウザ同士を P2P でつなぎ，利用者間のブラウジングの情報を共有することで，利用者のブラウジングのナビゲートを行い選択肢の幅を個人の範囲を超えたものにするブラウジングスタイルである．そしてこのグリッド上でエクストラリンクと呼ばれる，すべての利用者の Web ページ遷移の軌跡を集約したものを共有する．エクストラリンクを可視化したマップをもとに，利用者は他の利用者が Web ページ訪れた回数でリンクの遷移を行うことが出来る．よって，そのリンク先に有益な情報があるとは限らず，完全に人任せのブラウジングとなる．また，エクストラリンクマップ上のページを他の利用者が閲覧していた場合，その利用者にショートメッセージを送ることも可能である．キーワード検索では，全ての利用者が検索したキーワードを蓄積し，複数の単語に対して予測キーワードを提示し検索を絞ることが可能である．そして，P2P を用いた 1 対 1 での Web ページの共有も可能である．Web ページ共有ではフリーハンドの書き込みを行うことも出来る．

3.2 Web ブラウザによるデジタルコンテンツのリアルタイム共有

専用ソフトを用いず，Web ブラウザのみを利用したリアルタイム共有を行うサービスや研究は以前から数多く存在する．多くのこれらの関連研究では，特別なソフトウェアのダウンロードやインストールやアプリケーションの通信設定などの行為は，コンピュータに不慣れな人々がこれらのサービスを利用することの障害になると考えられている．また，専用のソフトウェアのアップデートなどで，再度ダウンロードやインストールを必要とする場合や，利用者でのバージョンの競合が起こるなどの可能性が存在する．しかし，一般的な Web ブラウザを用いることで，これらの参入障壁を低減させている．また，コンピュータに慣れ親しんでいる利用者にとってもこれらのシステムをすぐに利用できるという利点も存在する．

代表的なサービス及び幾つかの研究とその内容を以下に記す．

3.2.1 TWIDDLA

TWIDDLA[17] は，図 3.2 のようなリアルタイム共有ホワイトボードである．このサービスを利用することで，Web ブラウザ上に自由に，図形，文字，自由曲線などの書き込みのできるホワイトボードを利用者間で共有できるようになる．その上，ホワイトボードには，画像，Web ページ，ドキュメントなどを貼り付け，オンライン会議のために使用することが出来る．

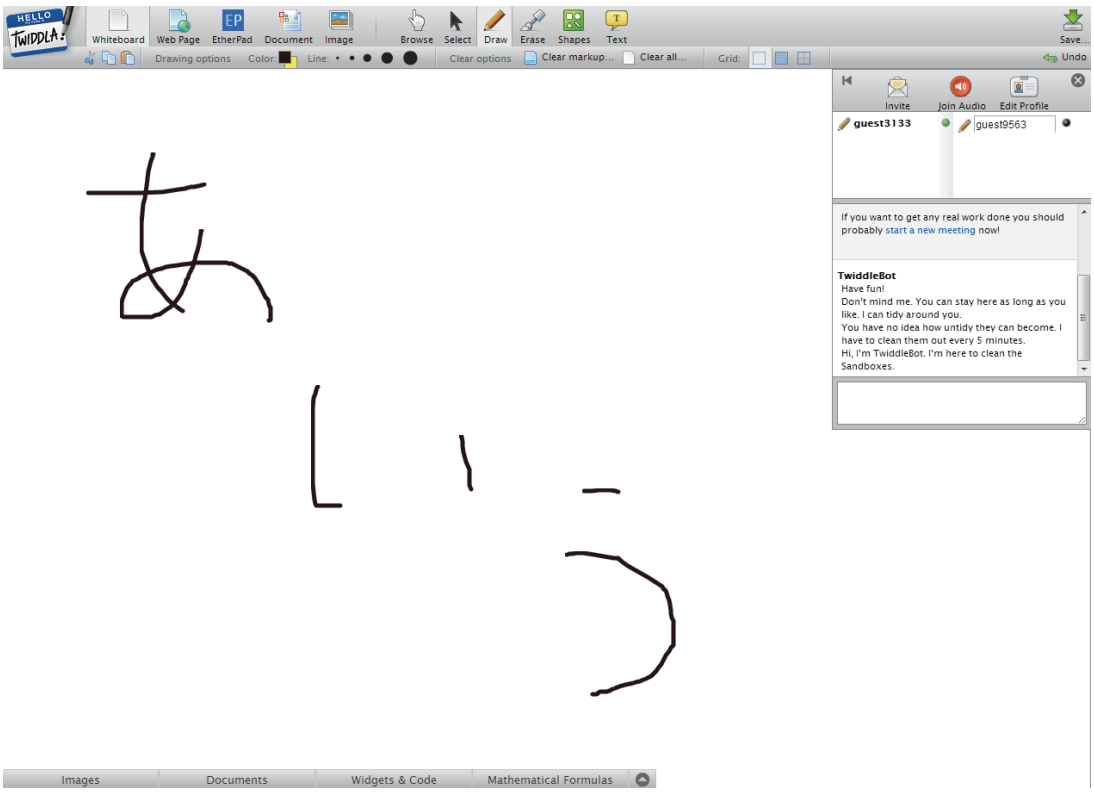


図 3.2 TWIDDLA

3.2.2 RESA

Thomas Sandholm and Hang Maxime Ung らの研究 [16] は，HTML5 を利用できる Web ブラウザのみを利用した，グループの音声ベースの連携を可能にするシステムである RESA (REal-time, Social Audio and location casting) を提案した．このシステムでは，コーディネータとオペレータティブと呼ばれる利用者に分かれる．オペレータティブは一般的な Web ブラウザを利用できる携帯端末の利用が前提となっている．オペレータティブは GPS の位置情報をコーディネータに定期的に送信する．その送信された位置情

報はコーディネータの画面に表示され，定期的に更新される．そして，コーディネータはテキスト形式で入力したメッセージをオペレータティブに対し送信することが出来る．コーディネータが送信したメッセージは，TTS(Text–To–Speech) サーバを経由し音声に変換されオペレータティブに送信される．その送信された音声はオペレータティブの端末で即時に再生される．このシステムは，長距離の自転車レースやマラソンでの利用が想定されている．

3.2.3 動的な Web ページ共有の研究

Dietwig Lowet and Daniel Goergen らの研究 [13] は，JavaScript が利用されている動的な Web ページの共有を可能にするアプローチを提案し，システムを実装した．筆者らは JavaScript エンジンに対して，UI(User Interface) により生じるイベントなどを同期する入力同期と，JavaScript の実行後の DOM(Document Object Model) ツリーを同期する出力同期のアプローチを提案した．システムの実装は，ユーザエクスペリエンスとスケーラビリティが優れている入力同期のアプローチを採用し実装が行われた．システムのプロトタイプでは，Google Maps[10] の共有を実現している．しかし，このシステムは Firefox での使用に限定されている．

3.2.4 synchronite

Christian Thum and Michael Schwind らの研究 [14] は，動的な Web ページの共有を可能にする synchronite と呼ぶシステムを提案した．Dietwig Lowet and Daniel Goergen らの研究と異なる点として，使用 Web ブラウザを限定しない点と，Comet のパブリッシャ/サブスクライバに基づいたイベント複製アプローチを使用している点があげられる．

3.3 グループウェア

グループウェアは，企業向けの情報共有システムの総称である．共有できる情報は，メール，掲示板，スケジュール，ドキュメントなどであり，業務に有益な機能が集約されている．アプリケーションとしてのグループウェアは商品化されている．例として，IBM の Lotus Notes[18] や Microsoft の Microsoft Exchange[19] が挙げられる．また，ブラウザを利用したグループウェアも多数検討されている．そしてブラウザを利用したグループウェアを様々なネットワーク環境下での影響を調査，評価を行なっている研究がある．

Carl Gutwin and Michael Lippold らの研究 [15] は，幾つかのネットワーク設定を用いて，グループウェアを元にした幾つかの異なるネットワーキングのアプローチの性能評価を行った．LAN，MAN，WAN の 3 つの環境で，ブラウザからサーバへ，またサーバから

ブラウザへ 1 万メッセージ送信するテストを行った．ブラウザからサーバへメッセージを送信するテストでは，XHR(XMLHttpRequest)，WebSocket，Java Applet の 3 つの方法でテストが行われた．すべての環境において WebSocket が最も早く 1 万メッセージをサーバへ到達し終えた．Java Applet は LAN と MAN の環境では WebSocket とほぼ同様の結果となったが，WAN の環境では大きな差がついた．XHR は他の 2 つの方法より大幅に遅れていた．サーバからブラウザへメッセージを送信するテストでは，LongPoll，iframe，Multipart，WebSocket，Java Applet の 5 つの方法で行われた．ブラウザからサーバへメッセージを送信するテストと同様に，WebSocket を使用した方法が最も早くメッセージを送信し終えた．iframe，Multipart，Java Applet は，ブラウザからサーバメッセージを送信するテストでの Java Applet と同様の結果になった．LongPoll がすべての環境で最も到達が遅かった．

また，XHR-multipart を利用した共同パズルゲームと WebSocket を利用したフリースケッチのみの共有ホワイトボードのいくつかの使用テストを行った．双方の場合で，テスト参加者はポインタの遅延などの利用に問題になるものは感じないと言及した．

3.4 リアルタイム動画再生共有システム

本研究と同様に，リアルタイムに動画の再生共有を行うシステムの提案を行う研究が存在する．高野祐太郎， 大島浩太，田島孝治，高田治，寺田松昭らの研究 [21] では，FLV(Flash Video) 形式の動画に対し特別なソフトウェアを用いて動画の再生共有と通話とチャットによるコミュニケーションを行うシステムを提案している．システムはクライアント，動画配信サーバ，同期サーバで構成され，動画配信サーバから配信される動画を同期サーバで制御することで複数クライアントでの動画の再生共有を実現している．複数クライアントでの動画の再生共有が可能であるが，動画に対する再生停止や再生位置変更，動画選択などの操作権限は親となる利用者ただ一人しか利用できない．その他の利用者がそのような操作を行いたい場合には，チャットや通話のコミュニケーションの機能を利用して操作権限を持つ親の利用者にその旨を伝え操作してもらわなければならない．

また，この研究では複数の視聴者で動画を視聴している際に，再生している動画の再生時刻がずれて異なる場面を視聴している状態 (以下「同期のずれ」と略記) を考慮している．同期のずれを低減する方式として筆者らは，すべての利用者の，動画配信サーバとのネゴシエーション，受信動画ストリームの初期バッファリング，初期フレームのデコード処理が終了するまで，先にそれらを終了した利用者を待機させ全利用者の足並みを揃え開始する方式を提案している．高野らは Steinmets らの研究 [22] を参考に同期のずれの目標値を 120ms 未満として設定して達成している．

Steinmets らの研究では，ビリヤードの玉の転がる様子の動画とその軌道の動画の 2 つ

の動画を並べ 1 つの動画として視聴した際に 2 つの動画のずれが 120ms 未満であれば人間にとって違和感がない再生が可能であるとしている．

大塚雅博，片岡春乃，末田欣子，下村道夫，淺谷耕一，水野修らの研究 [23] では，動画を用いたコミュニケーションを行う際に，動画の同期のずれにどのくらいの許容範囲があるのかの実験を行った．この研究では，2 つの別れた部屋にそれぞれ電話とモニターを用意した．モニターには特別な装置を用いて遅延が付加され，2 つのモニター間で動画の同期のずれが引き起こされている．実験は，ホストが被験者とモニターに流されている動画を視聴しながら電話を行い，ホストが被験者に幾つか質問することで行われた．使用された動画は，映画，バラエティ，スポーツの 3 つが使用された．実験の結果，展開の変化の大きいスポーツの動画を視聴しながらコミュニケーションを行った場合でも，2 秒以下であればコミュニケーションが円滑に取れると筆者らは述べている．

3.5 関連研究のまとめ

関連研究のまとめと本研究の比較を表 3.1 に示す．表 3.1 は，本研究との比較項目を以下に示す．

- 共有コンテンツ
- 専用のソフトのインストール
- 双方向操作
- 同時利用

共有コンテンツは，各研究及びサービスでどのようなデジタルコンテンツが共有されているかを示す．専用のソフトのインストールは，一般的な Web ブラウザの利用以外を専用のソフトウェアのインストール及びダウンロードが必要としてまとめた．双方向操作は，各研究のシステム及びサービスを利用している全利用者が平等にシステムへの操作権限を有しているかを示す．同時利用は，各研究のシステム及びサービスの共有するコンテンツを多人数で扱えるかどうかを示す．

Lowet らの研究では，専用のソフトのインストールは行わないが，Web ブラウザにプラグインのインストールを行う必要がある．Sandholm らの研究で双方向操作が△とした理由は，このシステムでは送信者が複数の受信者に送信を行うことが前提だが，送信者と受信者の双方になることができ，擬似的に双方向になることができるからである．また，高野らの研究では動画の再生共有の操作権限が親となる利用者一人となるために×としている．

デジタルコンテンツをリアルタイムに共有するためには，これらの項目を満たすことは，利用者の利用しやすさを考慮すると重要であると考えることが出来る．

表 3.1　関連研究のまとめと本システムとの比較

関連研究	共有コンテンツ	専用のソフトのインストール	双方向操作	同時利用
Skype	音声，映像通話 IM ゲームなど	必要	○	多対多
Alves らの研究	チャット	必要	○	多対多
Zhou らの研究	フリーハンド スケッチが なされた動画	必要	○	多対多
Antwave	ブラウザ情報 Web ページなど	必要	○	1 対 1
TWIDDLA	ホワイトボード ドキュメント Web ページ	不要	○	多対多
Sandholm らの研究	位置情報 文章 音声など	不要	△	1 対多
Lowet らの研究	動的な Web ページ	不要 (要プラグイン)	○	多対多
Thum らの研究	動的な Web ページ	不要	○	多対多
グループ ウェア	業務 コンテンツ	不要	○	多対多
高野 らの研究	動画	必要	×	1 対多

3.6 課題

以上の関連研究より，本研究を行う上で達成すべき課題を挙げる．

3.6.1 リアルタイム操作，反映

関連研究で挙げられた，すべてのリアルタイムのデジタルコンテンツ共有システムで達成されている必須項目である．動画を用いたコミュニケーションを行う上でも同様に必須である．利用者が行った操作がリアルタイムに反映され，他の利用者に共有されなければ，コミュニケーションを行う上で大きな問題点となる．

3.6.2 複数の利用者による操作

リアルタイムのデジタルコンテンツ共有システムで，利用者の一人だけに操作が制限されると，他の利用者は操作権限のある利用者に操作の要求をしなければならなくなる．それは，会話の間に操作要求をしなければならないので，円滑なコミュニケーションを阻害する可能性がある．よって，複数の利用者による操作を行うことは重要である．

3.6.3 利用者の負担

デジタルコンテンツのリアルタイム共有を行うシステムが要求する，事前設定などの負担はコンピュータに不慣れな人々が，デジタルコンテンツのリアルタイム共有を行うシステムの利用に対し大きな障害となる．3.2 で挙げた Web ブラウザのみを利用したデジタルコンテンツのリアルタイム共有を行うシステムは，特殊なソフトウェアのダウンロード及びインストールを行わず，ポートの開放などの特別な通信設定を必要としない．それらにより，コンピュータに不慣れな人々が，デジタルコンテンツのリアルタイム共有を行うシステムを利用することの障害を減らしている．動画を用いたコミュニケーションを行う上でも，障害を減らすことはコンピュータに不慣れな人々が利用できるようになるための，大きな利点となる．

3.6.4 動画の同期のずれ

動画を用いたコミュニケーションを行う上で最も重要になるのが，利用者間でコミュニケーションを行うために用いている動画の再生時刻のずれによる，視聴シーンのずれである．他のデジタルコンテンツのリアルタイム共有での同期のずれは，そのコンテンツが動画ほど動的なものではないので，問題にならないことが多い．動画を用いたコミュニケー

ションで動画の同期のずれが生じると，コミュニケーションが円滑に取れなくなる．よって，動画の同期のずれは問題となる．

第 4 章

提案方式

この章では，3.6 で述べた課題より，システムの要求条件を挙げる．そして，その要求条件を達成するシステムの方式の概要を述べる．

4.1 要求条件

3.6 で述べた課題より，本システムで達成すべき要求条件を以下のように定める．

要求条件 1:リアルタイム操作を達成する

投稿された動画を見るように動画をリアルタイムに再生停止や生成位置変更の操作可能で，ライブ形式の動画を見る様に複数の使用者とリアルタイムに同じ場面を視聴できること．

要求条件 2:複数の利用者による操作を可能にする

動画に対する再生停止や再生位置の変更の操作が一人だけではなく，参加者すべてまたは限定した複数人で行えるようにすること．

要求条件 3:利用者の負担を低減する

特別なソフトウェアのダウンロードやインストールを必要とせず，コンピュータに不慣れな利用者の使用を助けるだけでなく，すべての利用者が容易に利用できるようにすること．

要求条件 4:同期のずれを許容範囲に抑える

動画の同期のずれが複数の使用者と視聴しながらコミュニケーションを行なっても会話に齟齬が出ないようにすること．

4.2 方式概要

要求条件を達成するための，各項目に対する方式の概要を以下に記す．

4.2.1 リアルタイム操作を達成する

一般的な Web ブラウザを用いてリアルタイム操作を達成するために，JavaScript を用いて実装を行う．JavaScript を用いて実装を行うことで，インタフェースとして HTML で作成した Web ページ上に配置したボタンなどを利用者が操作した際にプログラムの実行を行うことで，リアルタイムに操作することを可能にする．

4.2.2 複数の利用者による操作を可能にする

複数の利用者による操作を可能にするために，すべての利用者から来た操作命令をサーバで逐次処理を行う．クライアントから来た操作命令をサーバで逐次処理を行うことで，サーバで管理している変数などを競合をおこすことなく，複数の利用者による操作を可能にする．

また，一般的な Web ブラウザを用いて動画の再生共有を行うために，動的な Web サイト共有で用いられている JavaScript のイベント同期の技術を使用する [13, 14]．JavaScript のイベント同期は，Web ページ上で発生した，マウスクリックやボタンの押下などで生じるイベントを他の利用者と共有する．そのことにより，そのイベントを用いて実行するコールバック関数を，全ての利用者で同じタイミングで実行することで動的な Web サイトの共有を行う．

4.2.3 使用者の負担を低減する

使用者の負担を低減するために，一般的な Web ブラウザの機能のみで、実装を行う．一般的な Web ブラウザを用いて実装を行うことで，特別なソフトウェアのダウンロードやインストールの手間を省くことが出来る．また，特別なソフトウェアのダウンロードやインストールを行わないことで，ソフトウェアのアップデートによる再度のソフトウェアのダウンロードやインストールの作業や，それによる利用者間のソフトウェアのバージョンの競合を避けることができる．

本システムでは，HTML5 に対応したブラウザを対象とする．

4.2.4 同期のずれを許容範囲に抑える

同期のずれを許容範囲に抑えるために，ある利用者が再生停止や再生位置の変更を行った際に，その時の動画の時刻を同時に送信する．そして，他の利用者で再生停止や再生位置変更が適用されると同時に，その利用者の動画の時刻を送信されてきた時刻に合わせる．

そして，同期のずれの許容範囲は，動画を用いたコミュニケーションが円滑に取れる 2 秒以下を目標とする．

第 5 章

設計

この章では，本システムの構成を述べ，その構成をもとに利用の流れを説明する．そして，利用の流れより本システムで実現すべき機能を抽出する．最後に，列挙した必要な機能の設計と動作シーケンスを示す．

