

複数ビデオソースの動的な構成を可能とする ライブ中継用ミキサ

橋本 浩二^{1,a)} 柴田 義孝¹

受付日 2015年3月27日, 採録日 2015年10月2日

概要：インターネットを利用したライブ中継は容易に実施可能となった。ライブ中継者の数は増加し、その内容も多様なものとなっている。ライブ中継の裾野が広がるにつれ、単一のビデオカメラを用いる簡単な構成のライブ中継だけでなく、複数のビデオカメラを用いた複数ビデオソースによるライブ中継に対しても、より容易に実施可能な仕組みの実現が期待されている。しかし、一般に普及している汎用 PC やスマートデバイスを対象とした複数のビデオソースによるライブ中継を支援するためのミキサの機能は十分には実現されていない。そこで本論文では、複数ビデオソースの動的な構成を可能とするライブ中継用ミキサを提案する。まず、ミキサへの入力となるビデオソースの数の動的な増減に対応するため、ミキサごとに論理的なソースグループを生成できるプロトコルを導入し、そのグループに参加している送信端末からのビデオストリームであれば、動的にミキサへ接続できる仕組みを実現する。また、ミキサの動的なカスケード構成も可能となるよう、ミキサ自身もソースグループに参加できる仕組みを実現する。さらに、ミキサの出力を既存のライブ配信ソフトウェアの入力として扱うための機能も実現する。必要とされる機能の設計とプロトタイプシステムの実装を行い、評価実験を通して、単一構成とカスケード構成のミキサに対するビデオソースの動的な追加機能を確認するとともに、既存の専用機器と同程度の入力数を処理できることを確認した。複数の送信端末とミキサ端末をソースグループとして扱い、ミキサへの入力として動的に追加/削除できる機能は、ライブ中継の実施において有用性があると考えている。

キーワード：ライブ中継システム, 多元中継, AV ミキサ, カスケード構成

Dynamic Configuration Method for Multiple Video Sources on Live Video Streaming Mixer

KOJI HASHIMOTO^{1,a)} YOSHITAKA SHIBATA¹

Received: March 27, 2015, Accepted: October 2, 2015

Abstract: Recently, live video streaming services on the Internet has become easily available. The number of users doing live video streaming is increasing and the contents are very variety. Now, the users of live video streaming services expect that they become easily available only a single video camera but also multiple video cameras. However, useful live mixer functions for supporting live video streaming by multiple video sources are not sufficiently realized on personal computers and smart devices. In this paper, we propose a new live video mixer that allows dynamic configuration of multiple video sources on personal computers. First, in order to support with dynamic changes in the number of video sources as input of the live video mixer, a source group management protocol is introduced for each mixer. Source terminals join a source group and send audio video streams to a mixer terminal. The sending streams are connected to the mixer terminal automatically. A dynamic cascade mixer configuration also becomes possible by other mixers to join to the source group. In addition, the proposed live video mixer includes functions for handling the output of the mixer as an input of existing live video streaming software. In experiments using a prototype system, we confirmed the dynamic configuration method of video sources at both single configuration and cascade configuration. It is also confirmed that the prototype system can handle the number of inputs of the same extent as the existing video mixer devices. The dynamic configuration functions of the live video mixer are very usefulness for supporting live video streaming by multiple video sources.

Keywords: live streaming system, multi-source live streaming, AV mixer, cascade configuration

1. はじめに

インターネットを利用したライブ中継用サービス [1], [2], [3] の台頭と、インターネットの末端において使用可能な通信速度の向上、および個人で利用可能な計算機端末の性能向上により、ライブ中継は容易に実施可能となった。民生品のビデオカメラと汎用 PC、またはスマートデバイス（スマートフォン、タブレット PC）を用いる簡単な構成であれば、業務用の機器を用いなくてもインターネットを利用したライブ中継が可能である。実施されているライブ中継の数は増加し、その内容も多様なものとなっている。ライブ中継の実施に係る敷居が下がり実施者の裾野が広がるにともない、単一のビデオカメラを用いる簡単な構成のライブ中継だけでなく、複数のビデオカメラを用いた複数ビデオソースによるライブ中継に対しても、より容易な実施を可能とするシステムの実現への期待が高まっている。

たとえば、複数の会場で同時に開催されるスポーツイベントや、神輿/山車/踊り手等が長い列を作る祭りのライブ中継を想定した場合、中継元は特定の 1 地点のみでなく物理的に離れた複数の地点となりうる。複数の中継元からのビデオソースをまとめて、最終的にライブ中継用のサービスを用いてストリーミング配信するためには、複数の中継元から送出されるビデオストリームをミキシングしたり切り替えたりする必要がある。このようなライブ中継を実施するためには、通常、専用機器としてのミキサ装置が利用される一方で、一般に普及している汎用 PC やスマートデバイスを対象とした複数のビデオソースによるライブ中継を実施するための、ライブ中継用ミキサに必要とされる機能は十分には実現されていない。そこで本論文では、複数ビデオソースの動的な構成を可能とするライブ中継用ミキサを提案する。

複数のビデオソースをミキシングするにあたり、まず、ビデオソースの動的な数の変化に対応できる機能が必要となる。比較的広い範囲を対象としたライブ中継では、中継元となる地点が時間とともに変化するるので、任意の時間に任意の場所からビデオストリームが送信されることを想定する。それらのビデオソースをすべて、あらかじめミキサへつないでおくことは容易ではない。準備のできたビデオソースを自動的にライブ中継用ミキサへつないだり、中継終了後のビデオソースを自動的に削除したりする機能が有用であると考えられる。

また、マルチアングルのビデオソースを中継元ごとに用意したり、それらを 1 つのビデオソースにまとめたりする

ためには、ミキサのカスケード構成を可能とする機能も必要である。撮影対象をマルチアングルでライブ中継する際には、その中継地点でミキサの機能を利用できることが望ましい。一方、ライブ中継全体として、そのような中継地点が複数になることを想定すると、カスケード構成可能なミキサが必要となる。

さらに、中継元におけるビデオ撮影者とミキサ操作者との間でタイミングをとれる程度のエンド間遅延内でストリーミングとミキシング処理が行われる必要もある。ライブ中継を実施する際、ミキサ操作者は携帯電話等を用いて中継元のビデオ撮影者と連絡をとりながらミキシング操作をすることがある。ビデオソースからミキサまでのストリーミングとミキシング処理による遅延が大きい場合、ビデオ撮影者とミキサ操作者間でタイミングを合わせる際に支障をきたすことになる。

以上をふまえて、複数ビデオソースの動的な構成を可能とするライブ中継用のミキサを開発するにあたり、その要件を次のとおりまとめる。

要件 1：ミキサへの入力となるビデオソースの数の動的な増減に対応できること。

要件 2：ミキサのカスケード構成が可能であること。

要件 3：ビデオソースからミキサまでの遅延がタイミングをとれる程度に小さいこと。

本研究開発の目的は、複数ビデオソースによるライブ中継を、より容易に実施可能とすることである。上述した要件を満たすライブ中継用ミキサを、時間や場所を問わず利用できる仕組みの実現を目指している。本論文では、その一環として、汎用 PC 上で動作するライブ中継用ミキサを開発したので報告する。

以下、2 章では既存技術と関連研究についてまとめる。3 章では、ライブ中継用ミキサを含む提案システムの概要と、複数ビデオソースの動的な構成を実現するための各機能を示す。4 章ではプロトタイプシステムの実装について述べ、5 章ではプロトタイプシステムを用いた評価実験と、その結果をふまえた考察を述べる。最後に、6 章で本論文をまとめる。

2. 既存システムと関連研究

1 章で述べた要件を満たすライブ中継用ミキサに関連する既存システムと研究についてまとめる。

まず、専用機器としてのミキサは、業務用の機器を中心に展開されているが、近年では比較的入手しやすい価格帯の機器も利用可能となってきた。複数のビデオソースを扱うインターネット上のライブ中継でも用いられている。しかし専用機器としてのミキサは、あらかじめ物理的に入力をつなげたうえで利用することを前提としているので、本論文で想定しているライブ中継におけるビデオソースの動的な構成を扱う際は、専用機器のみでは機能的に十分では

¹ 岩手県立大学
Iwate Prefectural University, Takizawa, Iwate 020-0693, Japan

a) hashii@iwate-pu.ac.jp

ない。

複数のビデオカメラを汎用 PC につないで利用できるライブ中継用のソフトウェア [4], [5], [6] も存在する。しかしこれらのソフトウェアも、専用機器としてのミキサ同様、あらかじめビデオカメラを物理的に PC へつないだうえで利用することを想定している。IP カメラと呼ばれるネットワーク対応のビデオカメラを入力として扱える機能を実装しているソフトウェアもあるが、ビデオソースとなる IP カメラの IP アドレスをあらかじめ設定しておく必要があり、ビデオソースの動的な増減に対して柔軟には対応できない。