5.1 システム構成

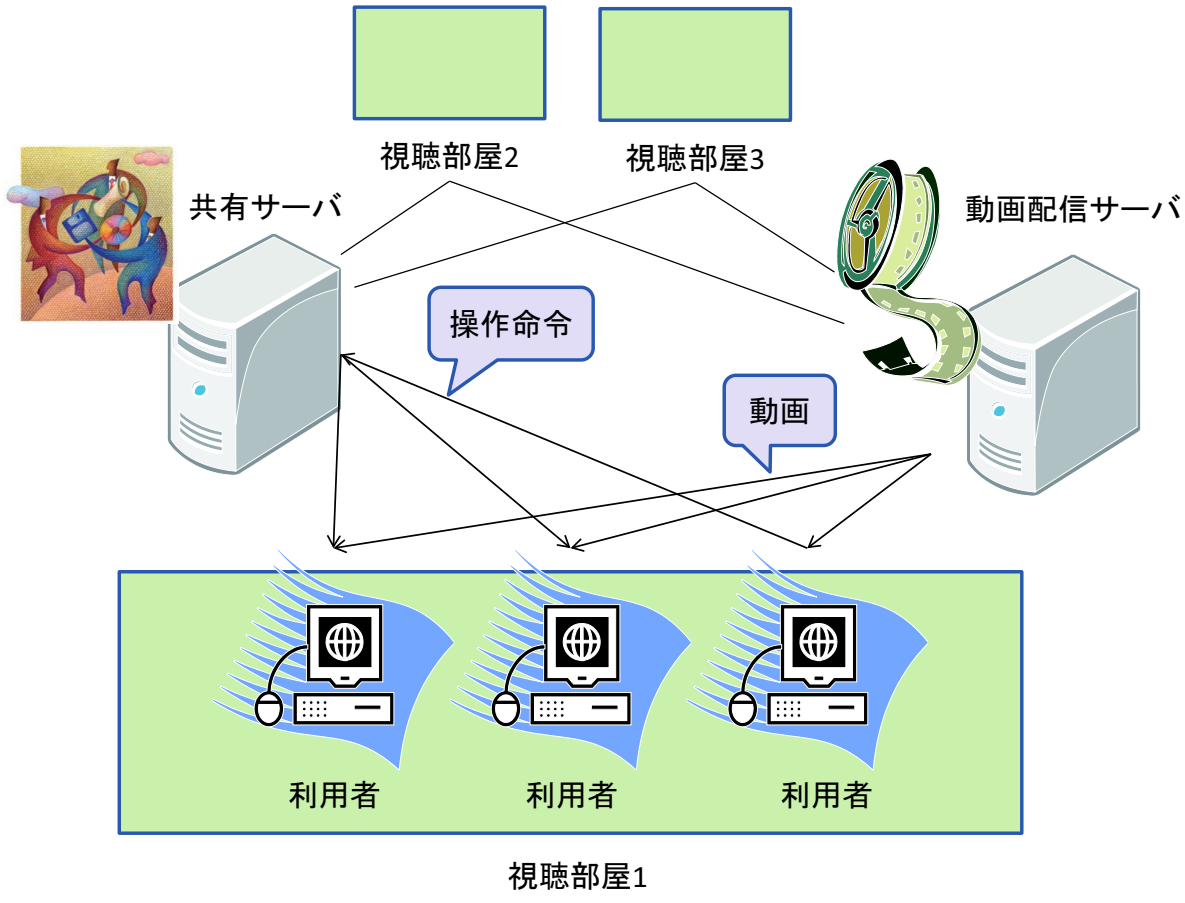


図 5.1 システム構成

図 5.1 に示す本システムは，以下の 3 部分より構成されている．

- 利用者の一般的な Web ブラウザ
- 動画配信サーバ
- 共有サーバ

利用者の一般的な Web ブラウザは，再生共有を行う動画および本システムの再生共有を行うための操作を受け付けるインタフェースの表示を行う．

共有サーバと動画配信サーバばそれぞれ独立して構成されている．動画配信サーバは，利用者の一般的な Web ブラウザに対して動画を提供する．

そして共有サーバは，以下の機能を持つ．

- 操作インタフェースの提供
- 視聴部屋の管理
- 動画の再生共有

利用者は，本システムを利用する際に共有サーバにアクセスを行うことで，本システムのインタフェースを取得する．

共有サーバ内で，図 5.1 の上部と下部に示した，視聴部屋と呼ぶ単位で利用者の視聴グループを管理する．一つの視聴部屋では，視聴部屋内の全ての利用者は同じ一つの動画の視聴を行う．そして，その視聴部屋内で行われた，ある利用者の再生停止や再生位置変更などの操作は，その視聴部屋内の全ての利用者の視聴している動画にそれらの操作が反映される．また，利用者は Web ブラウザで複数の本システムのインタフェースを開くことで，複数の視聴部屋に参加することが可能である．

5.2 利用の流れ

本システムの具体的な利用の流れのシーケンス図を図 5.2 に示す．以下で図 5.2 の流れを説明する．

(1) 本システムへアクセス

始めに，利用者は本システムを利用するために，一般的な Web ブラウザを用いて共有サーバの入室ページへアクセスを行う．

(2) 入室ページの提供

共有サーバは，アクセスしてきた利用者の Web ブラウザ対し入室ページの提供を行う．

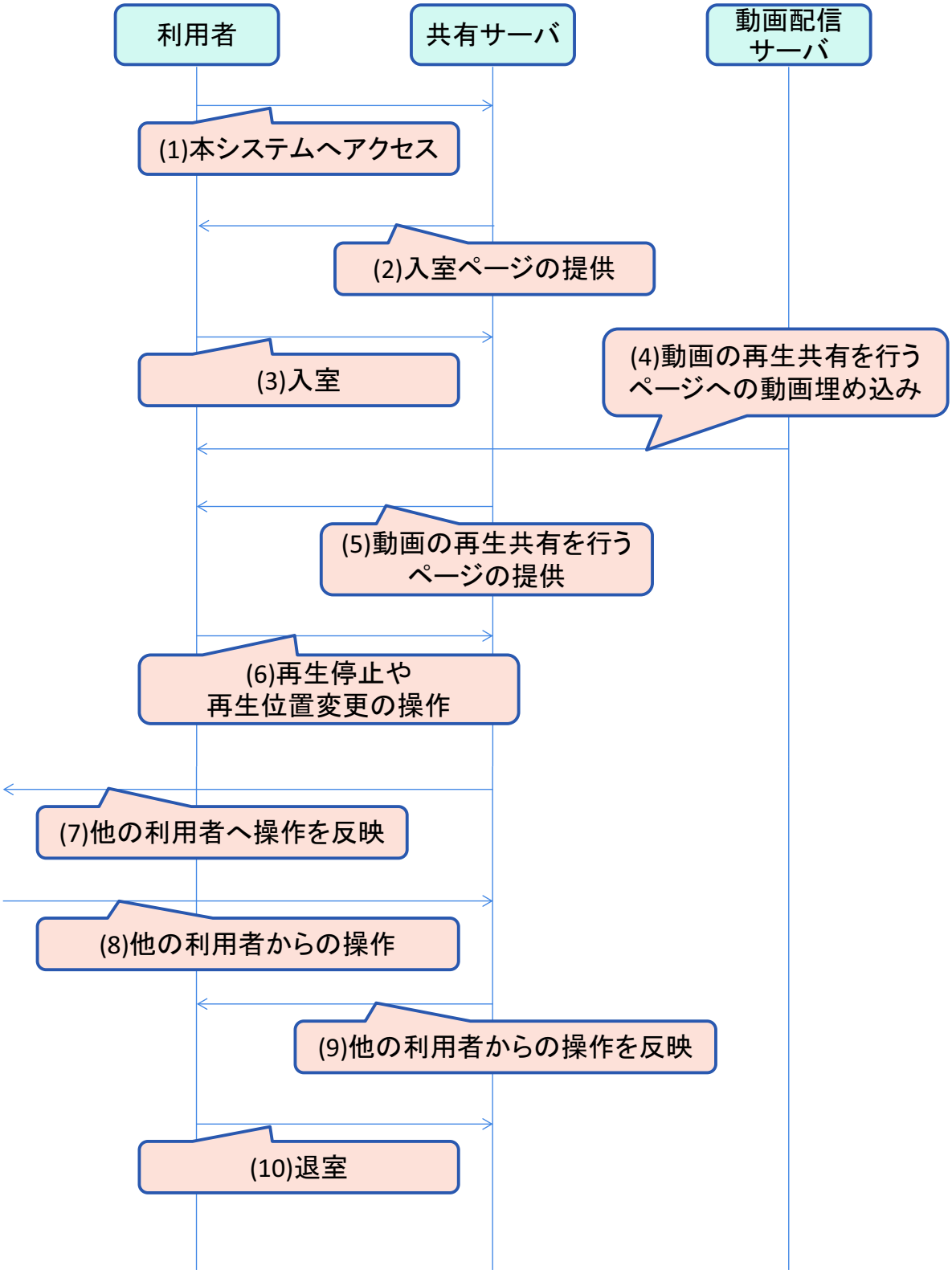


図 5.2　利用の流れのシーケンス図

(3) 入室

利用者は入室ページで，利用者を識別するための「ユーザ名」と入室する視聴部屋を決める「部屋名」，視聴する「動画」を選択し入室ボタンを押し入室する．

(4) 動画の再生共有を行うページへの動画埋め込み

ユーザが入室すると同時に，動画の再生共有を行うページへ，動画サーバから提供される視聴を行う動画を埋め込む．

(5) 動画の再生共有を行うページの提供

共有サーバは，利用者の Web ブラウザに視聴する動画を埋め込んだ，動画の再生共有を行うページを提供する．

(6) 再生停止や再生位置変更の操作

利用者は，動画の再生共有を行うページ上にある，再生停止ボタンや動画の再生位置を示したスライドバーの操作を行う．そして，その行った動作が共有サーバへ送られる．

(7) 他の利用者へ操作を反映

共有サーバは，操作を行った利用者の操作命令を，同じ視聴部屋内の他のすべての利用者へ送信する．

(8) 他の利用者からの操作

他の利用者が操作を行った時も (6) と同様に，その操作命令が共有サーバへ送信される．

(9) 他の利用者からの操作を反映

他の利用者が行った操作の操作命令は，(7) と同様に共有サーバから全利用者に送信される．そして，利用者の Web ブラウザで受信した操作命令は，視聴している動画に適用される．

(10) 退室

利用者は，本システムとの接続を切ることで退室する．

5.3 実現機能

5.2 の利用の流れより，本システムが実現すべき機能を抽出する．(1) 本システムへのアクセス，(2) 入室ページの提供は，共有サーバによる基本的な Web ブラウザと Web サーバの機能により達成される．(3) の入室では，利用者は共有サーバより提供された入室ページに，部屋名，ユーザ名，動画を選択し入室を行う．よって，それらの決定方法を決める必要がある．また，共有サーバ内では，視聴部屋ごとに利用者の視聴グループを区切るので，視聴部屋と各視聴部屋での利用者のユーザ名，視聴動画の管理方法の決定も必要である．そして視聴部屋の作成方法も実現すべきことである．(4) の動画の再生共有を行うページへの動画埋め込みは，HTML に記述することで様々な動画共有サービスの動画配信サーバから動画を取得し埋め込みが可能である．しかし，何の動画配信サーバを使用するかを決定する必要がある．(5) の共有サーバからの動画の再生共有を行うページの提供は，(2) と同様に基本的な Web サーバの機能で行うことが可能である．動画の再生共有を行うページが提供された時に，他の利用者が先に動画の再生共有を行っていた場合の，後から動画の再生共有に参加する利用者の動画の扱いも議論する必要がある．(6)，(8) の再生停止や再生位置の変更操作では，動画配信サーバより動画の再生共有を行うページに埋め込まれた動画の再生停止や再生位置変更の操作を取得し，それらを他の利用者に送信する．よって，利用者の動画の再生停止や再生位置変更の取得を実現する必要がある (7)，(9) の再生停止や再生位置変更の操作の反映では，ある利用者が送信した再生停止や再生位置変更を，全ての視聴部屋内の利用者に適用し動画の再生共有を行う．よって，再生停止や再生位置変更の命令を受信した利用者の動画の再生共有を行うページに埋め込まれている動画に適用する必要がある．また (6)〜(9) では，それらの再生停止や再生位置変更を行った際に生じる恐れのある動画の同期のずれを低減させる処理も必要である．(10) の退室では，退室の際に共有サーバ内の視聴部屋から利用者を削除する必要がある．また，視聴部屋から利用者がいなくなった際の視聴部屋の扱いも決定する必要がある．

以上より本システムで実現すべき機能をまとめると以下のようになる．

- 入室時処理
  - 入室時のユーザ名，視聴部屋，視聴動画の決定方法
  - ユーザ名，視聴部屋，視聴動画の共有サーバでの管理方法
  - 視聴部屋への入室
- 動画の再生共有
  - 再生停止や再生位置変更の操作の取得
  - 再生停止や再生位置変更の操作の他の利用者への送信

- 再生停止や再生位置変更の操作の利用者への適用
- 後から視聴部屋へ入ってきた利用者の動画の再生共有

- 退室処理
  - 視聴部屋からの利用者の削除
  - 利用者のいない視聴部屋の扱い

5.4 機能設計

以下で，入室処理，動画の再生共有，退室処理の 5.3 で抽出した実現機能の設計を行う．しかし，これらの実現機能の設計を行う前に，動画の再生共有を行うインタフェースの具体的な機能を決定しておく必要がある．本システムは，一般的な Web ブラウザを用いて動画の再生共有を行うインタフェースを提供するので，動画の再生共有を行うインタフェースは HTML(HyperText Markup Language) を元に設計を行う．

5.4.1 動画の再生共有を行うインタフェース

動画の再生共有を行うインタフェースは，通常の動画の視聴と同様の操作を行えるインタフェースを提供する．通常の動画視聴で利用する操作は以下の 4 つを挙げることができる．

- 再生
- 一時停止
- 停止
- 再生位置の変更

再生は，動画の視聴を開始する操作である．一時停止は，視聴中の動画をその場面で再生を止める操作である．停止は，動画の視聴を終了し，始めの状態に戻る操作である．再生位置の変更は，テレビを利用した VHS(Video Home System) や，DVD(Digital Versatile Disc) の視聴では，通常早送や巻き戻しなどの操作により再生位置の変更を行うが，今日の PC 上での動画再生では，現在の動画の再生位置が示されたスライドバーのボタンの位置を移動させることにより動画で再生されている場面の移動を行う操作である．

再生，一時停止，停止は，HTML の $\langle BUTTON \rangle$ タグを用いて設置を行う．また，再生および一時停止は同時に操作を行うことは無いので，設置するボタンの表示は，再生中は一時停止の表示を行い，一時停止及び停止中には再生の表示を行い，各々の操作の実行を可能にする．再生位置の変更の操作を行うのに用いるスライドバーは，HTML5 の既存のスライダーを用いず，JavaScript で操作を行い，画像の座標と動画の再生時刻を関連

付けることにより実現する．なぜなら，HTML5 の既存のスライダーは現時点で Firefox などの一部のブラウザで対応が行われていないからである．また，動画の時刻もスライドバーとともに表示を行う．

それらの他に，複数の利用者で動画の再生の共有を行うので，視聴部屋に参加している全ての利用者の名前を表示できることが好ましい．また，利用者個人のユーザ名の表示や視聴部屋の部屋名の表示も行う．本システムの利用は，コミュニケーションを行うことが前提となっているため，簡易的なチャット機能を実装する．

5.4.2 入室処理

以下では，入室時に行う処理の設計を行う．始めに，入室時のユーザ名，部屋，動画の決定方法，次にそれらの入室時のユーザ名，部屋名，動画の共有サーバでの管理方法の決定を行う．最後に視聴部屋への入室するための処理の設計を行う．

入室時のユーザ名，視聴部屋，視聴動画の決定方法

利用者が視聴部屋に入る前の，入室ページでのユーザ名，部屋名，動画を決定する．ユーザ名の決定は，利用者が任意に選択可能なので，利用者が自由に記入することのできるテキストボックスへの入力を行う．

視聴部屋の決定は，新規の視聴部屋の決定と既存の視聴部屋の決定の 2 通りの場合を考慮する必要がある．新規の視聴部屋の決定は，ユーザ名の決定と同様に，利用者が任意に部屋名の決定を行うことを想定するので，テキストボックスへの入力を行う．既存の視聴部屋の決定は，以下の 2 通りの方法を考えることが出来る．

- テキストボックスへの入力
- リストから選択

テキストボックスへの自由入力は，ユーザ名の決定と新規の視聴部屋の決定と同様に，テキストボックスへ部屋名を入力し，共有サーバで管理されている部屋名と照合し，一致したらその視聴部屋へ入室を行う方法である．リストから選択は，既存の視聴部屋のリスト表示を行い，その中から入室する視聴部屋を選択し入室を行う方法である．2 つの方法を比較した場合，既存の視聴部屋のリスト表示を行う方法では，既存の視聴部屋の数が多くなった際に，リストの中から目的の視聴部屋を捜索するのに手間がかかる．テキストボックスへの入力では，利用者が入室しようとしている既存の視聴部屋の部屋名さえ知っていれば容易に入室することが可能である．よって，本システムでは既存の視聴部屋の決定は，テキストボックスへの入力による決定を採用する．また，テキストボックスへの入力による既存の視聴部屋の決定は，リストからの選択と異なり新規の視聴部屋の決定と同じ

インタフェースを使用することで部屋の決定を行うことが可能である．

動画の決定方法を定める前に，本システムで扱う動画の種類を定める必要がある．現在インターネット上に存在し，一般的な Web ブラウザで再生可能な動画として，YouTube やニコニコ動画などの動画共有サイトに投稿されている動画や，Windows Media Player[24] のプラグインや，独自の Flash の動画プレイヤーを用いた Web ページへの埋め込みの動画などが存在する．また，2014 年までの正式勧告が目標となっている HTML5 の機能の一部である，video タグを用いることで，Web ブラウザのネイティブの機能を用いてプラグインのインストールをなしに Web ページに動画を埋め込むことを可能にする．本システムでは，埋め込み動画を API を用いて操作することのできる，世界中で最も使用されている動画共有サイトである YouTube の動画とブラウザのネイティブの機能を利用した HTML5 video の動画の 2 種類の動画を利用する．YouTube は従来は，FLV(Flash Video) 形式の動画の再生を行うために，Flash プラグインのインストールが必須であったが，現在では HTML5 video の機能を用いることで Flash プラグインのインストールを必要とせず動画を試聴することが可能である．よって，YouTube の使用は本システムの要求条件である，利用者の負担を低減するを阻害しない．また，現在 HTML5 video では，表 5.1 が示すようにブラウザによって利用できる動画の形式が異なる．全ての一般的な Web ブラウザで対応させるためには表 5.1 より，MP4 (h.264/AAC) と WebM (VP8/Vorbis) の 2 つの動画形式の動画を用意すれば良いことがわかる．しかし，HTML5 に対応していないバージョンの古い Web ブラウザでは再生することができないことが問題である．従って，本システムでは，HTML5 に対応した Web ブラウザを使用することは必須となる．

表 5.1　HTML5 video のブラウザでの利用できる動画形式 (文献 [25] 参照)．

ブラウザ/動画形式	MP4 (h.264/AAC)	WebM (VP8/Vorbis)
IE8	非対応	非対応
IE9 以上	対応	インストール
Chrome	対応	対応
Safari	対応	非対応
モバイル	対応	非対応
Firefox	非対応	対応
Opera	非対応	対応

そして動画の決定方法は，以下の 3 通りを考えることが出来る．

- テキストボックスへの URL の入力
- 検索ページを表示し動画検索
- リストから選択

テキストボックスへの URL の入力は，上記のユーザ名，部屋名の決定と同様に，自由入力可能なテキストボックスをインタフェースとして提供し，利用者が Web 上から取得してきた動画の URL を貼り付ける，または入力を行うことによって試聴する動画の決定を行う．検索ページを表示し動画検索は，本システムのインタフェースとして動画検索ページを提供し，そのページで試聴する動画の検索を行う．そして，検索結果から試聴する動画を決定する．リストから選択は，予め決められた動画のリスト表示を行い，その中から試聴する動画を決定する．著作権の問題を考慮して，本システムのプロトタイプでは独自に用意した動画を用いるので，いくつか用意した動画のリストから試聴する動画を選択する形式を採用する．実際のシステムでは，テキストボックスへの URL の入力と検索ページからの動画選択の組み合わせが有用であると考えられる．

ユーザ名，視聴部屋，視聴動画の共有サーバでの管理方法

本システムの各利用者の利用時間は，長く見積もっても映画の視聴時間である程度であり，何日または何週間もの利用は行われないと考える．従って，本システムを利用する利用者の入れ替わりがある程度頻繁に行われることが想定できる．よってユーザ名，視聴部屋，視聴動画の情報は容易に，作成や削除などの操作を行える必要がある．また本システムにおいて，利用者が再度同じ，ユーザ名，視聴部屋，視聴動画の情報を用いて動画の再生共有を行いたい場合，これらの情報の保存や復元を行わないので，再度同じこれらの情報を入力を行う必要がある．これは本システムで，これらの情報の保存や復元が必要なほど，動画の視聴の状態が複雑になると想定しないからである．

そこで，これらのユーザ名，視聴部屋，視聴動画の共有サーバでの管理は，様々なプログラミング言語で使用可能なリスト構造を用いて管理を行う．ユーザ名，視聴部屋，視聴動画の内包関係を木構造で表すと図 5.3 の様になる．各視聴部屋ごとに，ユニークな部屋名と視聴動画があり，それに複数の利用者が存在する形となるので，従って，図 5.3 で示した木構造をリスト表示するとソースコード 5.1 になり，ユーザ名，視聴部屋，視聴動画の管理はこのリストを用いて行う．また，ここで管理を行う視聴動画は，動画の URL もしくは，投稿型動画サイトの動画 ID である．

```
1  list[部屋番号][部屋名，視聴動画，ユーザ名，・・・]
```