ここで、本提案システムはビデオストリームの中継とミキシング機能を分散構成するシステムの 1 つとして位置づけられる。複数のビデオソースをミキシング配信する研究として、文献 [7] ではネットワークのバックエンドにおいて重複するストリーム転送を削減するために複数のミキサが用いられている。ビデオ会議システムの研究開発として、文献 [8] では大規模なテレビ会議を想定し、ミキサをカスケード接続したシステムを提案している。また、TV 会議システムの MCU (Multipoint Control Unit) を用いると、機能的にはビデオソースの数の動的な増減に対応できる。本研究開発を進めるうえで既存の TV 会議システムの要素技術は有用である。しかし MCU を含むこれらのシステムを用いて、ライブ中継時のミキシング処理に必要なプレビュー機能や、インターネットを利用したライブ中継サービスとの連携を行うためには機能的な追加が必要となる。加えて、複数地点ごとのミキサと全体のミキサを、ネットワーク環境を考慮してカスケード構成することも容易には実現できない。

一方、インターネットにおける映像や音声データのリアルタイム転送プロトコルとして以前より利用されている RTP [9] には、トランスレータとミキサの機能が定義されている。トランスレータはフォーマット変換等をとまなう中継システムであり、ミキサは複数の送信元からのストリームを 1 つに合成して出力する中継システムである。これらの機能は、ライブ中継においても適用可能な機能であるが、ライブ中継用のシステムの構成方法は RTP では定義されていない。RTP のトランスレータやミキサを利用した過去の研究としては、アプリケーションレベルでトランスレータやミキサの処理を行うメディアゲートウェイシステム [10] や、IP マルチキャストを想定した RTP セッションを複数統合する際にトランスレータの機能を用いる方法 [11] が提案されている。しかしこれらもライブ中継を想定したものではなく、上述した要件を満たすためには、機能的に十分ではない。

さらに、ストリームのモデルとしては、ビデオ会議におけるストリームのルーティングに関する研究 [12] や、P2P のビデオ会議に有効なストリームモデルの提案 [13] がされ

ている。文献 [14] では、帯域幅の制約によるヘテロジニアスな環境を想定したライブビデオ配信の研究も報告されている。これらの研究におけるストリームモデルは汎用モデルとして有用であるが、ライブ中継の実環境を想定し、ミキサのカスケード構成やビデオソースの動的な増減への対応を実現するものではない。

以上、既存システムと関連研究について述べた。1 章で示した要件を満たすライブ中継用ミキサは現在のところ実現されておらず、複数ビデオソースによるライブ中継の容易な実施を支援するためには、新しいシステムが必要であると考えられる。

3. ライブ中継用ミキサの機能

既存システムと関連研究をふまえ、ライブ中継用ミキサに必要な機能について述べる。まず、要件 1 に対しては、ミキサごとの送信端末を管理するための論理的なグループ (ソースグループ) を生成するプロトコルを導入し、そのソースグループに参加している送信端末からのビデオストリームであれば、動的にミキサへ接続できる仕組みを実現する。送信端末は任意の時間にソースグループへ参加と退出ができるものとし、参加している期間にビデオを送信すると、自動的にミキサ端末までストリームが中継され、ミキサ端末上でプレビューされる機能を実装する。また、ミキサの出力を既存のライブ配信ソフトウェアの入力として扱うためのプロセス間通信機能も実装する。要件 2 については、ミキサ端末自身も、後段のミキサにおけるソースグループへ参加できる仕組みにより、カスケード構成を実現する。そして要件 3 については、プロトタイプシステムにおける送信端末とミキサ端末間の遅延の評価を通して、HD 品質 (720p) のミキサが利用できることを示す。

以下、ライブ中継用ミキサとシステムの構成および各機能の詳細を述べる。

3.1 ライブ中継用ミキサとシステムの構成

提案するライブ中継用ミキサとライブ中継システムの構成概要を図 1 に示す。ミキサ端末 (Mixer Terminal) では、送信端末 (Source Terminal) が送信するストリームデー

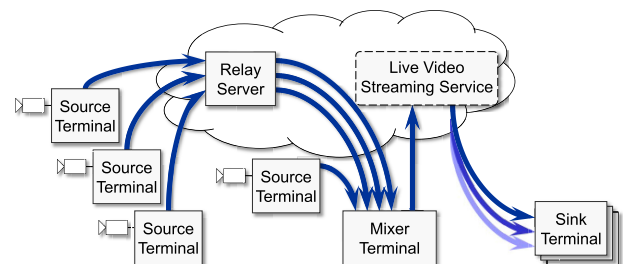


図 1 提案するライブ中継用ミキサとライブ中継システム

Fig. 1 Proposed live video mixer and live video streaming system.

タを受信し、ミキシングや切替え処理を行う。その出力をライブ中継用サービスへ送信し、最終的に受信端末（Sink Terminal）で視聴可能となる。

ミキサ端末では、ビデオカメラとマイクロフォンに加えて、ネットワーク経由のストリームを入力として扱う。キャプチャデバイスからの入力を処理するためには、通常、キャプチャデバイスと端末をあらかじめ物理的に接続しておく必要がある。一方、ネットワーク経由のストリームをミキサの入力として扱う際は、有線/無線を問わず、ストリーム用の論理的なコネクション（ストリームコネクション）を確立して利用する。あらかじめ物理的な接続をしておく形態の入力と異なり、必要なときにストリームコネクションを確立してミキサの入力として扱う機能が実現しやすい。

送信端末は、ビデオカメラとマイクロフォンからの入力データをリアルタイムに圧縮してミキサ端末へ送信する。ここで、送信端末からミキサ端末へストリームコネクションを直接確立できる場合は直接ストリームの転送を行う。一方、送信端末とミキサ端末が異なるプライベートネットワークにつながっていて、ストリームコネクションを直接確立できない等の場合も考慮し、中継サーバ（Relay Server）経由でストリーム転送できるような仕組みを実現する。

そして、ミキサへの入力となるストリーム数の増減に対応するために、ミキサ端末はライブ中継実施時に論理的なソースグループを生成し、送信端末はそのグループに参加したうえでビデオの送信を行う。図 2 は、ソースグループ（Source Group）を含む、システムの基本構成を示している。ミキサ端末が生成するソースグループには、送信端末または前段のミキサ端末が任意の時間に参加/退出できるものとする。ソースグループに参加している端末からのストリームを中継サーバ経由で受信したミキサ端末は、ミキシング処理後のストリームを出力する。その出力先は、後段のミキサ端末またはライブ中継用のソフトウェアとなる。また、ソースグループ内の端末とミキサ端末間で直接ストリームコネクションを確立できる場合は、中継サーバによる中継処理が省略される。

図 2 に示す基本構成をもとに、カスケードミキサの構成例を図 3 と図 4 に示す。

図 3 では、ミキサ端末 M_0 が生成したソースグループに、 $M_1 \sim M_n$ のミキサ端末が参加している。また、 $M_1 \sim M_n$ のミキサ端末もそれぞれソースグループを生成しており、各ソースグループには複数台の送信端末が参加している。ミキサ端末 M_0 と $M_1 \sim M_n$ はストリームコネクションを直接確立できるネットワークにつながっている一方、 $M_1 \sim M_n$ と各ソースグループ内の送信端末間には中継サーバが必要なネットワーク構成となっている。この構成例は、ソースグループを意味的に分ける場合を想定している。た

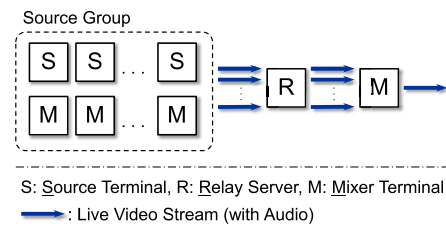


図 2 提案システムの基本構成

Fig. 2 Basic configuration of proposed system.

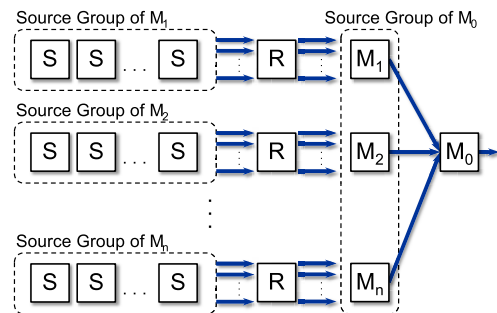


図 3 カスケードミキサの構成 (a)

Fig. 3 Configuration of cascade mixer (a).

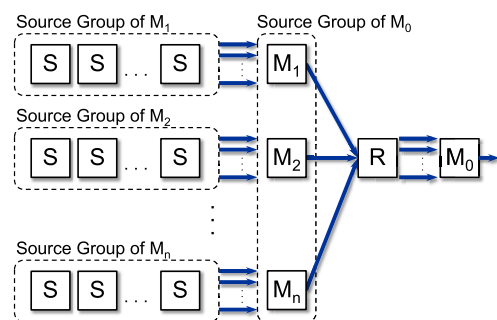


図 4 カスケードミキサの構成 (b)

Fig. 4 Configuration of cascade mixer (b).