ソースコード 5.1　リスト表示

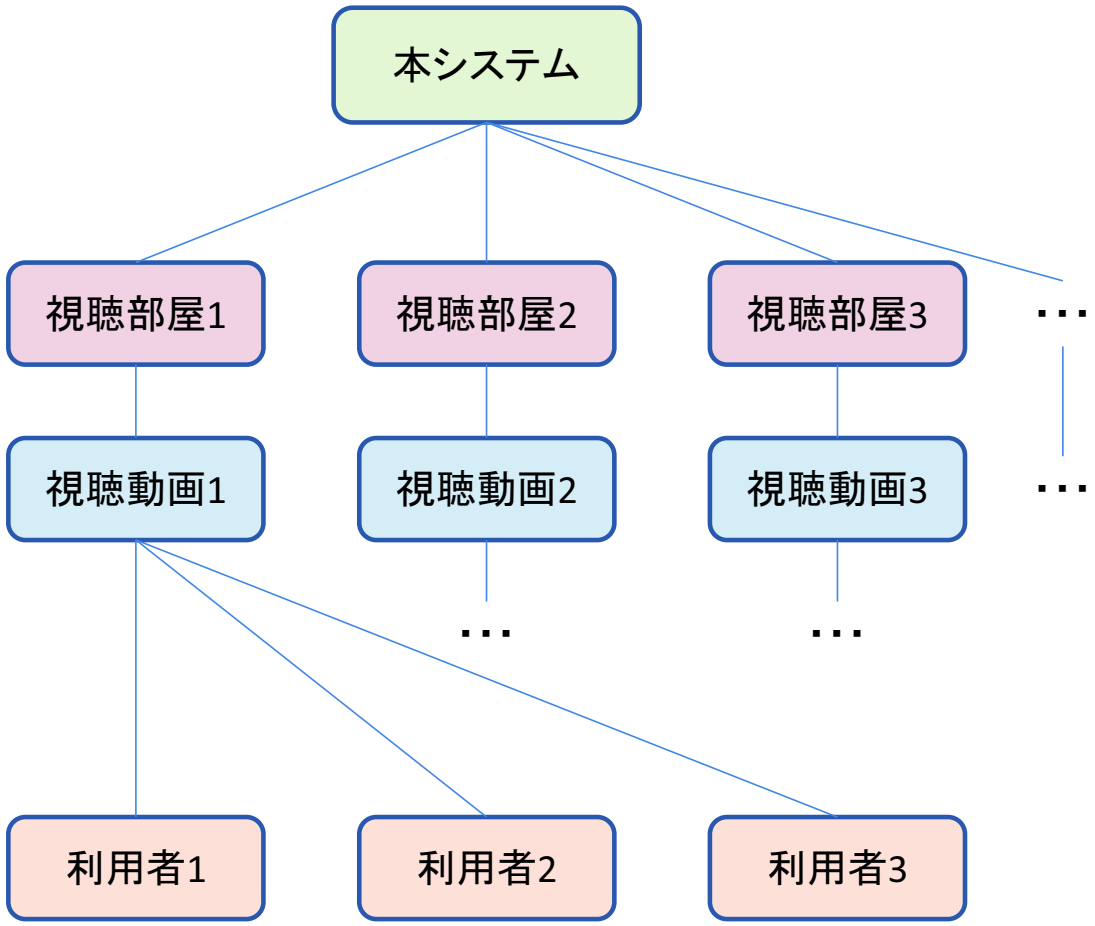


図 5.3　ユーザ名，視聴部屋，視聴動画の木構造

視聴部屋への入室

利用者が入室画面でユーザ名，部屋名を入力し，動画の選択を行いそれらの情報を共有サーバへ送信した際の共有サーバの処理から，利用者に動画の再生共有を行うインタフェースを提供するまでの設計を行う．

始めに，これらのユーザ名，部屋名，視聴動画の情報を取得した共有サーバは，これらの情報の共有サーバ内での重複を調べる必要があると考える．よって先に，利用者のユーザ名，視聴部屋の部屋名および，選択された動画の重複の扱いを定める．同一の視聴部屋内での利用者のユーザ名の重複は，利用者の混乱を招くため好ましくないので，避けるべきである．上記の入室時のユーザ名，視聴部屋，視聴動画の決定方法で定めた視聴部屋の決定方法では，入力した部屋名が既存の部屋名に存在する場合にはその既存の視聴部屋に参加し，入力した部屋名が既存の部屋名に存在しない場合には新たな視聴部屋として参加を行う様決定した．従って，視聴部屋の部屋名の重複は自動的に避けられる．選択された

動画の重複として考えることが出来るのが，異なる視聴部屋で同じ動画を視聴する場合である．しかし，視聴部屋は各々が独立しているためこれは問題ではない．また，本システムの入室ページでは，既存の部屋名を入力して入室する際に，その既存の視聴部屋で視聴している動画とは別の動画を選択することが可能である．しかし，一つの視聴部屋に対し試聴する動画は一つであるため，視聴部屋が始めに作成された時点に選択された動画の視聴を行い，後から選択された動画は適用しない．

以上のユーザ名，部屋名，視聴動画の情報の重複の扱いより，以下で重複検索の優先度を定める．ユーザ名の重複は，各視聴部屋内のみで考慮され，その視聴部屋ないでユーザ名の重複の検索を行うので，ユーザ名の重複の検索を行うより先に部屋名の重複の検索が行われるべきである．そして，視聴動画の重複の検索は行わない．

ここで，上記のユーザ名，部屋名，視聴動画の情報の重複の扱いと，5.4.1 で示した動画の再生共有を行うインタフェースで提供すべき情報より，視聴部屋へ入室する際の共有サーバで行うことが想定される処理は以下のようになる．

- 部屋名の重複の検索
- 視聴部屋の新規作成
- 利用者の既存の視聴部屋への入室
- ユーザ名の重複の検索

本システムの，ユーザ名，部屋名，視聴動画の情報は，ソースコード 5.1 で示したリストで表現するため，部屋名の検索及び，ユーザ名の検索はリスト内で行われる．従ってリストから視聴部屋を検索する擬似コードは，ソースコード 5.2 の様になる．

```
1 for(i=0; i<既存の視聴部屋数; i++){
2   if(list[i][0]==利用者が入力した部屋名){
3     利用者の既存の視聴部屋への入室;
4   }else{
5     視聴部屋の新規作成;
6   }
7 }
```

ソースコード 5.2　視聴部屋検索

視聴部屋の新規作作成は，ユーザ名，部屋名，視聴動画の情報を管理しているリストに，新たな列を加えることで達成できる．新たな列を加える擬似コードをソースコード 5.3 に示す．

```
1 list[視聴部屋数+1]=新規視聴部屋;
2 list[視聴部屋数+1][0]=新規部屋名;
3 list[視聴部屋数+1][1]=新規視聴動画;
4 list[視聴部屋数+1][2]=ユーザ名;
```

ソースコード 5.3　視聴部屋の新規作作成

利用者の既存の視聴部屋への入室を行う際に，ユーザ名の重複の検索を行う．ユーザ名の検索も，視聴部屋と同様な方法で行うことが出来る．また，ユーザ名が重複した場合は，視聴部屋への入室を行わせず，入室画面にユーザ名が重複しているというアラートを利用者へ送信し表示する．ユーザ名の重複を検索する擬似コードを，ソースコード 5.4 に示す．

```
1 for(i=0; i<視聴部屋内の利用者数; i++){
2   if(list[部屋番号][i+2]==ユーザ名){
3     利用者へアラートの送信;
4   }
5 }
```

ソースコード 5.4　ユーザ名の重複の検索

既存の視聴部屋へ入った利用者は，その視聴部屋より，視聴動画，視聴部屋内の全ユーザ名を取得し，動画の再生共有を行うインタフェースへそれらの情報が埋め込まれる．また，既存の視聴部屋内にいるすでに動画の共有再生を行なっている利用者たちに，新規の利用者のユーザ名の入った視聴部屋の全ユーザ名のリストをの送信を行う．そして，新規の利用者が動画の再生共有に加わったと言うチャットウィンドを用いたアノテーションを行う．

5.4.3　動画の再生共有

動画の再生共有を行う上で最も重要となる再生停止や再生位置の変更の機能の設計を行う．動画の再生共有を行う際に，利用者の Web ブラウザと共有サーバ間での相互通信が活発に行われるので，先に使用サーバ及び通信方法の決定を行う．そして，利用者からの再生停止や再生位置変更の操作の取得，取得した操作の他の利用者への送信，そして受信した利用者のそれら再生停止や再生位置変更の操作の視聴動画への適用の設計を行う．最後に，後から視聴部屋へ入ってきた利用者の動画の再生共有の扱いの決定及びその設計を行う．

共有サーバ及び通信方法

動画の再生共有を行う上で，利用者が再生停止や再生位置の変更の操作を行なった際に，他の利用者の動画に可能な限りすぐに反映させることができることが理想である．JavaScript で XHR(XMLHttpRequest) を利用した通信では，Web ブラウザ側からサーバへのデータの送信を行うことや，Web ブラウザ側からサーバへのデータの送信要求を行うことが可能である．しかし，Web ブラウザ側からサーバへデータの送信要求を行うが，サーバから Web ブラウザへ対しデータのプッシュを行うことが不可能である．サーバ側でデータが送信可能な状態であっても Web ブラウザ側からのデータの送信要求が来

ない限り，そのデータは Web ブラウザ側に到達できないので，データが送信可能な状態になってから Web ブラウザ側から送信要求が来るまでの期間が送信時間のロスになる．そして，通信を行うたびにコネクションを張る必要がある．

また, サーバ側からのプッシュ送信が可能な通信技術として Comet が存在する．Comet を利用した通信では，Web ブラウザ側からデータの送信要求を予め送りコネクションを張っておき，そのコネクションをサーバ側のデータが送信可能な状態になるまで保持しておくロングポーリングを行う．そして，サーバ側のデータが送信可能な状態になった時にそのコネクションを用いて Web ブラウザ側に送信することでプッシュ通信を可能にしている．しかし，このロングポーリングを行うことで，長時間 HTTP コネクションを占有するため，コネクションの本数が多くなりサーバの負荷が増大する可能性がある．ロングポーリングを行なっているコネクションはサーバ側で処理待ちの状態で待機を行なっており，そのスレッドで他の処理を行うことが不可能である．また，擬似的な双方向通信であるため，通信が発生するたびにハンドシェイクを行う必要がある．

完全な双方向通信を行うことの出来る通信技術として WebSocket が登場した．WebSocket を利用した通信では，Web ブラウザとサーバが一度コネクションを張ると，コネクションを切断するまで専用のプロトコルを用いてそのコネクション上で，自由に双方向でデータのプッシュ送信を行うことが可能である．従って，Web ブラウザ側はデータ受信のためのポーリングを必要とせず，待ち時間なしでデータを受信することが可能である．また，1 つの Web ブラウザとサーバ間の通信でコネクションが 1 つで，ハンドシェイクは開始時の 1 度だけなのでサーバの負荷が上記の Comet に比べ少ないと言う利点もある．XHR，Comet，WebSocket の通信の比較を図 5.4，まとめを表 5.2 に示す．従って，本システムでは WebSocket を採用する．

表 5.2　XHR，Comet，WebSocket の通信のまとめ

通信技術	サーバ負荷	双方向通信	即時送信
XHR	×	×	×
Comet	×	△	○
WebSocket	○	○	○

WebSocket 通信の行えるサーバアプリケーションに，Node.js[26] が存在する．Node.js は Socket.io[27] のライブラリを用いることで，容易に WebSocket 通信を行うことが出来る．また Node.js は Google V8 JavaScript Engine で JavaScript で記述されたサーバプログラムの処理を行う．そしてイベントを待機し，その発生したイベントに従い処理を行うイベント駆動型であるため，再生停止や再生位置の変更などの操作に対し処理を行う本

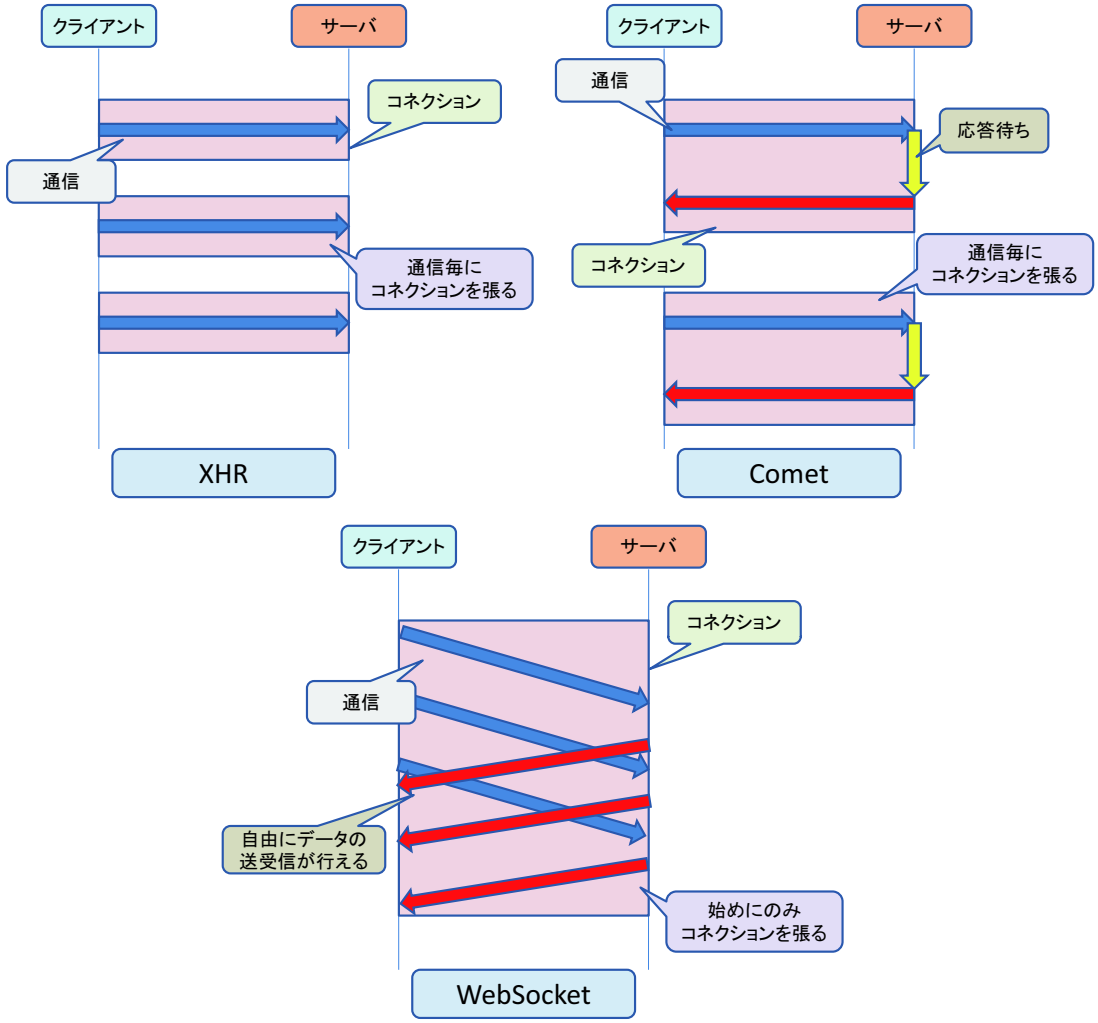


図 5.4　XHR，Comet，WebSocket の通信の比較

システムで非常に有用である．Node.js と Socket.io を使用した Web ブラウザとサーバの送受信の例をソースコード 5.5 に示す．ソースコード 5.5 では，emit，on のメソッドで同じ’event’ を記述することで，データの送受信が行われる．emit では送信するデータを記述し，on では受信したデータを元にしたコールバックの実行を行う．以上のように，データの送受信を容易行うことが可能である．従って，本システムでは Node.js と Socket.io の組み合わせを採用する．

```
1 /*送信*/
2   socket.emit('event', {data});
3
4 /*受信*/
5   socket.on('event', function (data) {callback});
```

ソースコード 5.5　Node.js と Socket.io を使用した通信例

再生停止や再生位置変更の操作の取得

本システムは，一般的な Web ブラウザでインタフェースを提供するため，Dietwig Lowet and Daniel Goergen らの研究 [13] で採用されている JavaScript イベントの入力同期を用いて実現を行うことが可能である．JavaScript イベントの入力同期とは，Web ページに対する利用者のボタンのマウスクリックやキーの押下，テキストボックスへの文字入力などで生じる JavaScript のイベントを取得し，その JavaScript を他の利用者へ送信を行い，他の利用者の Web ブラウザであたかもそのイベントが発生した様に見せる方法である．再生，一時停止，停止は，ボタンの押下によるイベントをトリガーとして取得することが可能である．また再生位置の変更は，スライドバーのボタンのドラッグを行い，ボタンの位置を変更した後のマウスアップによるイベントをトリガーとして取得することが可能である．従って本システムで動画の再生を共有を行う際に，利用者の再生停止や再生位置変更の操作の JavaScript のイベントを取得することで，再生停止や再生位置変更の操作の情報を取得できたと言える．

再生停止や再生位置変更の操作の他の利用者への送信

上記の再生停止や再生位置変更の操作の取得で取得したイベントを，視聴部屋内の全ての他の利用者へ送信する方法を検討する．まず，本システムは P2P 通信を行わないので，利用者から取得したイベントをまず共有サーバへ送信する必要がある．送信は上記の共有サーバ及び通信方法で述べた，Node.js と Socket.io の emit の機能を用いて行う．そして，サーバから視聴部屋内の他の利用者へ，利用者から送信された再生停止や再生位置変更の操作のイベントの送信を行う方法は，利用者からサーバへ送信するのと同様に emit の機能を用いて行う．

また，ここで動画の再生共有を行う上での視聴動画の同期のずれを低減することを考慮する必要がある．利用者からサーバへ，再生停止や再生位置変更の操作のイベントを送信する上で，考慮できる視聴動画の同期のずれを低減する方法は，これらの操作が他の利用者に送信され適用された際に，そこで動画の再生位置の足並みを揃えることである．つまり，再生停止や再生位置変更の操作のイベントを送信する際に，その操作を行った利用者の動画の再生時刻を共に送信し，その再生時刻も他の利用者に適用することである．

従って共有サーバは，ソースコード 5.1 のリスト構造より，ソースコード 5.6 の擬似コードに示すような形で視聴部屋内の全ての利用者に再生停止や再生位置変更の操作のイベントの送信を行う．

```
1 for(i=0; i<視聴部屋内の利用者数; i++){
2   list[部屋番号][i+2].emit('操作イベント', 再生時刻);
3 }
```

ソースコード 5.6　サーバから利用者へのイベント送信

また，Node.js はシングルスレッドで動作するため，これらの操作命令は共有サーバで逐次に処理が行われる．従って，ほぼ同時に送信されてきた操作命令も順序付けられ処理が行われ，共有サーバ内で保持している変数や利用者の操作などの一貫性が保たれる．

再生停止や再生位置変更の操作の利用者への適用

共有サーバから送信された，再生停止や再生位置変更の操作のイベントの受信は，Node.js と Socket.io の on の機能を用いて行う．そして，再生停止や再生位置変更の操作のイベントを受信した後に，各々の受信したイベントに対するコールバック関数を実行する．コールバック関数内では，YouTube や HTML5 video などの使用している動画の API を用いて再生停止や再生位置変更の操作の利用者への適用を行う．再生は，まず，操作の適用中に利用者から別の入力が行われる事による，操作の混乱を防ぐため先に動画の再生共有を行うインタフェースの操作をすべて不可にする．そして送信されてきた再生時刻の適用を行なってから再生を開始を行い，そしてインタフェースの操作を可能にする．一時停止は，再生と同様に先にインタフェースの操作をすべて不可にする．そして再生とは逆に，一時停止を行なってから送信されてきた再生位置の適用を行い，そしてインタフェースの操作を可能にする．停止も同様に，インタフェースの操作を不可にし，一時停止を行なってから再生時刻を 0 にする．そして最後にインタフェースの操作を可能にする．再生位置の変更は，まずインタフェースの操作を不可にする．次に，送信されてきた動画の再生時刻を適用するために，動画の再生を一時停止をおこなう．そして，送信されてきた動画の再生時刻を適用する．そしてここで，再生位置を変更が行われた際に，視聴動画が再生中であれば再生を行い，一時停止または停止中であれば停止のままにする．よって再生位置の変更を送信する際には現在の動画の再生停止の状態も共に送信すべきである．最後に，インタフェースの操作を可能にする．