たとえば、スポーツの試合において特定の選手ごとにソースグループを用意し、マルチアングルで撮影した複数のビデオを、 M_0 と $M_1 \sim M_n$ が設置されている場所に集め、各ミキサ操作者がミキシング処理を行うような場合である。

図 4 の例でも図 3 同様に、ミキサ端末 M_0 が生成したソースグループへ $M_1 \sim M_n$ のミキサ端末が参加し、 $M_1 \sim M_n$ のソースグループに複数台の送信端末が参加している。図 3 の構成と異なり、ミキサ端末 M_0 と $M_1 \sim M_n$ の間には中継サーバがあり、 $M_1 \sim M_n$ と各ソースグループ内の送信端末間はストリームコネクションを直接確立できるネットワーク構成となっている。この構成例は、ソースグループを物理的に分ける場合を想定している。たとえば、長い列を作る祭りを中継する際、その祭りの中継ポイントとなる複数の地点ごとにソースグループを用意する場合である。中継ポイントごとに $M_1 \sim M_n$ のミキサ操作者は複数のビデオソースをミキシングしたり切り替えたりした結果を物理的に離れた M_0 へ中継サーバ経由で送信し、 M_0 で最終的なライブ中継用のストリームを作ることが可能となる。

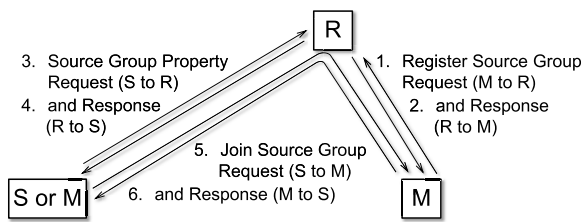


図 5 ソースグループ構成メッセージフロー

Fig. 5 Message flow for configuration of source group.

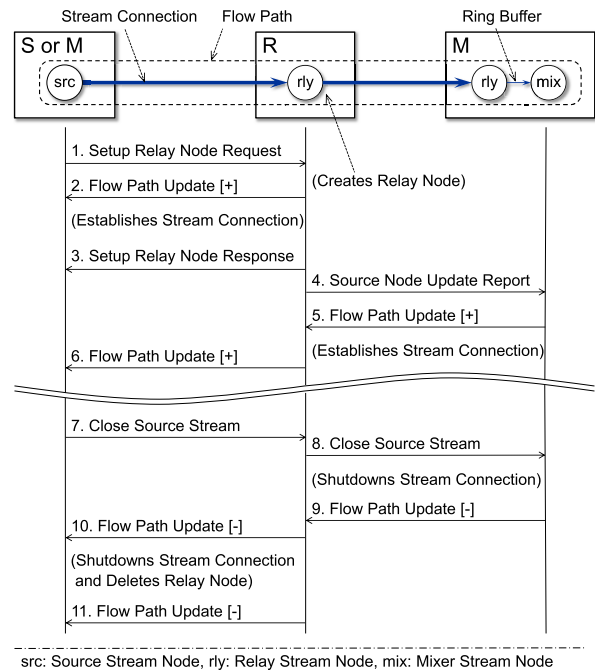
3.2 ソースグループの構成とストリームノードの接続

続いて、ソースグループを構成するためのメッセージフローを図 5 に示す。最初に、ミキサ端末は Register Source Group Request メッセージを中継サーバへ送り、ソースグループに関する情報（ソースグループプロパティ）を中継サーバへ登録する。ソースグループプロパティには、そのソースグループがどのようなライブ中継用のソースグループなのかを説明するテキストが含まれている。また必要に応じて、そのソースグループに参加できる利用者の ID を含めることもできる。一方、送信側の端末（送信端末または前段のミキサ端末）は、Source Group Property Request メッセージを中継サーバへ送信することにより、登録されているソースグループプロパティのリストを取得することができる。そのリストの中から参加すべきソースグループを選び、Join Source Group Request メッセージを中継サーバ経由でミキサ端末へ送る。ミキサ端末では必要に応じて利用者認証を行い、送信側の端末へ参加の可否を送り返す。

一方、図 5 において、ソースグループを生成するミキサ端末と送信側の端末間でストリームコネクションを直接確立できるネットワーク構成の場合は、メッセージのやりとりに関しても中継サーバを必要とせず、Register Source Group Request と Response メッセージは省略される。