後から視聴部屋へ入ってきた利用者の動画の再生共有

他の利用者が動画の再生共有を行なっている際に，途中で新規利用者が参加した場合の新たな利用者の始めの動画の再生共有の扱いとして以下の 3 つが挙げられる．

- 新規利用者を含めた視聴部屋内の利用者の再生位置を揃えて動画の再生共有を開始する．
- 新規利用者の再生位置を揃えずに，動画の再生共有を開始する．

- 視聴部屋内の誰かが，再生停止や再生位置の変更を行うまで動画の再生共有を行わない．

「新規利用者を含めた視聴部屋内の利用者の再生位置を揃えて動画の再生共有を開始する」は，再生中に新規利用者が視聴部屋に参加した場合，その時の再生時刻を取得し，その再生時刻に合わせて視聴部屋内の全ての利用者で動画の再生共有を行う．この場合は，既存の利用者の視聴中の動画が新規の利用者が参加することで若干一時停止が行われる．「新規利用者の再生位置を揃えずに，動画の再生共有を開始する」は，「新規利用者を含めた視聴部屋内の利用者の再生位置を揃えて動画の再生共有を開始する」と同様に再生時刻を取得するが，他の利用者と足並みを揃えずに新規の利用者だけに適用する．この場合は，取得した再生時刻を新規の利用者に適用するまでに，既存の利用者の動画再生が進んでいるため，初期の動画の再生共有では動画に同期のずれが生じる．「視聴部屋内の誰かが，再生停止や再生位置の変更を行うまで動画の再生共有を行わない」は，新規の利用者を含め視聴部屋内の利用者の内の一人が再生停止や再生位置の変更を行うまで，新規の利用者の動画の再生共有は行わない．この場合では，すぐに新規の利用者が動画の再生共有に参加することができない．また新規の利用者が再生の操作を行った場合，既存の利用者が今まで視聴していた場面を無視して始めから再生が行われるようになる．本システムでは，動画の再生共有を用いたコミュニケーションを行うことを前提としているため，個人の動画の視聴のスムーズさより新規の利用者を含めた視聴部屋内の全利用者が同じ場面を視聴していることが好ましいので，新規利用者を含めた視聴部屋内の利用者の再生位置を揃えて動画の再生共有を開始する方式を採用する．

新規利用者を含めた視聴部屋内の利用者の再生位置を揃えて動画の再生共有を開始する方式では，新規の利用者が入室した際に，視聴動画の現在の視聴場面の時刻を取得する必要がある．視聴部屋の動画の時刻は，共有サーバで管理されているわけではなく，個々の利用者で独立している．よって，動画の時刻は視聴部屋内の既存の利用者の内の一人から取得する必要がある．ここで，視聴部屋内の代表の利用者として視聴部屋を作成した一番初めに参加している利用者 (リストの先頭) の動画の時刻を利用する．新規の利用者と他の既存の利用者で揃えて動画の再生共有を始めるために，代表の利用者の再生時刻を視聴部屋内の全ての利用者に送信しなければならない．代表の利用者は，現在の再生時刻に動画の再生位置の変更の操作を行うことで，新規の利用者を含めた視聴部屋内の全利用者で動画の再生共有を再開することが可能である．

5.4.4 退室処理

以下で，視聴部屋から利用者が退室した際の共有サーバでの視聴部屋からの利用者の削除の設計をおこなう．また，視聴部屋から利用者がすべて退室した際の視聴部屋の扱い及

びその処理の設計を行う．

視聴部屋からの利用者の削除

利用者が本システムの利用を終了は，Web ブラウザのタブや Web ブラウザそのものを終了することによる，共有サーバとの接続を切断することで行われる．共有サーバと利用者が切断されることで，共有サーバは視聴部屋から利用者の削除を行う必要がある．視聴部屋の管理はソースコード 5.1 に示した，リスト表示で行われているので，そこから利用者を削除すればよい．再生停止や再生位置変更の操作の送信は，視聴部屋内のリストの先頭から順次送信されるので，リストから利用者を削除する際は，リストに空きを作るのではなく，リストを詰める形式が好ましい．そして，退室した利用者を削除したリストを，視聴部屋内の全利用者に送信を行いインタフェースの視聴部屋の利用者表示の更新を行う．

利用者のいない視聴部屋の扱い

利用者のいない視聴部屋の扱いは，一定期間視聴部屋を残す方式と，完全に削除を行う方式が挙げられる．一定期間視聴部屋を残す方法は，視聴部屋内の利用者が 0 人になったとしても視聴部屋を残しておき再度利用者が入室した場合に視聴部屋内で動画の再生共有が行える方式である．この方式では，利用者が 0 人の視聴部屋が多数存在する可能性があることや，視聴部屋を残す時間を定めるのが難しいという問題点がある．完全に削除を行う方式では，視聴部屋にいる最後の一人の利用者がいなくなると同時に視聴部屋を完全に削除する方式である．この方式では，利用者が 0 人の視聴部屋が存在する可能性が無い．よって利用者のいない視聴部屋の扱いは，完全に削除を行う方式を採用する．削除を行う方法は同様の理由で利用者の削除と同じリストを詰める方式を採用する．

視聴部屋からの利用者の削除と利用者のいない視聴部屋の扱いをまとめた擬似コードをソースコード 5.7 に示す．

```
1  for(i=0;i<視聴部屋の利用者数; i++){
2    if(利用者名==list[視聴部屋][i+2]){
3      if(利用者が一人){
4        部屋を消す;
5        break;
6      }else{
7        利用者をリストを詰めて削除する;
8        在室者名を更新
9        for(var j=0;j<新しい視聴部屋の利用者数;j++){
10         list[視聴部屋][j+2].emit('利用者更新',list[視聴部屋]);
11       }
12     }
13   }
14 }
```

ソースコード 5.7　退室処理

5.5 動作シーケンス

図 5.5，5.6，5.7，5.8，5.9，5.10，5.11 で，設計を行った以下の項目の動作のシーケンス図を示す.

- 新規の視聴部屋への入室
- 既存の視聴部屋への入室
- 再生
- 一時停止
- 停止
- 再生位置変更
- 退室

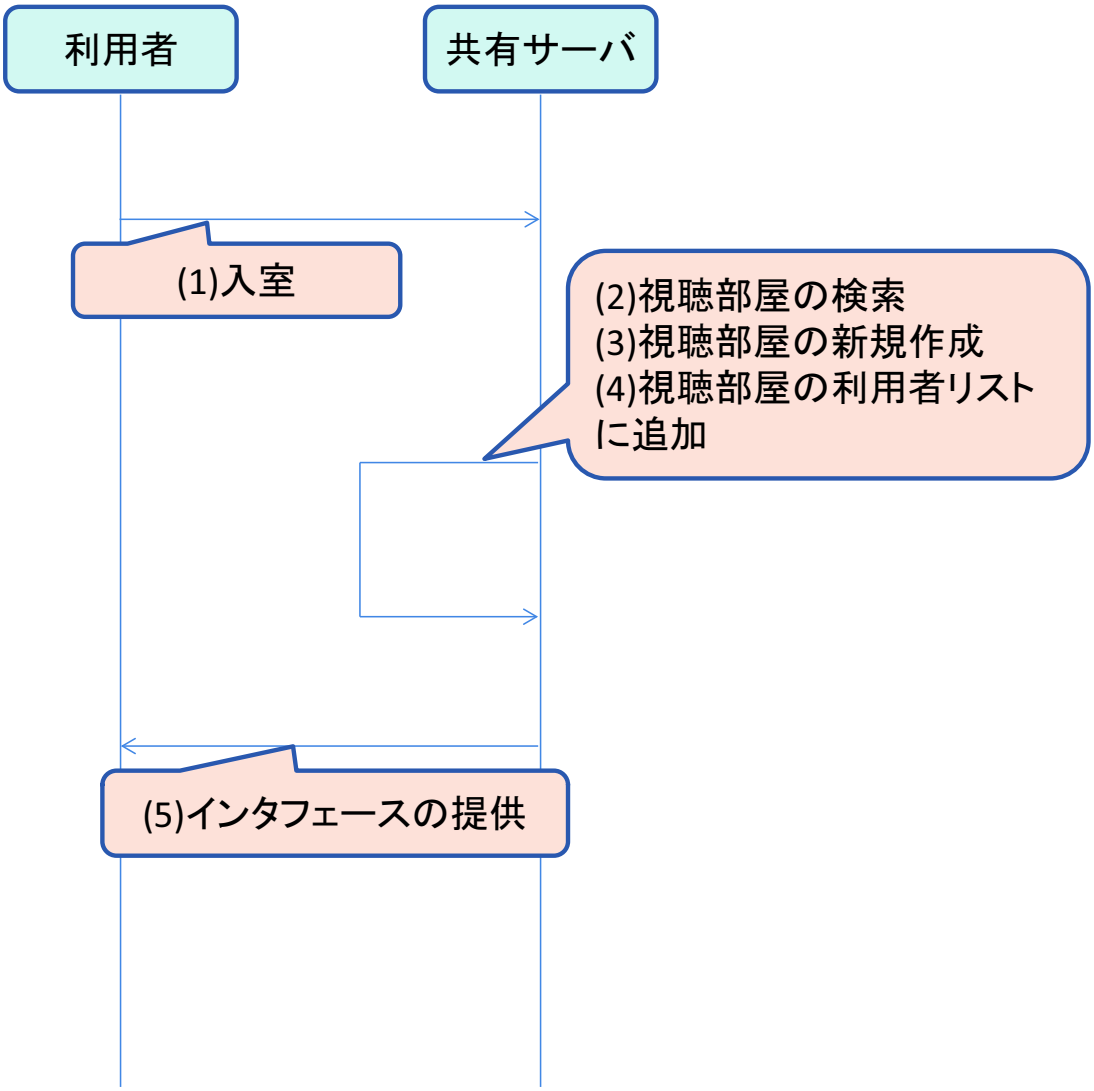


図 5.5　新規の視聴部屋への入室

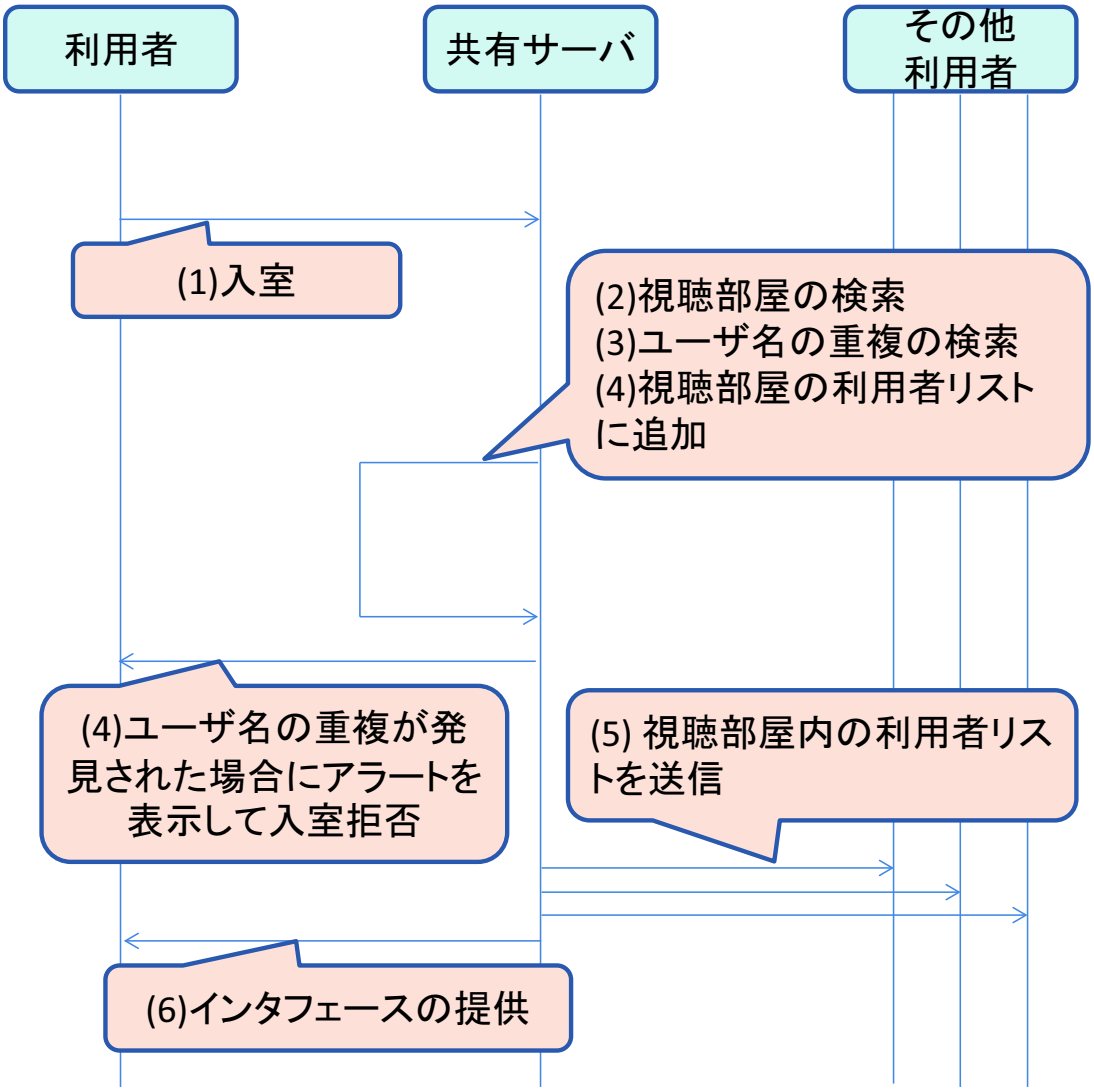


図 5.6　既存の視聴部屋への入室

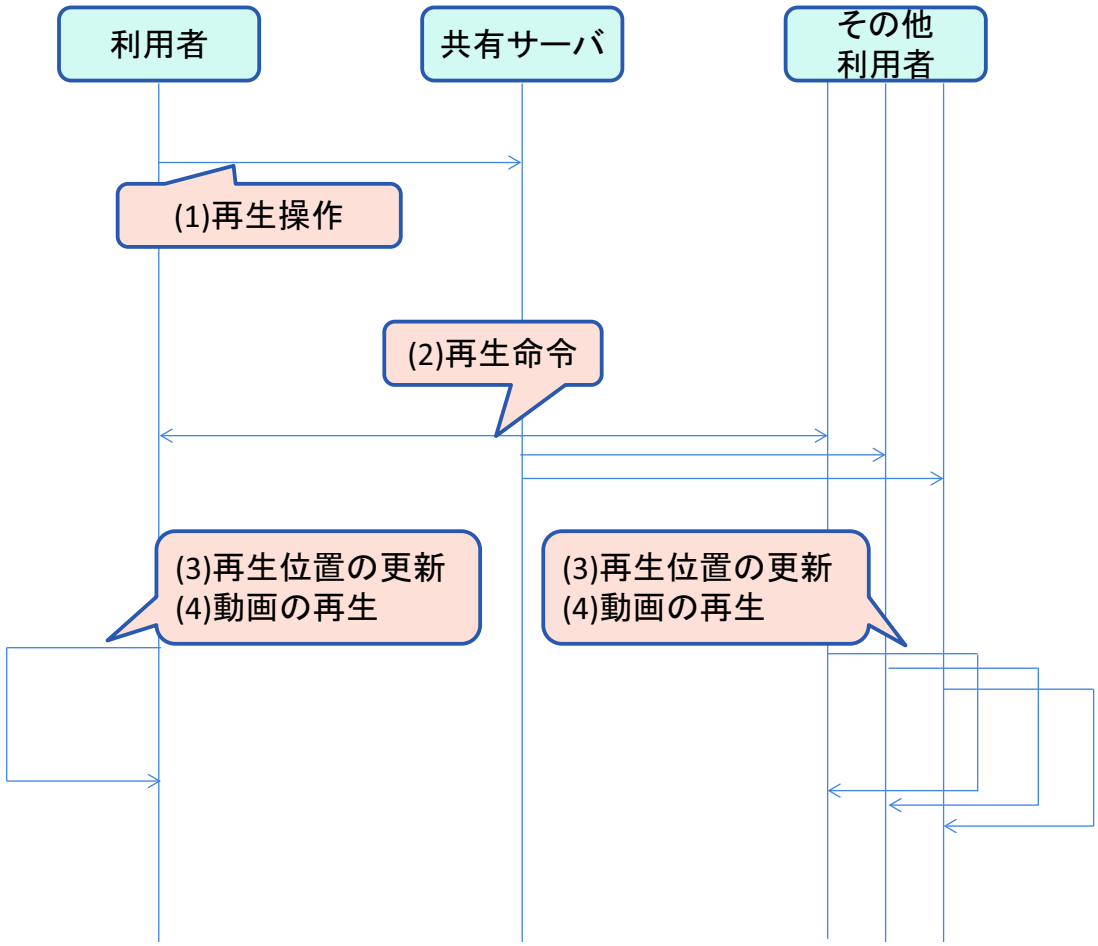

利用者
共有サーバ
その他
利用者
(1)再生操作
(2)再生命令
(3)再生位置の更新
(4)動画の再生
(3)再生位置の更新
(4)動画の再生

図 5.7　再生

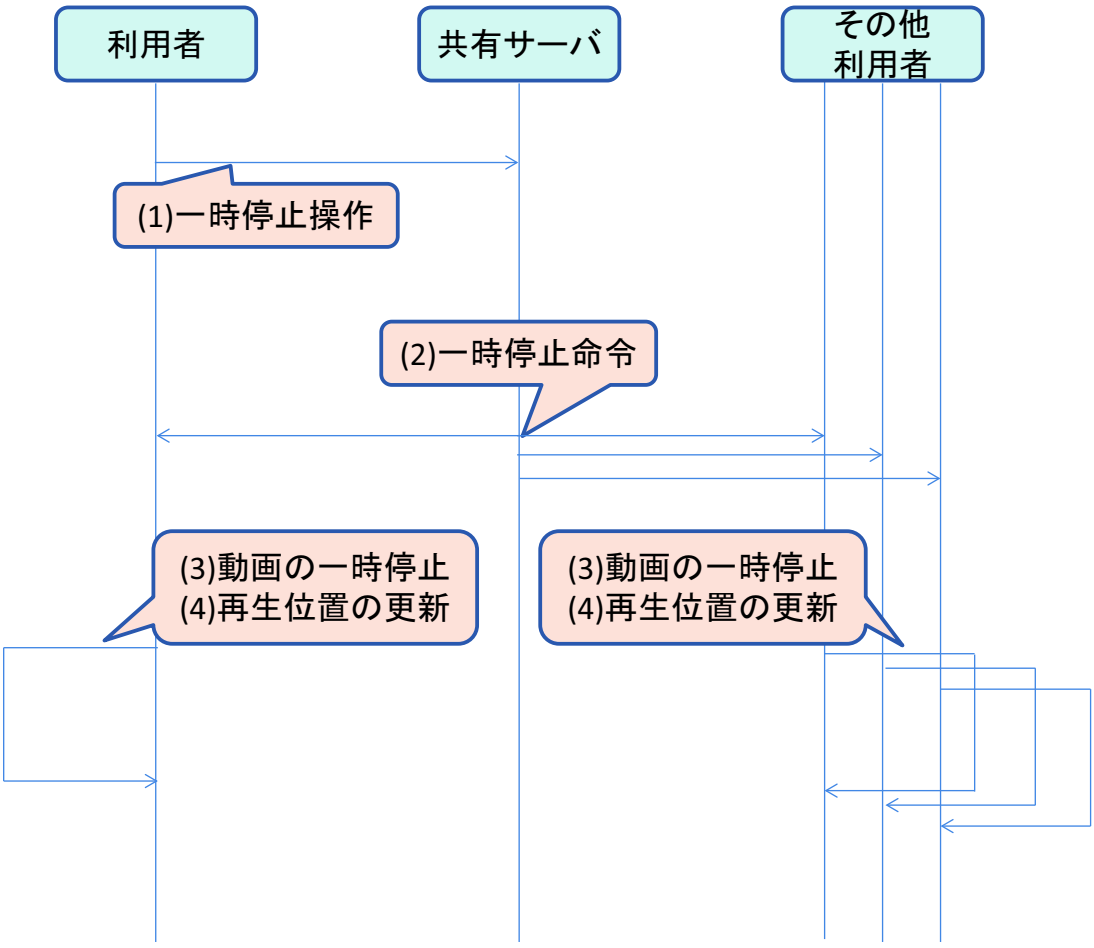

利用者
共有サーバ
その他
利用者
(1)一時停止操作
(2)一時停止命令
(3)動画の一時停止
(4)再生位置の更新
(3)動画の一時停止
(4)再生位置の更新

図 5.8　一時停止

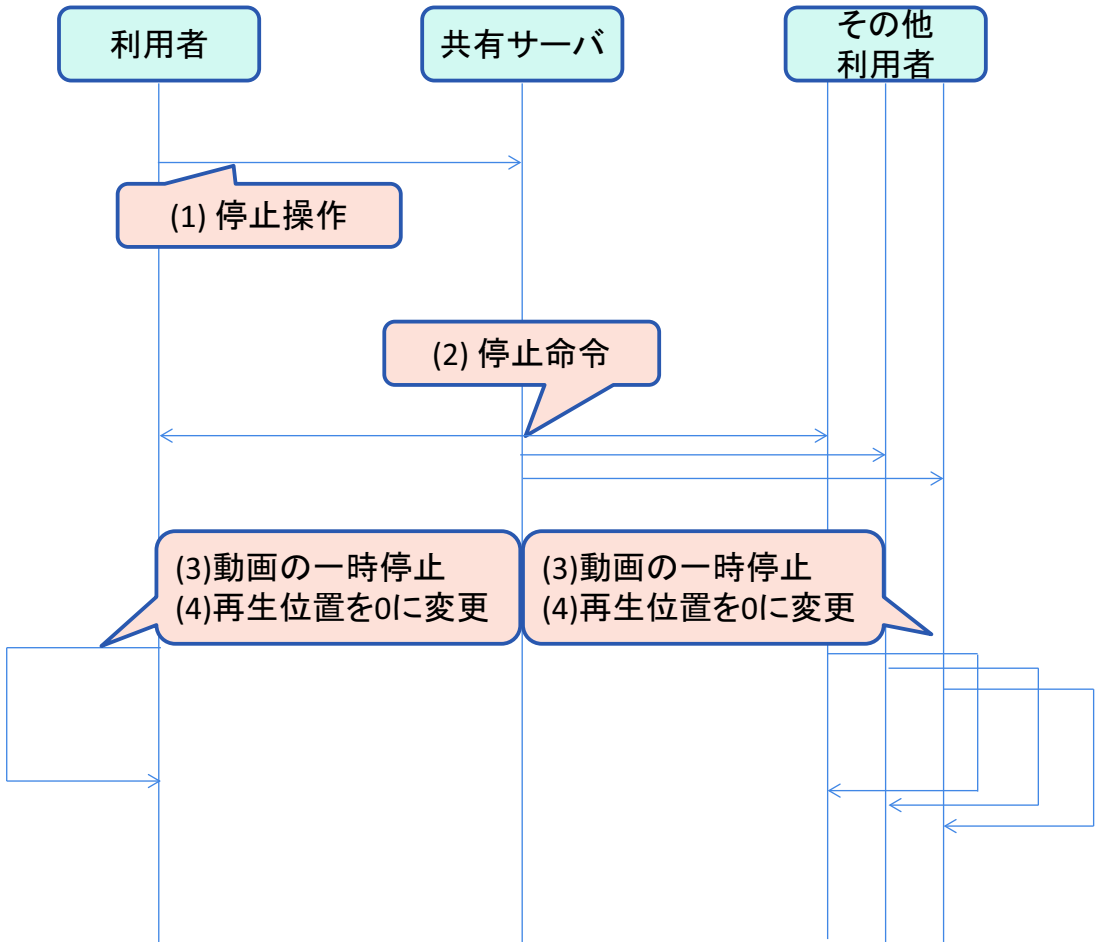

利用者
共有サーバ
その他
利用者
(1) 停止操作
(2) 停止命令
(3)動画の一時停止
(4)再生位置を0に変更
(3)動画の一時停止
(4)再生位置を0に変更

図 5.9　停止

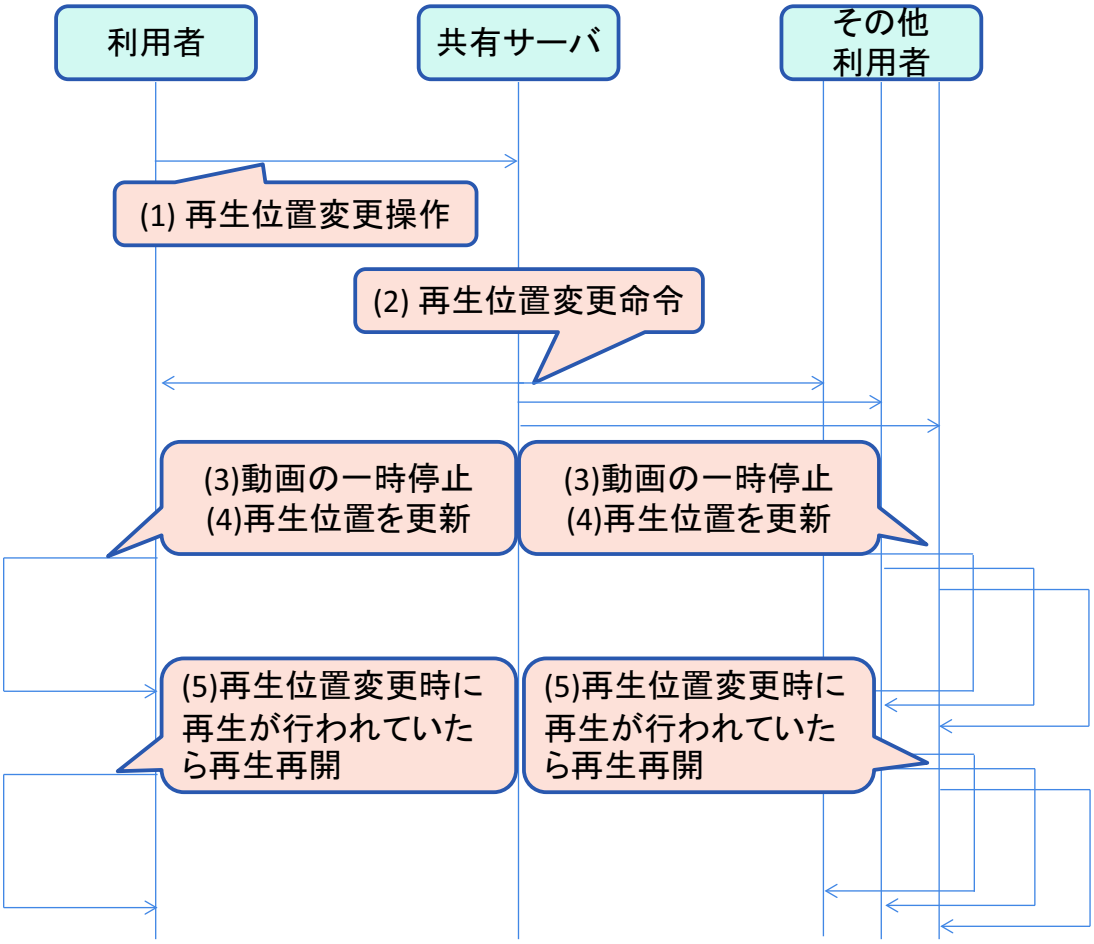

利用者
共有サーバ
その他
利用者
(1) 再生位置変更操作
(2) 再生位置変更命令
(3)動画の一時停止
(4)再生位置を更新
(3)動画の一時停止
(4)再生位置を更新
(5)再生位置変更時に
再生が行われていた
ら再生再開
(5)再生位置変更時に
再生が行われていた
ら再生再開

図 5.10　再生位置変更

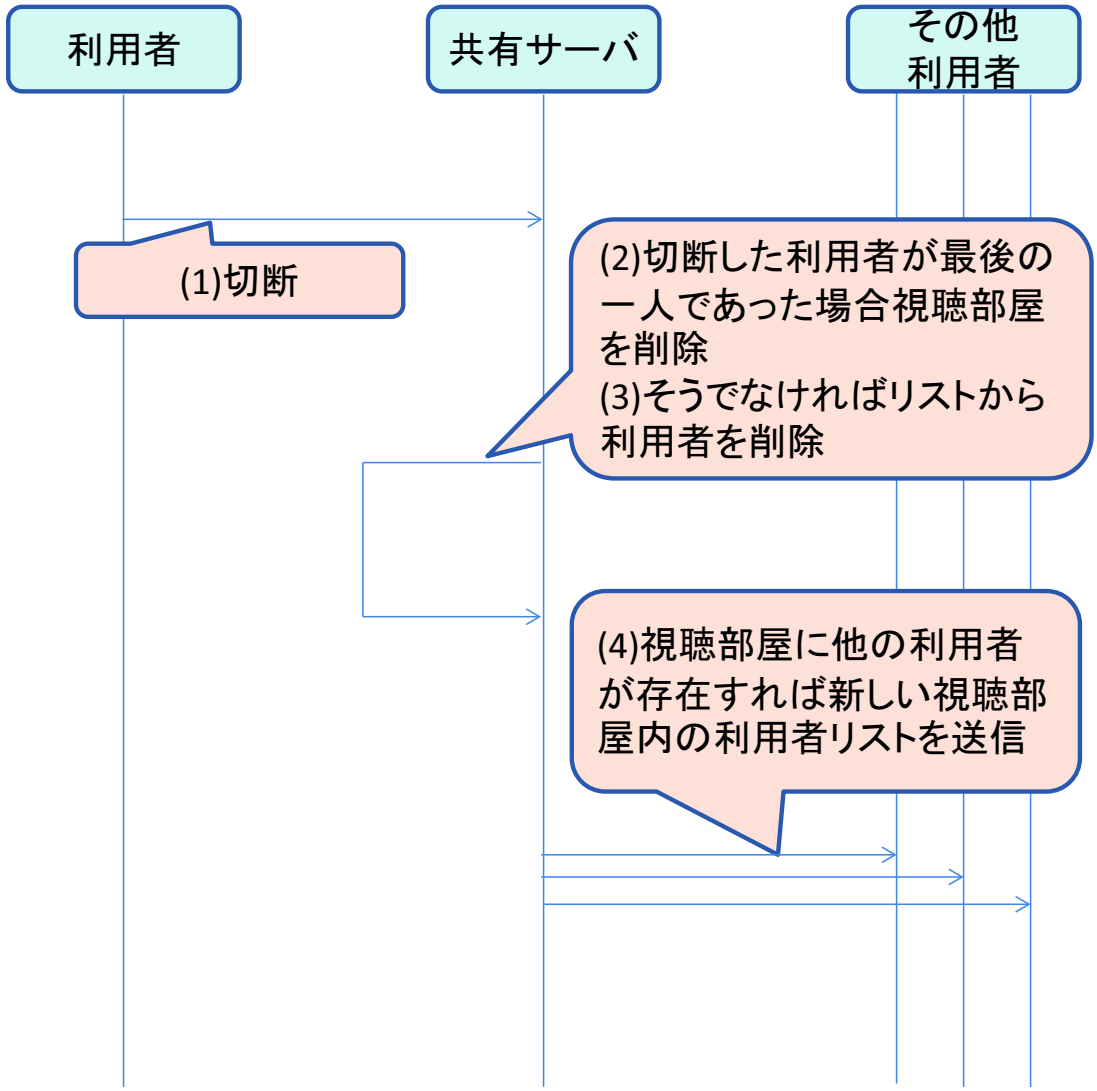

利用者
共有サーバ
その他
利用者
(1)切断
(2)切断した利用者が最後の一人であった場合視聴部屋を削除
(3)そうでなければリストから利用者を削除
(4)視聴部屋に他の利用者が存在すれば新しい視聴部屋内の利用者リストを送信

図 5.11　退室

第 6 章

実装

この章では，本システムの実装について述べる．始めに本システムの実装環境を，そして機能の実装について述べる．最後に本システムのインタフェースを示す．

6.1 実装環境及び構成

本システムの実装環境及び構成を表 6.1 に示す．共有サーバは Node.js で，Socket.io を用いて WebSocket 通信を行うサーバを JavaScript で実装した．動画サーバは，共有する動画に YouTube を使用した場合は YouTube の動画サーバを，HTML5video の機能を用いて再生する動画は本実験では，共有サーバ内に置き実験を行った．クライアントも共有サーバと同様に JavaScript で作成した．双方の実装で共有サーバ，クライアントプログラムの総ライン数は 500 行ほどである．

表 6.1 本システムの開発環境および構成

項目	YouTube	HTML5video
OS	Mac OSX Lion	
共有サーバ	Node.js 0.8.11，Socket.io 0.9.10	
動画配信サーバ	YouTube	共有サーバと同様
開発言語	JavaScript	
API	Youtube Player API	HTML5video API

6.2 使用 API

実装に使用した API を表 6.2 に記載する．使用している各 API は，基本的に同様の項目にまとめることが出来る．これらの API を呼び出すことで，動画の再生共有が行われる．

動画長の取得は，視聴動画の始点から終点までの時間の取得を行う API である．動画読み込み準備完了後の動作は，動画の読み込みが可能になった Web ページで，その直後にどのような動作を行うことが出来るのかを指定できる API である．動画の状態の変更取得は，読み込み完了，読込中，再生，一時停止，再生位置の変更が行われたというこれらの動画の状態を取得するための API である．

表 6.2　使用 API

項目	YouTube Player API	HTML5 video API
再生	ytplayer.playVideo()	video.play()
一時停止	ytplayer.pauseVideo()	video.pause()
再生位置取得 変更	ytplayer.getCurrentTime() ytplayer.seekTo(time)	video.currentTime
動画長の取得	ytplayer.getDuration()	video.duration
動画読み込み準備完了後の動作	onYouTubePlayerReady(playerId)	”loadedmetadata”をトリガしたイベントリスナー
動画の状態の変更取得	onytplayerStateChange(newState)	各イベントをトリガしたイベントリスナー

6.2.1　YouTube Player API

YouTube の場合では，これらの API の使用はすべて関数により行われる．onYouTubePlayerReady(playerId) は，Web ページ上に YouTube の動画プレイヤの読み込みが完了することで呼び出される関数である．読み込み後に必要な動作を関数中に記述することで動作を行う．onytplayerStateChange(newState) は，YouTube の動画の状態が変更されると呼び出される関数である．呼び出しに利用される状態は以下の 6 つである．また，後ろの括弧内の数字は関数の引数である newState に代入される値である．

- 未開始 (-1)
- 終了 (0)
- 再生中 (1)
- 一時停止中 (2)
- バッファリング中 (3)
- 頭出し済み (5)

6.2.2 HTML5 video API

HTML5 video の場合では，再生と一時停止関数を呼び出すことで実行するが，再生位置の取得及び変更や動画長の取得は変数の読み書きにより行われる．YouTube Player API と異なり，再生位置の取得や変更は video.duration という変数の読み書きにより行うことが可能である．また動画読み込み直後の動作および動画の状態の変更取得は，様々な状態で生じるイベントをトリガとして，関数を呼び出すことで実行が可能である．本システムで用いたイベントを以下に示す．

- loadedmetadata
- play
- pause
- timeupdate

loadedmetadata は，表 6.2 に示したように，メタデータを読み込み，動画の読み込み準備が完了した時に発生するイベントである．play は，動画の再生が行われた時に生じるイベントである．pause は，動画の一時停止が行われた時に生じるイベントである．timeupdate は，動画の通常再生や，利用者の動画の再生位置の変更などの，動画の再生位置の変更により生じるイベントである．

6.3 機能実装

第 5 章で行った設計を元に，本システムで実装を行った．

6.3.1 入室処理

入室処理は，ソースコード 5.2，5.3，5.4 の擬似コードに実装を行った．視聴部屋検索の実際のソースコードをソースコード 6.1 に示す．既存の部屋への入室が決定した際に，即時に利用者が必要とする情報は視聴部屋内の動画であるので，視聴部屋検索 RoomSearch() の返り値は，動画 ID もしくは動画のファイル名とした．また，既存の視聴部屋名と利用者の入力した部屋名が一致した際に，5 行目の NewMember() で利用者が入力したユーザ名の検索と新しい利用者の既存の部屋への入室を行う．NewMember の返り値は，ユーザ名の重複が発見されれば false，されなければ視聴部屋に新しい利用者を加え true のブール値を返す．また，ユーザ名の重複が発見されなければ，RoomSearch は，入室を行う視聴部屋の動画 ID または動画ファイル名を返す．そして，利用者が入力した部屋名と既存の視聴部屋が一致しなかった際には，NewRoom() で新しい視聴部屋を

作成し，RoomSerch() は利用者の選択した動画を返す．

```
function RoomSearch(room,roomonline,uname,sentaku,list){
  var i;
  for(i=0; i<roomnum; i++){
    if(list[i][0]==room){
      if(NewMember(i,roomonline,uname,list)){
        num=i;
        return list[i][1]; //ユーザ名重複なし
      }else{
        return -1;//ユーザ名重複あり
      }
    }
  }
  NewRoom(room,roomonline,uname,sentaku,list);
  return sentaku;//部屋の重複なし
}

/*
    roomonline:視聴部屋にいる利用者数
    room:利用者の入力した部屋名
    uname:利用者の入力したユーザ名
    sentaku:利用者が選択した動画の動画IDもしくは動画のファイル名
    list:本システムの利用者と視聴部屋を管理しているリスト
*/
```

ソースコード 6.1　視聴部屋検索

6.3.2　動画の再生共有

動画の再生共有は，ソースコード 5.5，5.6 の擬似コードを元に実装を行った．実際のソースコードは再生命令の送受信を例として示す．

利用者からサーバへの命令の操作では，再生，一時停止，停止の操作命令を Web ページのボタンに addEventListener で click イベントをトリガすることで命令の取得を行う．また，再生位置の変更は，スライドバー上のボタンをクリックしスライドを行い，ボタンを離した位置と動画長の比率より，再生位置の決定を行う．再生位置の変更命令は，ボタンを離した時の mouseup のイベントをトリガとして命令の取得を行う．そして，Socket.io の emit を用いて送信を行う．

```
document.getElementById("start_btn").addEventListener("click",function(){
  if(ytplayer.getPlayerState()==1){//動画の状態が再生なら
    //一時停止命令を共有サーバに送信する
    socket.emit('pause_video',ytplayer.getCurrentTime(),num);
  }else{
    //再生命令を共有サーバに送信する
    socket.emit('play_video',ytplayer.getCurrentTime(),num);
  }
},false);

/*
num:視聴部屋の部屋番号
*/
```

ソースコード 6.2　利用者からサーバへの再生操作命令の送信

YouTube Player API を用いた，利用者からサーバへの再生操作命令の送信をソースコード 6.2 に示す．HTM5 video API を用いた場合，ytplayer.getPlayerState()==1 を video.paused，ytplayer.getCurrentTime() を video.currentTime とすることで実現できる．また，emit で送信を行うイベント名を，再生を play_video，一時停止を pause_video，停止を stop_video，再生位置の変更を change_time とした．

サーバから利用者への操作名の送信では，再生，一時停止，停止，再生位置の変更の全ての操作命令を Socket.io の on で取得を行い，それを視聴部屋内の利用者すべてに送信を行う．また，視聴部屋内の全ての利用者に送信を行うために，ソースコード 6.3 に示したリストのユーザ名と同時にその利用者のセッション ID の管理も行う．そして，そのセッション ID に対し送信を行うことで特定の利用者に送信を行うことが可能である．サーバから視聴部屋内の全ての利用者への再生操作命令の送信をソースコード 6.3 に示す．

```
1  sckt.on('play_video',function(time,num){
2    for(var i=0;i<roomonline[num];i++){
3      io.sockets.socket(list[num][i+2][1]).emit('play_video',time);
4    }
5  });
6
7  /*
8    time:再生ボタンが押された動画の時刻
9    num:視聴部屋番号
10   roomonline:視聴部屋内の利用者数
11   list:本システムの利用者と視聴部屋を管理しているリスト
12 */
```

ソースコード 6.3　サーバから利用者への再生操作命令の送信

利用者での操作命令の適用は，共有サーバより送信されてきたイベントを Socket.io の on で受け，その中のコールバック関数を実行することにより実現する．YouTube Player API を用いた，再生操作命令を受信した利用者の視聴動画の再生をソースコード 6.4 に示す．

```
1  socket.on('play_video',function(time){
2    ytplayer.seekTo(time);
3    c_time();
4    P_flag = true;
5    ytplayer.playVideo();
6    //0.1秒ごとに更新
7    timer=setInterval("c_time()",100);
8  });
9
10 /*
11  time:再生ボタンが押された動画の時刻
12  c_time():現在の動画の時刻をスライドバーに適用を行う関数
13  P_flag:再生位置の変更時に利用する動画が現在再生停止かを示すフラグ
14 */
```