送信側の端末がソースグループへ参加すると、その端末はミキサ端末へストリームを送信できる状態となる。図 6 は、ストリーム処理を行うストリームノードと、ストリームノードをつなぐストリームコネクションの確立/解放に必要なメッセージフローを示している。提案システムにおいて、ストリームノードはビデオまたはオーディオストリームの入出力をともなうストリーム処理機能の単位である。ストリームノード間を接続しているのがストリームコネクションであり、ストリームノードとストリームコネクションの一連のまとまりをフローパス (Flow Path) と呼ぶ。

送信側の端末からミキサ端末までのストリームコネクションの確立と解放手順を、図 6 のメッセージフローに沿って示す。まず、送信側の端末は、ストリーム送信を行うソースストリームノード (Source Stream Node) を生成する。次に、Setup Relay Node Request メッセージを中



src: Source Stream Node, rly: Relay Stream Node, mix: Mixer Stream Node

図 6 ストリームノードとストリームコネクション

Fig. 6 Stream nodes and stream connections.

継サーバへ送り、中継サーバ内に中継処理のみを行う中継ストリームノード (Relay Stream Node) を生成する。中継サーバでは、Flow Path Update [+] メッセージを送信側の端末へ送信した後、送信側の端末と中継サーバ間でストリームコネクションを確立する。ストリームコネクションの確立が終了した後、中継サーバは Setup Relay Node Response メッセージを送信側の端末へ送り返す。そして Source Node Update Report メッセージをミキサ端末へ送ることにより、ビデオソースが追加されたことをミキサ端末へ通知する。ミキサ端末では、Flow Path Update [+] メッセージを中継サーバ経由で送信側の端末へ送信した後、中継サーバとミキサ端末間のストリームコネクションを確立する。ミキサ端末内には、ミキシング処理を行うミキサストリームノード (Mixer Stream Node) の前段に中継ストリームノードが生成される。中継サーバ内の中継ストリームノードはストリーム用のパケットを中継するが、ミキサ端末内の中継ストリームノードは、ミキサの入力となるビデオまたはオーディオデータを端末内のミキサストリームノードへ直接渡す。以上が、ストリームコネクションの接続手順であり、解放時のメッセージフローは図 6 の 7～11 に示すとおりである。また、送信側のストリームノードに対して受信側のストリームノードが削除される場合には、受信側の端末から Flow Path Update [-] メッセージを送信側の端末へ送ることによって、フローパスの更新も行う。

ここまで、図 6 のメッセージフローに沿って、送信端末側からストリームコネクションを確立/解放する流れを示したが、ミキサ端末側から確立/解放を行う場合は、Flow Path Update [+/-] メッセージを利用する。図 6 におい

て、Source Node Update Report メッセージを受信したミキサ端末では、そのソースストリームノードからの中継を自動的または選択的に受信することができる。受信する場合は、Flow Path Update [+] メッセージを中継サーバへ送り、ストリームコネクションを確立する。一方、ミキサ端末側からストリームコネクションを解放する場合は、ストリームコネクションを解放した後に Flow Path Update [-] メッセージを中継サーバに送る。

提案システムでは、フローパスとストリームコネクションを論理的に分けて管理することにより、局所的な通信断と復帰にも柔軟に対応できる。想定しているライブ中継では、送信端末が移動することも考慮している。たとえば、送信端末がネットワークへ無線でつながっている場合、突然の通信断へ対応できる仕組みが必要となる。そこでフローパスが確立されていれば、ストリームコネクションの一部で一時的な通信断が発生したとしても、該当箇所のストリームコネクションのみを接続し直すことが可能な設計とした。ここで、ストリームコネクションの一時的な切断後、一定時間以上再接続が行われない場合は、中継ストリームノードとフローパスは解放される。また、ストリームコネクションが確立されていても、何らかの原因により送信端末側からのストリームデータをミキサ端末で受信できない場合も、中継ストリームノードとフローパスを解放する。受信側のストリームノードでは、送信側のストリームノードに対して定期的に受信状況を報告するが、送信側ではその報告を受けていない時間を計測し、受信側ではストリームデータを受信していない時間を計測している。これらの時間に閾値を設け、その閾値を超えた場合、送信側または受信側からフローパスを解放する。その際、ストリームコネクションが確立されていれば、ストリームコネクションを解放した後にフローパスを解放する。正常な処理としては、図 6 に示す Flow Path Update [-] メッセージによりフローパスは解放されるが、メッセージ用のコネクションが利用できない状況も考慮し、送信側と受信側それぞれでタイムアウトによるフローパスの解放が行われる。