ソースコード 6.4　利用者での再生操作命令の適用

また，インタフェースの動画の再生時刻を表すスライドバーの実装で，HTML5 video API を用いると，動画の再生中は，6.2.2 で示した timeupdate のイベントが定期的に発

生するのでそれをトリガして，動画の現在の再生時刻を取得することでスライドバーのボタンの移動を行なっている．しかし，YouTube Player API では，同様のイベント及び関数が用意されていないため，setInterval を用いて一定間隔で現在の動画の再生時刻を取得することで実装した．

6.3.3 退室処理

退室処理は，ソースコード 5.7 の擬似コードを元に実装を行った．退室処理の実際のソースコードを，ソースコード 6.5 に示す．利用者が共有サーバからの接続を切ると，disconnect のイベントが送信される．共有サーバでそのイベントを Socket.io の on で取得を行う．そしてそのイベントを取得した共有サーバでは，リストより接続の切れた利用者の検索を行う．そしてそのリストより，視聴部屋の利用者の残り人数が 1 人でもいる場合は，splice() を利用して利用者の削除を行う．そして削除を行った視聴部屋の利用者のリストを残りの利用者に送信を行い，インタフェースの視聴部屋内の利用者のユーザ名の一覧の更新を行う．また，視聴部屋の利用者がいないのであれば同様の方法で視聴部屋の削除を行う．

```
1  sckt.on('disconnect', function(){
2    var i,j;
3    for(i=0; i<roomnum; i++){   //退出したIDを検索
4      for(j=0;j<roomonline[i]; j++){
5        if(sckt.id==list[i][j+2][1]){
6          if(roomonline[i]==1){
7            list.splice(i,1);//人が一人なら部屋を消す
8            roomonline.splice(i,1);
9            roomnum--;
10         }else{
11           list[i].splice(j+2,1);   //人をリストから消す
12           roomonline[i]--;   //部屋から人を消す
13           //接続が切れた利用者を消したリストを
14           //視聴部屋のメンバに送信
15           for(var ii=0;ii<roomonline[i];ii++){
16             io.sockets.socket(list[i][ii+2][1]).
17             emit('new_member',list[i],roomonline[i]);
18           }
19         }
20         break;
21       }
22     }
23   }
24 });
25
26
27 /*
28   roomnum:全視聴部屋数.
29   roomonline:視聴部屋ごとの利用者数
30   list:本システムの利用者と視聴部屋を管理しているリスト
31 */
```

ソースコード 6.5　退室処理

6.4 本システムの動作の流れ

ここで，各動画サーバを用いた場合の本システムの流れの説明を行う．また，視聴部屋内にいる本システムの利用者の内，動画の再生停止や再生位置変更を行なっている利用者を操作利用者とする．

6.4.1 Youtube の動画サーバを使用した場合

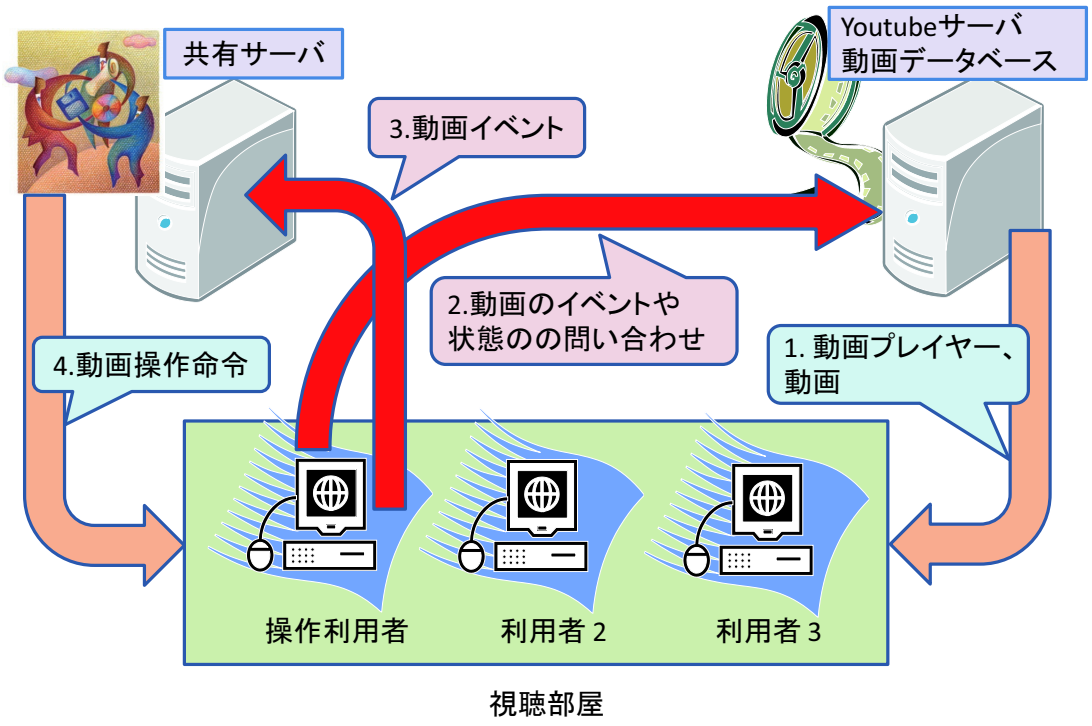


図 6.1 Youtube の動画サーバを使用した場合

Youtube サーバを使用した場合の本システムの流れを図 6.1 に示す．図 6.2 の流れを説明すると以下のようになる

1. 利用者が共有を行う部屋に入室すると，YouTube より動画プレイヤーが入室した利用者のブラウザのページに埋め込まれ YouTube の動画サーバよりプレーヤーに動画が埋め込まれる．
2. 操作クライアントは，動画の再生停止や再生位置の変更を行った際に変更された現在の再生位置などの情報を，YouTube サーバから取得する．
3. 共有を行うために，操作クライアントが取得した再生位置やどのような操作を行ったなどのイベントを共有サーバへ送信する．

4. 共有サーバは，操作クライアントから受信したイベントをすべての利用者に送信する．
5. 視聴部屋内のすべてのクライアントで共有されてきたイベントをもとに動画の再生停止，再生位置変更などの状態の変更を行う

6.4.2 HTML5 video の機能を用いて再生する動画をもつ動画サーバを使用した場合の本システムの流れ

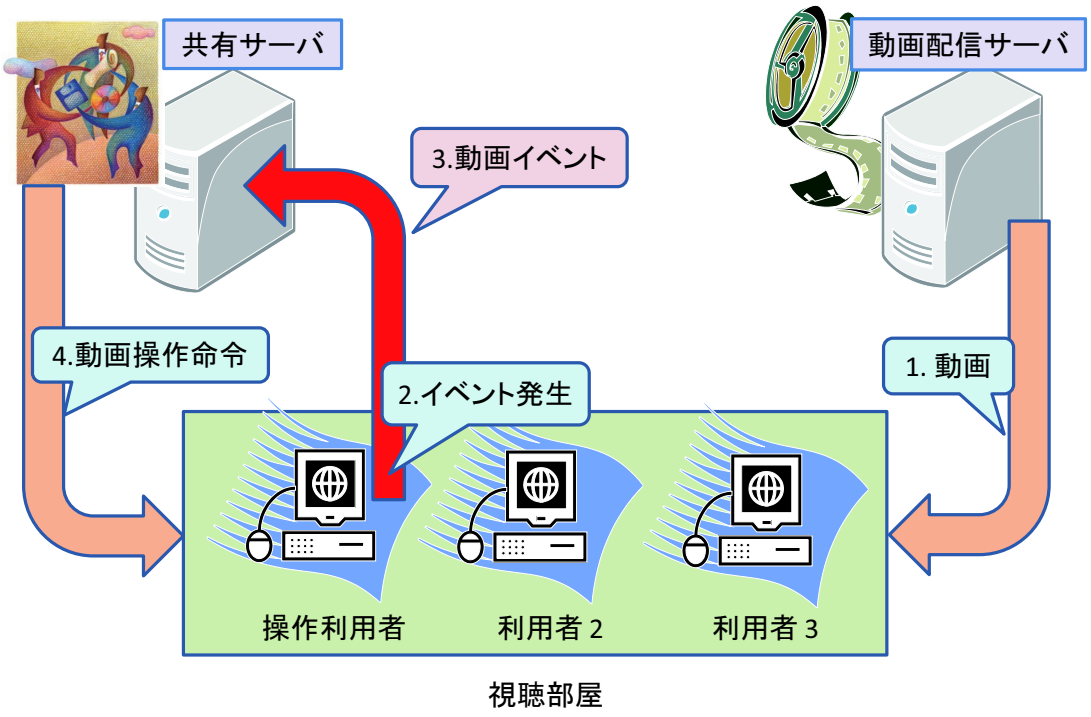


図 6.2 HTML5video の機能を用いて再生する動画をもつ動画サーバを使用した場合

HTML5 video の機能を用いて再生する動画をもつ動画サーバを使用した場合の本システムの流れ図 6.2 に示す．図 6.2 の流れを説明すると以下のようになる．

1. 利用者が共有を行う部屋に入室すると，動画サーバより入室した利用者のプレーヤーに動画が埋め込まれる．
2. 操作クライアントが，視聴中に動画の再生停止，再生位置の変更などを行った際にプレイヤーより発生したイベントを取得する．
3. Youtube の動画サーバを使用した場合と同様に，操作クライアントが共有を行うために取得した再生位置やどのような操作を行ったなどのイベントを共有サーバへ送信する．

4. 共有サーバは，操作クライアントから受信したイベントをすべての利用者に送信する．
5. 視聴部屋内のすべてのクライアントで共有されてきたイベントをもとに動画の再生停止，再生位置変更などの状態の変更を行う．

6.4.3 2 つの流れの相違点

2 つの実装では，使用する動画配信サーバが異なりそれにより動画プレイヤーの扱いが異なる．YouTube の動画を使用する場合には Youtube の動画サーバより動画プレイヤーが送信されページに埋め込まれる．HTML5 video の動画を使用する場合にはブラウザのネイティブの機能の動画プレイヤーで再生が行われる．また，YouTube API を用いて動画情報や動画の状態，イベントを取得する際に YouTube のサーバへアクセスし取得する場合がある．HTML5 video の場合では，動画情報などは事前にプレイヤーに読み込むため，ブラウザのプレイヤーにアクセスすることで取得する．

6.5 インタフェース

本システムのインタフェースを図 6.3，6.4 に示した．はじめに，利用者は共有サーバの URL にアクセスすることで入室画面を表示する．入室画面で「部屋名」「ユーザ名」を入力し動画を選択して送信ボタンを押すことで動画プレイヤーが表示される．一般的なブラウザを対象とした実装なので，図 6.4 に示したアンドロイドスマートフォンなどの携帯端末でも利用することが可能である．また，Youtube の動画を利用した場合と，HTML5 video を Web ブラウザのネイティブの機能を用いて再生を行う動画を利用した場合でインタフェースは同じである．

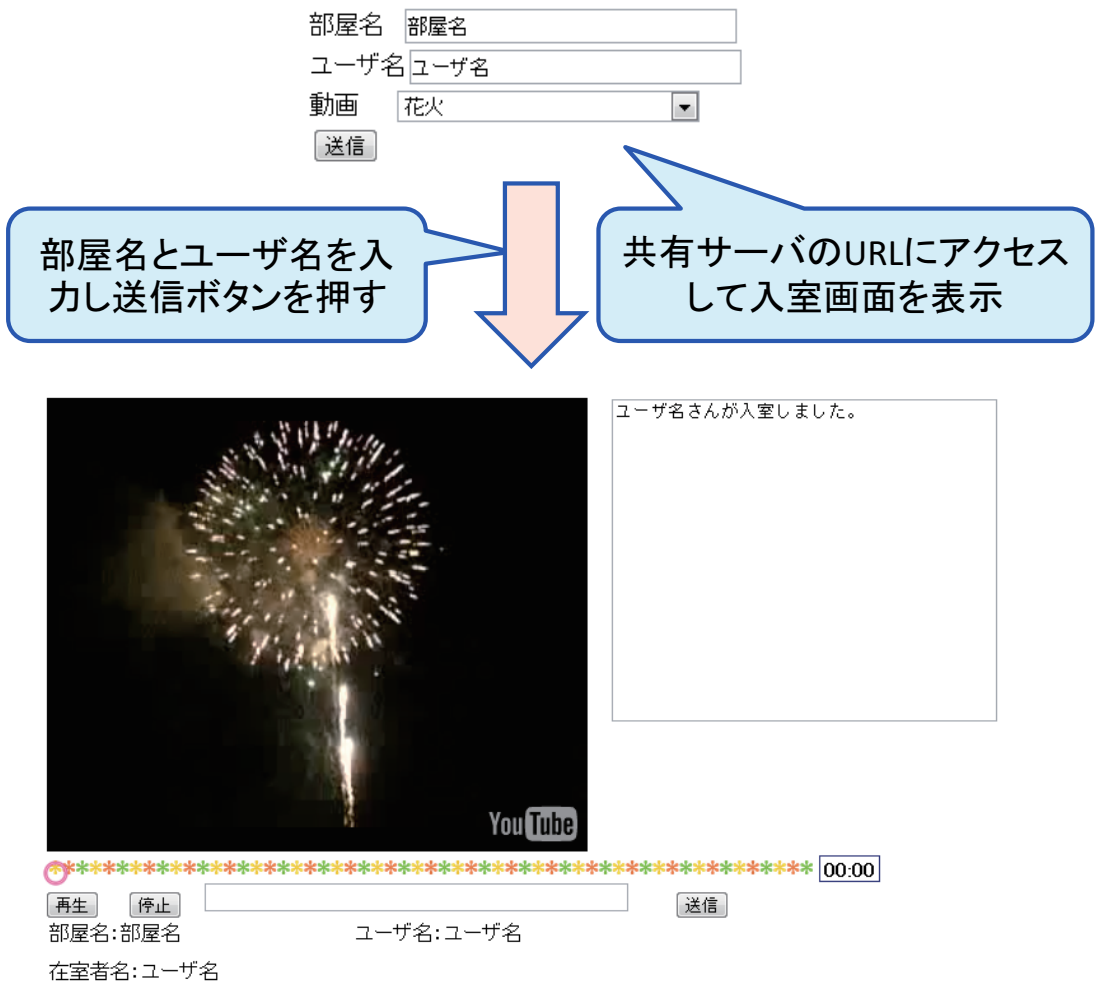


図 6.3　PC のブラウザでのインタフェース

図 6.4　アンドロイドのブラウザでのインタフェース

第 7 章

実験と評価

この章では，本システムで行う実験の概要を述べ，その実験結果について考察を加える．そして，本システムを 2.6 で挙げた利用例に沿って使用した場合の性能評価および，4.1 で挙げた要求条件を達成したかの評価を行う．

7.1 実験

本実験では，本システムの利用者が増加した際に，共有サーバで本システムがどの位のメモリ，CPU を利用するかの測定や，本システムがどの位の利用人数まで耐えうるかの負荷実験を行った．また，視聴部屋内の利用者の一人のネットワークに遅延を掛けた場合に，どれほど動画の再生共有に影響が出るのかの測定を行う遅延実験及び，同期のずれの測定についても記述する．

7.1.1 負荷実験

以下で，負荷実験の実験内容，実験を行った環境，実験結果を述べる．そして実験で使用した負荷機について記述する．

実験内容

本システムを，多くの利用者が利用する状況を想定した負荷実験を行う．実際に多くの利用者で実験を行うのは困難なので，本実験では仮想の利用者として負荷機を作成した．負荷機は，本システムに対して多数の接続を行い，あたかも多くの利用者が動画の再生共有を行なっているかのような状況を作り出す．負荷機の詳しい動作は後述する．

実験を行う環境は，表 7.1 の環境で行った．また負荷機及び測定器は，共有サーバ上で動作させた．

予備実験として，最大利用可能者数の調査を行った．接続している利用者は，今実験では動画の再生共有に関する操作は行わない．視聴部屋の数に応じて最大利用者は異なるので，共有サーバに作成する視聴部屋を 1，5，10，50，100 として全視聴部屋の利用者の人数を足した総最大利用可能者数の再度測定を行った．測定結果を表 7.2 に示す．利用者数

表 7.1　実験環境

項目	仕様
OS	Windows 7 64bit
実装メモリ (RAM)	12.0GB
プロセッサ	Intel Core i7 860
測定機	パフォーマンスモニター

を増加している途中で接続の失敗が目立つようになり，最終的にはサーバの機能が停止したので，接続の失敗が現れたあたりを最大利用可能者数とした．また表 7.2 で，部屋数が 50 と 100 の時の総最大利用可能者数が約 600 となっているが，これは共有サーバの性能の限界ではなく，負荷機の生成できる最大の接続によるものだと考えられる．

表 7.2　部屋数ごとの総最大利用可能者数

部屋数	総最大利用可能者数
1 部屋	約 100 人
5 部屋	約 300 人
10 部屋	約 400 人
50 部屋	(約 600 人)
100 部屋	(約 600 人)

ここで，本システムの共有サーバのメモリ及び CPU の使用量を測定する．メモリ及び CPU の使用量測定のための利用者の利用シナリオとして，表 7.2 で用いた部屋数の場合で，利用者が最大利用可能者数より約を取った数になるまで増加しながら，動画の再生共有を行うことを想定する．また，利用者の入室間隔は 1 秒おきで，入室する視聴部屋はランダムに決定する．

そして，利用者が動画の再生共有を行う際の，再生停止や再生位置変更などの操作量によりどのくらいメモリ及び CPU の使用量が変化するかを測定するため，このシナリオでは入室済みの全利用者が以下の 4 通りの間隔で操作行う．

- 入室直後に 1 回
- 100 秒に 1 回
- 10 秒に 1 回
- 1 秒に 1 回

すなわち本実験では，5 つの異なる部屋数で，4 通りの操作間隔の計 20 回の測定を行う．

実験結果

各実験項目で測定した結果を各部屋数ごとにメモリ使用量と CPU 使用量ごとに分け図 7.1，7.2，7.3，7.4，7.5，7.6，7.7，7.8，7.9，7.10 に示す．メモリおよび CPU の使用量は，5 回測定を行った平均値をグラフにした．メモリの使用量のグラフは，縦軸がメモリの使用量で，横軸が本システムにアクセスしている利用者の人数である．CPU の使用量のグラフは，縦軸が CPU の使用率で，横軸が本システムにアクセスしている利用者の人数である．また，CPU の使用量は，共有サーバに利用しているプロセッサが Intel Core i7 860 のため，仮想 CPU の合算のパーセンテージである 800% が最大値である．

全ての最大視聴部屋数のメモリ使用量及び CPU の使用量が，再生停止や再生位置の変更などの操作間隔が 1 秒に 1 回の際に，利用者の増加に伴い大幅に増加している．はじめは全ての操作間隔で同じ程度だったメモリ使用量及び CPU の使用量が，増加する理由は利用者が増加するにつれて同時期に，動画の再生停止や再生位置の変更などを行う利用者が増加するからであると考えることががができる．逆に，利用者の操作間隔が長い場合では，同時期に動画の再生停止や再生位置の変更を行う利用者が少ない，またはいないためにメモリ及び CPU の使用量の増加が少ないと考えることが出来る．また，最大視聴部屋数が 5 部屋の時は，再生停止や再生位置の変更などの操作間隔が 10 秒に 1 回の際に，1 秒に 1 回の時と同じ程度のメモリ及び CPU の使用量となっている．実際の利用ですべての利用者が 1 秒に 1 回再生停止や再生位置の変更などの操作を行うとは考えることはできず，実際は 10 秒に 1 回より少ない回数で操作が行われるとではないかと考える．実際の利用例を伴った評価は 7.2.1 に後述する．

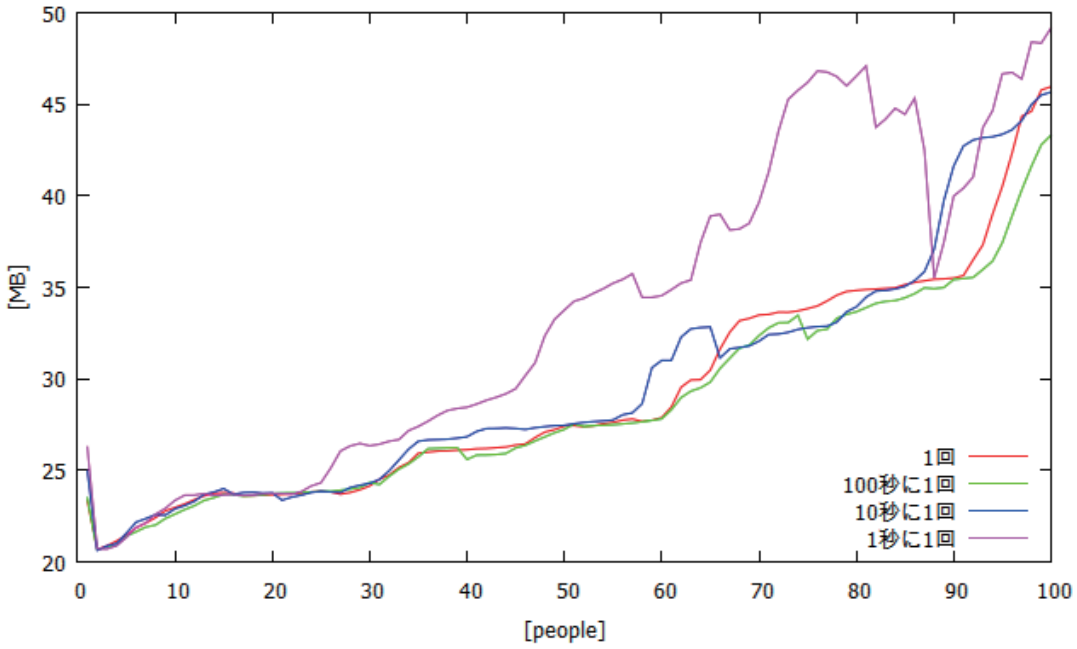


図 7.1　最大視聴部屋数 1 のメモリ使用量

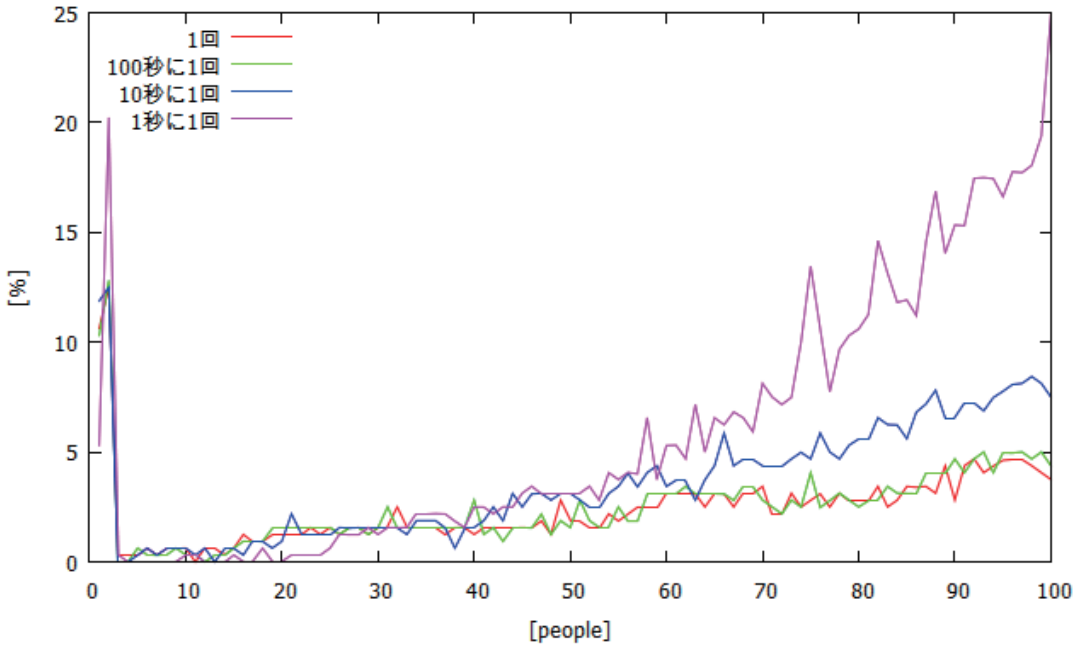


図 7.2　最大視聴部屋数 1 の CPU 使用量

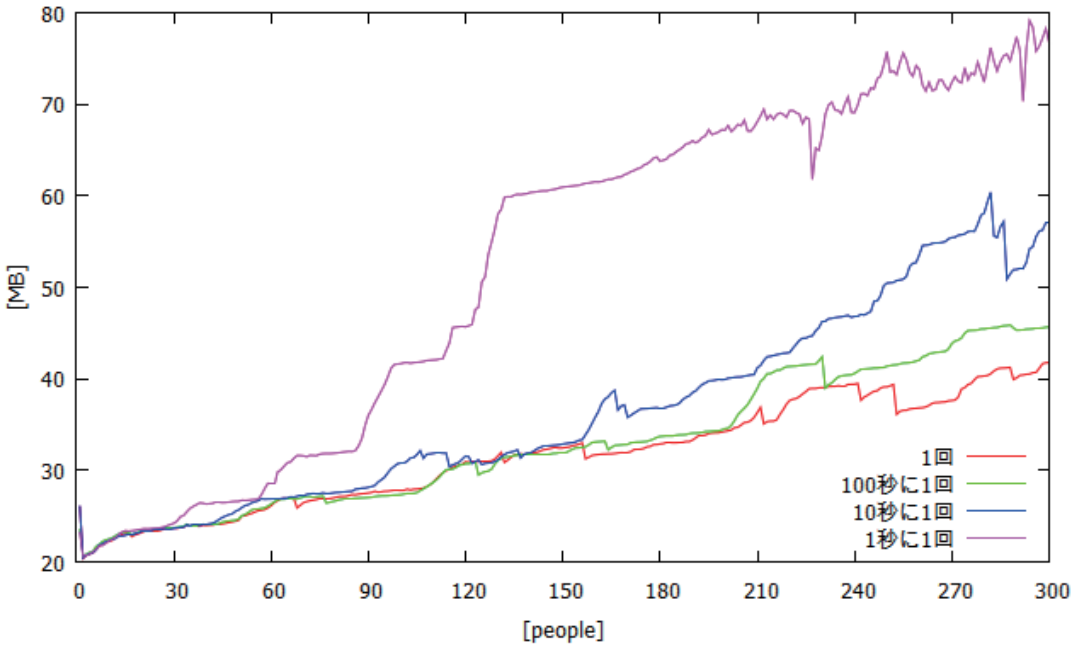


図 7.3　最大視聴部屋数 5 のメモリ使用量

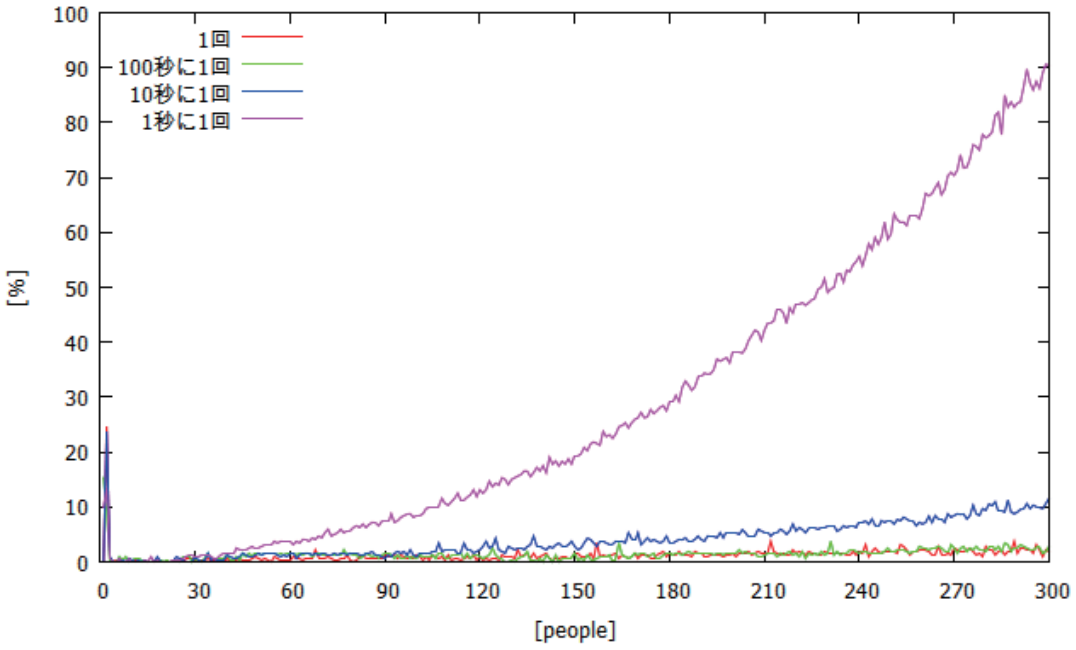


図 7.4　最大視聴部屋数 5 の CPU 使用量

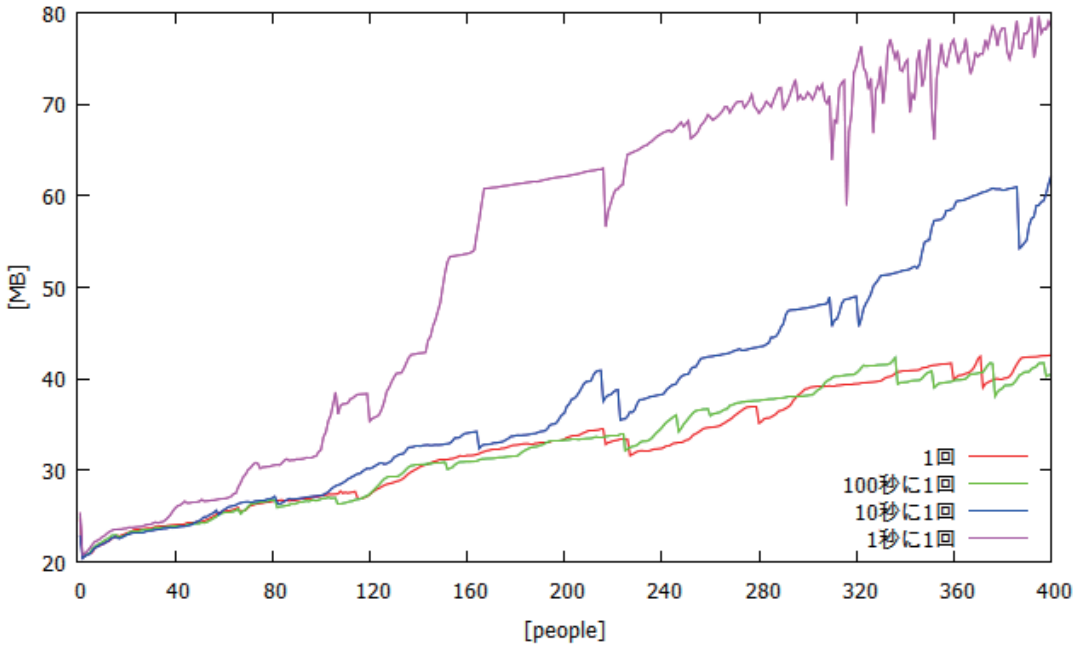


図 7.5　最大視聴部屋数 10 のメモリ使用量

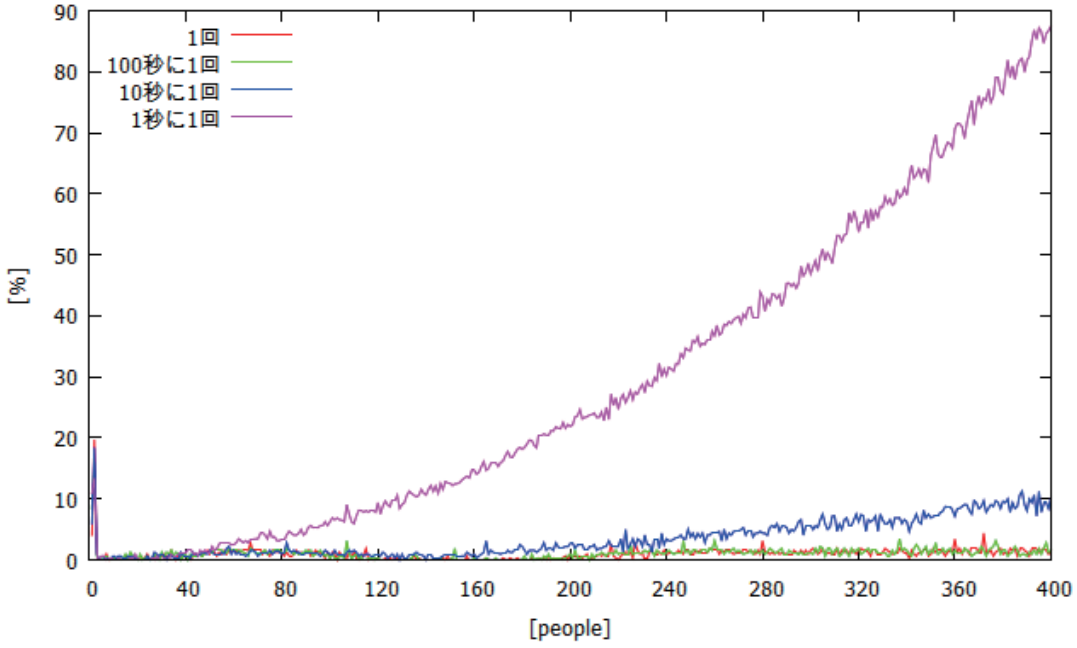


図 7.6　最大視聴部屋数 10 の CPU 使用量

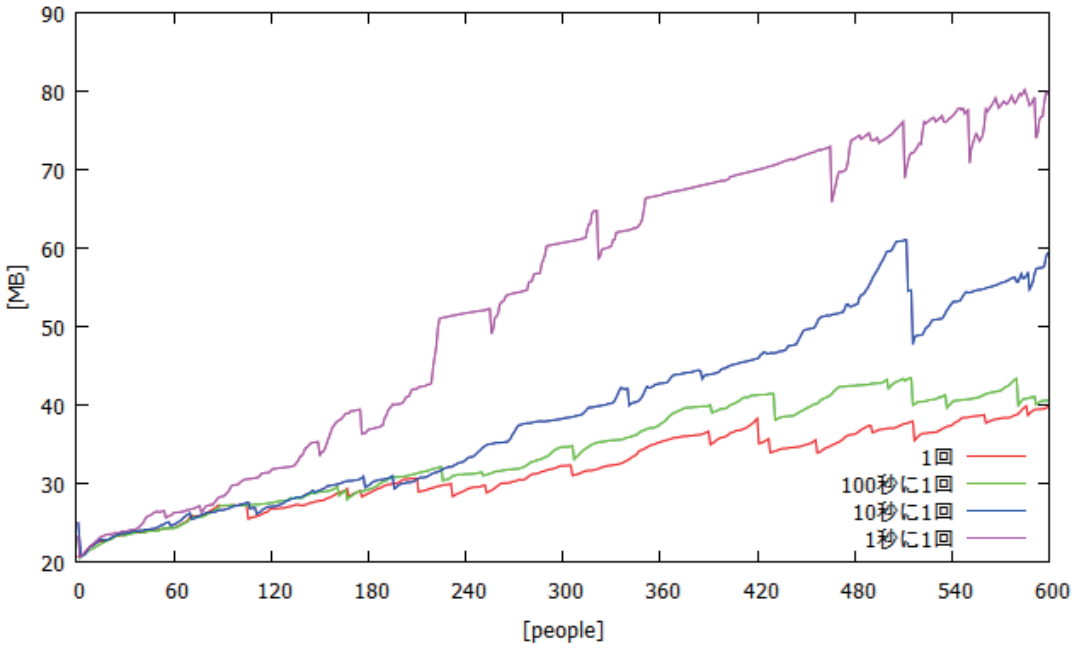


図 7.7　最大視聴部屋数 50 のメモリ使用量

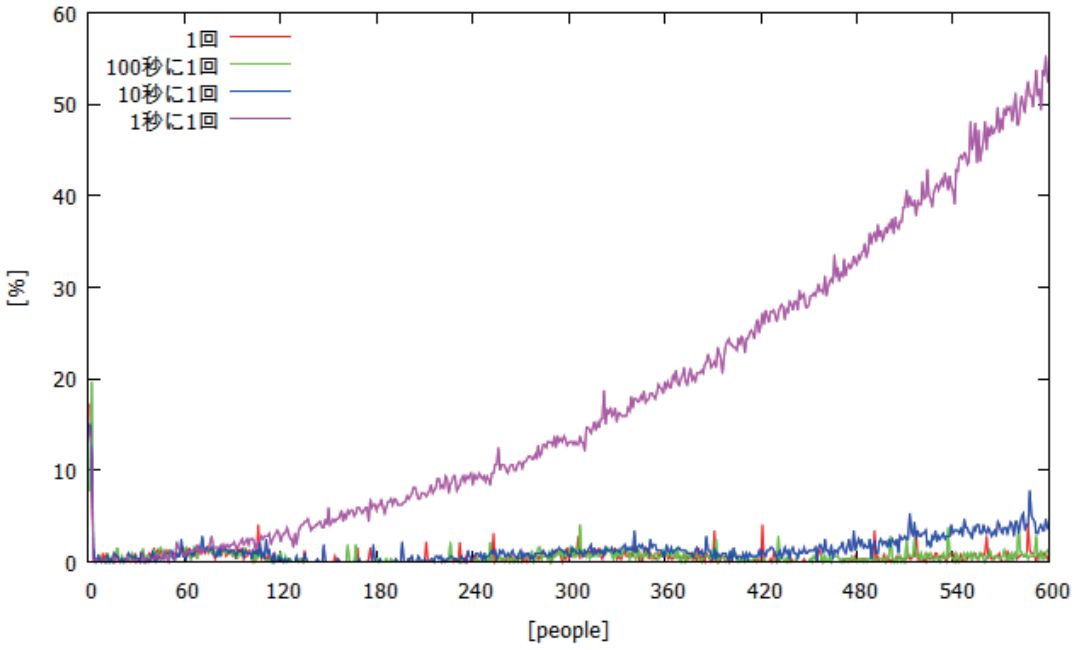


図 7.8　最大視聴部屋数 50 の CPU 使用量

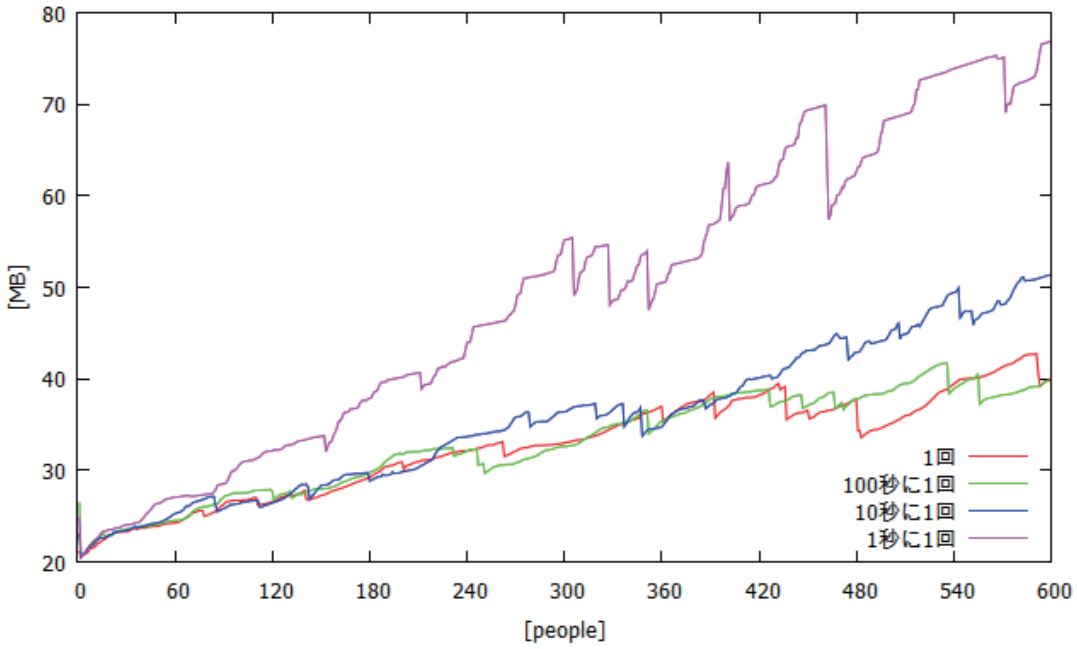


図 7.9　最大視聴部屋数 100 のメモリ使用量

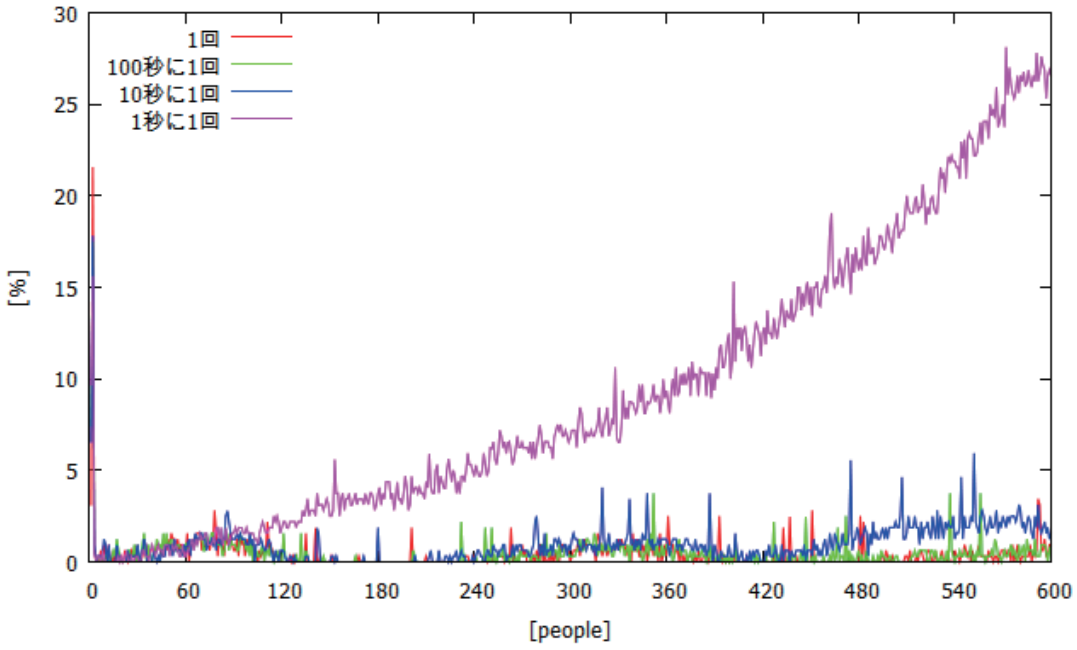


図 7.10　最大視聴部屋数 100 の CPU 使用量

負荷機

本実験では，共有サーバに仮想的な利用者として接続する負荷機を作成し使用した．以下で，負荷機の仕様及び，どのように本システムに負荷を掛けたのかについて述べる．負荷機の開発環境を 7.1.1 に示す．プログラムの規模は合計で 1000 行程度である．