3.3 ミキサストリームノードの機能

ミキサ端末における、中継ストリームノードとミキサストリームノードの機能モジュール構成を図 7 に示す。ミキサへの入力の動的な増減へ対応するため、ミキサストリームノードでは入力用のリングバッファの追加/削除を可能としている。リングバッファは、ミキサストリームの入力となる中継ストリームノードごとに生成される。中継ストリームノードでは、受信したパケットからビデオフレームデータまたはオーディオサンプルデータを作り、デコードする。その後、デコードされたデータをプレビューによりプレビュー表示するとともに、リングバッファへもデータをコピーする。ミキサストリームノード内では Audio

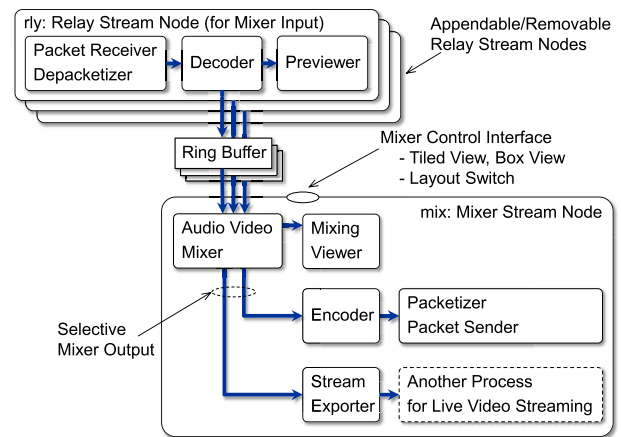


図 7 ミキサの機能モジュール構成

Fig. 7 Functional module configuration of mixer.

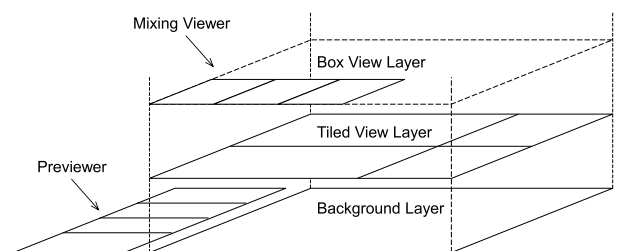


図 8 プレビューとミキシングビュー

Fig. 8 Previewer and mixing viewer.

Video Mixer モジュールがリングバッファからデータを読み込み、ミキシング処理を行い、その結果は Mixing Viewer に表示される。

ミキシング処理後のデータを、さらに後段のミキサへ送信する場合は、Encoder モジュールによりエンコードされたデータがパケットに分割され、送信される。一方、ミキシング処理後のデータをライブ中継用ソフトウェアやライブ中継用サービスへ送信する場合は、Stream Exporter モジュールを利用する。

ミキシング操作は、ミキサストリームノードの Mixer Control Interface を通して行われる。ビデオをタイル状に並べる Tiled View と、縦または横に並べる Box View をレイアウトの基本とし、そのレイアウトを切り替える機能も実現する。図 8 は、ミキサ端末上でミキシング操作をする際の空間的な配置を示している。中継ストリームノードでデコードされたビデオデータはプレビュー (Previewer) 上に自動的にプレビュー表示されるとともに、ミキシング処理の入力データとして利用可能な状態になる。ミキサ端末操作者は、プレビューに表示されたビデオを確認し、必要に応じてミキシングビュー (Mixing Viewer) 上でミキシング処理の対象とすることができる。ミキシングビューは、背景を表示する Background Layer 上に Tiled View Layer と Box View Layer が重なる 3 層構造となっている。Box View Layer は、Tiled View Layer 上で Picture-in-Picture の機能を提供する層である。

中継ストリームノードが削除された場合は、ビューワにおける表示も自動的に除去されて背景が表示される。ライブ中継の内容にもよるが、予測が困難な原因による中継ストリームノードの削除が発生することを想定すると、今まで表示されていたビデオが突然背景表示に変わることは望ましくない。一方、数秒間のビデオの停止は許容できることがある。これらを考慮し、中継ストリームノードの削除時には、最後に表示していたビデオフレームを一定時間表示させる。その時間は設定値として与える。この機能により、中継ストリームノードの削除後、背景が表示されるまでの間に、ミキサ端末の操作者にはミキサを操作する時間的な余裕ができる。

以上、提案するライブ中継用ミキサの機能を述べた。引き続き、プロトタイプシステムの実装についてまとめる。

4. プロトタイプシステム

複数ビデオソースの動的な構成を可能とするライブ中継用ミキサに対して、実機を用いた機能的な検証を行うため、Windows 7を搭載した汎用 PC 上にプロトタイプシステムを実装した。