表 7.3　負荷機の開発環境

項目	仕様
OS	Windows 7 64bit
実装メモリ (RAM)	12.0GB
プロセッサ	Intel Core i7 860
開発言語	C#
統合開発環境	Visual C#

本負荷機のインタフェースを図 7.11 に示す．本負荷機の利用は，以下の 7 項目の値を指定し開始ボタンを押すことで開始される．

- アドレス
- ポート番号
- 接続クライアント数
- 最大部屋数
- 入室間隔
- 滞在時間
- 作業回数

接続設定の項目で，アドレス及びポート番号を指定することで接続を行うサーバの決定を行う．クライアント設定で，接続を行うサーバで動作している本システムにどのように負荷を掛けるかの決定を行う．接続クライアント数は，本負荷機が動作している間に何人の利用者がサーバへ接続を行うかの決定を行う．最大部屋数は，本システムの視聴部屋の最大設置数の決定を行う．入室間隔は，利用者がどの位の間隔で入室を行うのかを秒単位で決定を行う．滞在時間は，入室した利用者が何秒間視聴部屋に滞在するかの決定を行う．作業回数は，滞在時間で指定した時間内に何回再生停止や再生位置の変更の操作を行うかの決定を行う．

接続クライアントは，入室間隔の秒数の間隔で，各々に対しスレッドを作成し動作を行う．スレッド内で，仮想の利用者はスレッドの作成順に番号を割り振りそれをユーザ名と

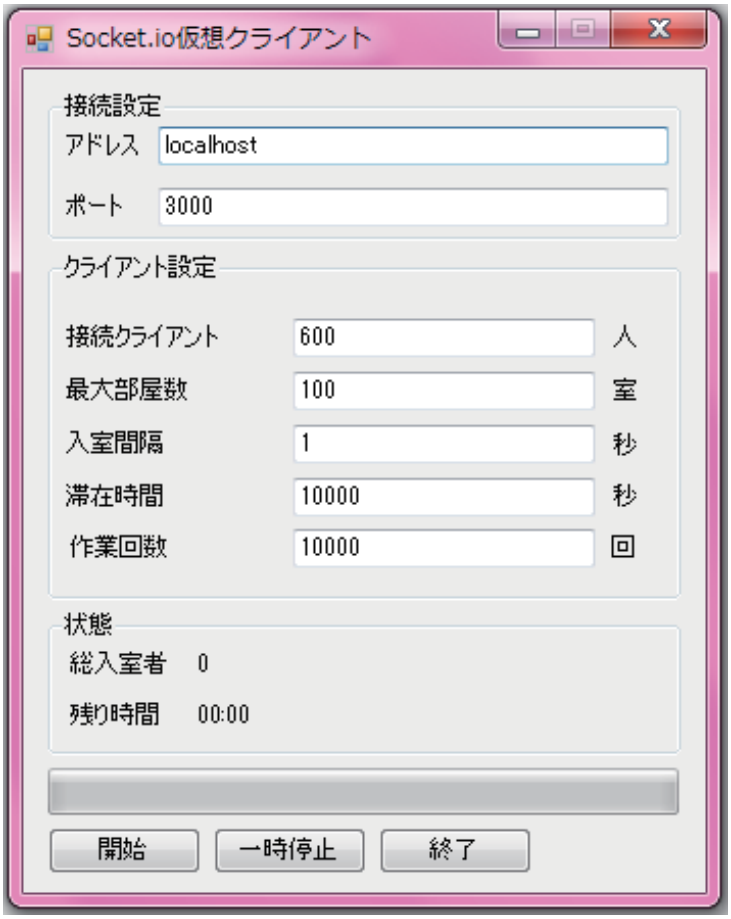


図 7.11　負荷機のインタフェース

して，最大部屋数で指定した数内の数を乱数で生成しそれを部屋名として入室を行う．入室した仮想の利用者は，滞在時間の間に，再生，一時停止，再生位置の変更の 3 つの操作の内 1 つを毎回ランダムで作業回数分だけサーバに送信を行う．そして，滞在時間が過ぎた仮想の利用者は視聴部屋から退室する．

本負荷実験の際は，入室間隔を 1 秒に設定し，滞在時間と作業回数の比より作業間隔を計算し実験を行った．

7.1.2　遅延実験

以下で，遅延の実験内容，実験を行った環境，実験結果を述べる．

実験内容

本システムを利用している利用者の一人の通信が遅れている場合を想定し遅延実験を行い，他の利用者と動画の再生場面が何秒ずれているかの調査を行った．

遅延実験は図 7.12 の様な構成で行った．パケットの遅延が施されたクライアントを「遅延クライアント」として，パケット遅延装置である EthdelayEx version 1.2.2 を用いてパケットを遅延させた．また遅延クライアントは，パケット遅延装置を通じて LAN へ接続されている．そして遅延クライアントが，他のクライアントと動画の同期のずれが生じているかの比較を行うための基準のクライアントを「比較クライアント」とする．比較クライアントは，負荷機とともに共有サーバと同じコンピュータ上でローカルに共有サーバへアクセスする．

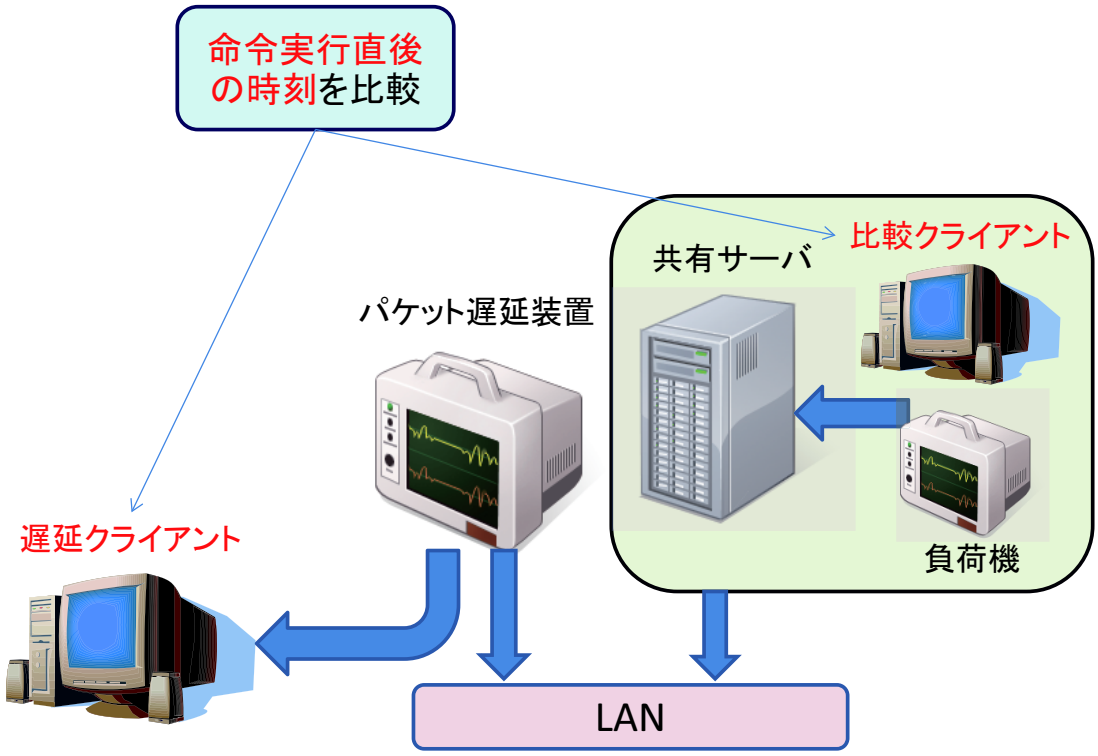


図 7.12　実験構成

負荷機を 28 人，遅延クライアント，比較クライアントの計 30 人の利用者が共有サーバの一つの視聴部屋にアクセスしている状態で実験を行った．また，遅延クライアントのパケット遅延を以下項目に設定した．

- 10ms
- 100ms
- 1000ms
- 2000ms

また，多人数の利用者が存在する場合での同期のずれを調査として，遅延なしの状態で多くの利用者が複数の視聴部屋にバラバラにいる場合と，1 部屋にまとまっている場合で

の調査も行った.

実験結果

遅延実験の測定結果を表 7.4 , 7.5 に示す.

表 7.4　遅延実験の実験結果

パケット遅延	同期のずれ
10ms	1 秒未満
100ms	1 秒未満
1000ms	2 秒未満
2000ms	3 秒未満

表 7.5　遅延なしの多人数参加時の同期のずれ

パケット遅延	同期のずれ
200 人 10 部屋	1 秒未満
100 人 1 部屋	1 秒未満

高野らの研究 [21] では，Steinmets の研究 [22] で示された 2 つの並んだ動画の再生位置のずれが人間にとって違和感のない 120ms を目標に専用のソフトウェアを用いて実装が行われた．しかし本研究では，動画を用いたコミュニケーションを行う上で，コミュニケーションが円滑に行えれば動画の再生位置のずれは厳格に減らす必要はないと考え，場面の移り変わりの激しいスポーツの動画を視聴して電話で会話を行った際に，2 秒以内であればコミュニケーションが円滑に取れるという大塚らの研究の結果を用いる.

よって以上の結果より大幅なネットワークの遅延が無ければ，本システムの動画の再生共有でコミュニケーションを行うことが出来る.

7.2　評価

以下で，2.6 で挙げた実際の利用状況を想定した本システムの性能評価と，本システムが 4.1 で挙げた要求条件を達成したかどうかの評価を行う.

7.2.1 性能評価

以下で，2.6 で挙げた各利用場面で，本システムを利用した際の性能評価を行う．

娯楽の場面での利用

娯楽の場面での利用は，2.6 で挙げた各利用場面で友人間での動画の再生共有が行われる．友人間での動画の再生共有は，1 グループあたり多くても 10 人程度であると想定できる．視聴している動画がスポーツのような動きの激しい動画であるなら，視聴動画の再生停止や再生位置の変更は，局所的な数分の間に頻繁に行われるものと想定することが出来る．また利用者間の物理的な位置関係は分散されていることが想定でき，使用するコンピュータの性能や通信速度もまばらであると考えられる．

7.1.1 の負荷実験の最大視聴部屋数と最大利用者数の関係より，実験に使用した共有サーバ 1 台で少なくとも最大 50 グループの利用者が本システムを利用することが可能である．また遅延実験より，大幅なパケット遅延が発生しない限り，動画の同期のずれは 2 秒以内で利用することが可能である。

教育の場面での利用

教育の場面での利用は，2.6 で挙げた各利用場面で生徒らと教師間での動画の再生共有が行われる．生徒らと教師間での動画の再生共有は，生徒 1 グループあたり教師含めは 30 人程度であると想定できる．映像教材を用いて授業を行なっていると想定すると，視聴動画の再生停止や再生位置の変更は，1 時間に数回とあまり多くないと想定できる．また利用者間の物理的な距離は 1 室内で纏まっていると想定でき，使用するコンピュータの性能や通信速度は比較的一様であると考えられる．

7.1.1 の負荷実験の最大視聴部屋数と最大利用者数の関係より，実験に使用した共有サーバ 1 台で少なくとも最大 10 グループの利用者が本システムを利用することが可能である．また遅延実験の結果より，娯楽の場面と同様に動画の同期のずれは 2 秒以内で利用することが可能である。それに加え，共有サーバをローカル環境で利用すればより動画の同期のずれを減らすことが可能である。

7.2.2 要求条件の達成

以下で，本システムが 4.1 で挙げた要求条件を達成したかどうかの評価を行う．

リアルタイム操作を達成する

本システムを実装したことにより達成されている．各使用者が動画に対し再生停止や再生位置変更などの操作を行うことで他のクライアントにリアルタイムに操作が反映される．

双方向操作を可能にする

の項目は，本システムで実装を行い可能にした．サーバの Node.js がシングルスレッドで動作することで，複数クライアントの動作命令を逐次的に処理を行うため，各動作命令の競合や衝突などをせずに本システムは動作する．

使用者の負担を低減する

一般的な Web ブラウザを用いて実装を行ったことで達成された．本システムの使用者は，特別なソフトウェアのダウンロードやインストールを行うことや，特殊な通信設定を行うことなくインターネットを通じて動画の再生共有を行い他の使用者とコミュニケーションを行うことが出来る．また，本システムは以下のブラウザでの動作を確認した．

- Windows
  - Internet Explorer9
  - Firefox 13
  - Chrome 24
  - Opera 12
  - Safari 5
- Mac
  - Firefox 18
  - Safari 6
- Android
  - Sleipnir(WebKit)

同期のずれを許容範囲に抑える

7.1.2 より，本システムは大幅なパケット遅延がない限り 2 秒以下の同期のずれで利用することが可能である．

また，実際に自宅でサーバを起動し，学校で本システムを 3 人で利用を行った際には，動画を用いたコミュニケーションを阻害するレベルの遅延を感じることはできなかった．

第 8 章

結論と今後の課題

この章では，本研究の結論及び今後の研究を述べる．

8.1 結論

高速なインターネットや高性能なパーソナルコンピュータの普及で様々なファイルやコンテンツが共有できるようになった．特に動画投稿サイトなどに動画を投稿し，使用者たちがその動画に対しコメントを残すことで動画をもとにしたコミュニケーションが行われていた．しかし，それは単にそれぞれが動画を視聴しているだけで，リアルタイムに動画の再生共有を行なってはいない．本稿では，一般的な Web ブラウザを用いた動画の再生共有システムの提案を行った．本システムを用いることで，すべての使用者はリアルタイムに動画に対し再生停止や再生位置変更などの操作を行いながら，お互いにコミュニケーションを取ることが出来る．また，一般的な Web ブラウザを用いることで，使用者は特別なソフトウェアのダウンロードやインストールを行うことなく本システムを用いてコミュニケーションを取ることが可能である．

そして，本システムは，リアルタイム操作の達成，複数の利用者による操作を可能にする，利用者の負担を低減する，同期のずれを許容範囲に抑えるの要求条件を達成し，測定したメモリと CPU の使用量を元に幾つかの利用例で有用性を示した．

8.2 今後の研究

今後の研究として，遅延実験及び遅延実験の評価の実施を行う予定である．

謝辞

本研究を遂行するにあたっては，いろいろな方々にお世話になりました．

まず，指導教員の末田欣子先生には日頃から熱心なご指導，そしてご鞭撻を賜わりました．また，末田先生のみならず多田好克先生にも，ご多忙中にもかかわらず論文の草稿を丁寧に読んで下さり，大変貴重なご助言をいただきました．ここに厚く御礼申し上げます．

そして，本研究が行なえたことは，研究方針や方法論について議論をし，共に研究生活をおくってきた基盤ソフトウェア学講座の学生諸氏のおかげでもあります．また，同末田研究室の中原祥吾には，負荷機の作成に協力して頂きました．最後に，これらの皆さんに感謝いたします．

参考文献

[1] “総務省”, 平成 24 年版 情報通信白書： http://www.soumu.go.jp/johotsusintokei/whitepaper/ja/h24/index.html.

[2] “総務省”, 平成 23 年版 情報通信白書： http://www.soumu.go.jp/johotsusintokei/whitepaper/ja/h23/index.html.

[3] Youtube： http://www.youtube.com/.

[4] “GigaOM”, CEO Larry Page & 7 amazing stats about Google： http://gigaom.com/2012/04/05/ceo-larry-page-7-amazing-stats-about-google/, 2012–04–05.

[5] ニコニコ動画： http://www.nicovideo.jp/.

[6] ニコニコ生放送： http://live.nicovideo.jp/.

[7] USTREAM： http://www.ustream.tv/new.

[8] Skyape： http://www.skype.com/.

[9] “Skype 日本語ブログ”, Skype は 10 年目を迎えました！： http://blogs.skype.com/ja/2012/08/31/happy_9th_birthday_skype.html, 2012–08–31.

[10] Google Maps： https://maps.google.com/.

[11] Pedro Alves and Paulo Ferreira： “ReConMUC – Adaptable Consistency Requirements for Efficient Large–scale Multi–user Chat,” the ACM 2011 conference on Computer supported cooperative work, pp. 553–562, 2011.

[12] 井上恭輔，小野琢也，山下桂司，野田昌弘，岡田正，寺元貴幸：” 君を感じるインターネット –超次元コラボレーションブラウザ「Antwave」という提案–”, 情報処理学会第 47 回プログラミング・シンポジウム報告集, pp. 43–54, 2005.

[13] Dietwig Lowet and Daniel Goergen： “Co–Browsing Dynamic Web Pages,” Proceedings of the 18th international conference on World wide web , pp. 941–950, 2009.

[14] Christian Thum and Michael Schwind： “synchronite– A Service for Real–Time Lightweight Collaboration,” P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), pp. 215–221, 2010.

[15] Carl Gutwin and Michael Lippold： “Real–Time Groupware in the Browser:Testing the Performance of Web–Based Networking,” Proceedings of the ACM 2011 conference on Computer supported cooperative work, pp. 167–176, 2011.

[16] Thomas Sandholm and Hang Maxime Ung： “Turn–by–turn directions go social,” Interacting with Sound Workshop: Exploring Context–Aware, Local and Social Audio Applications, pp. 16–21, 2011.

[17] TWIDDLA : http://www.twiddla.com/.

[18] “IBM”, Lotus Notes : http://www-01.ibm.com/software/lotus/products/notes/.

[19] “Microsoft”, Microsoft Exchange : http://www.microsoft.com/exchange/2010/ja/jp/.

[20] Qiru Zhou and Dong Liu : “Interactive Visual Content Sharing and Telestration: A Novel Network Multimedia Service,” ntelligence in Next Generation Networks (ICIN), pp. 1–6, 2010.

[21] 高野祐太郎他 : ” 投稿動画視聴におけるユーザ間リアルタイムコミュニケーション支援システム”, 電子情報通信学会論文誌 Vol. J93–D No.10, pp. 2302–2316, 2010.

[22] R. Steinmets : “Human perception of jitter and media synchronization,” IEEE J. Sel. Areas Comm., vol.14, no.1, pp. 61–72, 1996.

[23] 大塚雅博他 : “共同体験型コミュニケーションサービスの受容性評価手法の提案” , 電子情報通信学会技術研究報告. CQ, コミュニケーションクオリティ 111(11), 77–82, 2011–04–14.

[24] Windows Media Player : http://windows.microsoft.com/ja-JP/windows/windows-media-player.

[25] “MSDN マガジン”, 実用的なクロスブラウザー HTML 5 のオーディオとビデオ : http://msdn.microsoft.com/ja-jp/magazine/hh781023.aspx, 2012–02.

[26] Node.js 日本ユーザグループ : http://nodejs.jp/.

[27] Socket.io : http://socket.io/.