(1) メッセージ通信と利用者認証

提案するシステムは分散システムであり、システムを構成する各端末は、図 5 と図 6 に示したメッセージを含む多くのメッセージを相互にやりとりする必要がある。プロトタイプシステムでは、TCP 上にテキストベースの簡単なメッセージプロトコルを実装した。また、送信側の端末がソースグループへ参加する際には、必要に応じて利用者認証を行えるよう、企業内ネットワーク等で利用されている LDAP (Lightweight Directory Access Protocol) を用いた利用者認証機能を適用した。プロトタイプシステムでは Windows 用の Open LDAP [15] を中継サーバまたはミキサ端末上で動作させ、メッセージ通信用の TCP コネクションを確立する際に、利用者認証を可能としている。

(2) ストリームノードおよびミキサの実装

各ストリームノードは、既存システム [16] のストリーミング機能を改良して実装した。ビデオとオーディオの伝送には RTP を基本とした簡易ストリーミングプロトコルを、TCP または UDP 上で動作するよう実装し、ストリームノード内部におけるメディア処理には Direct Show [17] を用いている。ビデオおよびオーディオのミキシング機能も既存システム [16] におけるミキシング機能を改良して実装した。文献 [16] のミキサは、図 8 に示す 3 層構造には対応していない。また、入力 of 動的な追加/削除機能も実装されていないので、動的に追加/削除可能かつスレッドセーフなリングバッファをミキサの入力としてつなぐ仕組みを新たに実装した。図 7 の Audio Video Mixer は、リングバッファから一定間隔 (評価実験では、ビデオは 1/30 sec., オーディオは 1/25 sec.) でビデオとオーディオのデータを

取得する。その際、時間的な揺らぎを吸収するため、リングバッファ内でバッファリング (ビデオは 2/30 sec., オーディオは 2/25 sec.) を行っており、その分はミキシング処理の遅延となる。一方、ミキサの入力となる中継ストリームノードとミキサストリームノードにおけるミキシング処理を異なるスレッドに分けることにより処理の並列性を高めている。また、ビデオデータのミキシングの際は、ビデオデータを RGB (各 8 bit, 計 24 bit/pixel) に統一し、オーディオデータは PCM (サンプリング周波数 48 kHz, 量子化ビット数 16 bit) に統一した入力データをミキシングしている。

また、図 7 における Stream Exporter モジュールでは、ローカル TCP コネクションを用いて、ビデオとオーディオデータを他のプロセスへ渡す仕組みを実装した。プロトタイプシステムでは、ライブ配信ソフトウェアの 1 つとして Microsoft Expression Encoder 4 [18] を想定し、Microsoft Expression Encoder 4 における入力として利用できる DirectShow のソースフィルタも実装した。そのソースフィルタと Stream Exporter モジュール間でローカル TCP コネクションを確立し、ミキシングされたビデオとオーディオデータを Microsoft Expression Encoder 4 の入力としている。

(3) ビューワと操作用 GUI

ビューワと操作用の GUI は、Java 言語を用いて実装した。図 8 で示したプレビューワとミキシングビューワを、図 9 に示すとおり実現した。Tiled View Layer では、列と行を任意の数に変更したり、表示区画をまとめたりすることができる。Box View Layer では、縦また横に並べたビデオをまとめて、配置位置を変更したり大きさや透明度を変更したりすることが可能である。図 10 はミキシング操作用 GUI の画面イメージである。図 7 における Mixer Control Interface を用いて、ビデオに関しては Tiled View および Box View の操作とレイアウトの切替えが可能であり、オーディオに関しては音量調整が可能である。

Windows 以外のプラットフォームへの移植も考慮し、GUI は Java 言語を用いている一方、ストリームの送受信やミキシング処理には、パフォーマンスを上げるため C 言語

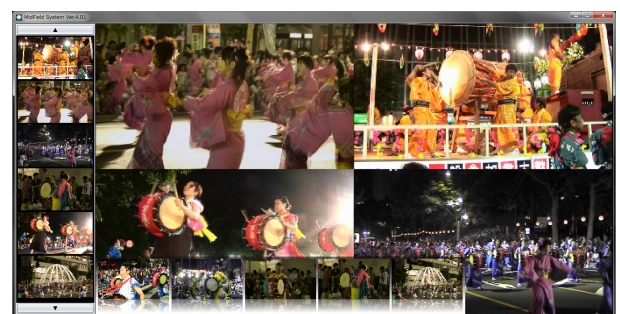


図 9 ミキシング操作用ビューワ

Fig. 9 Viewer for mixing operation.

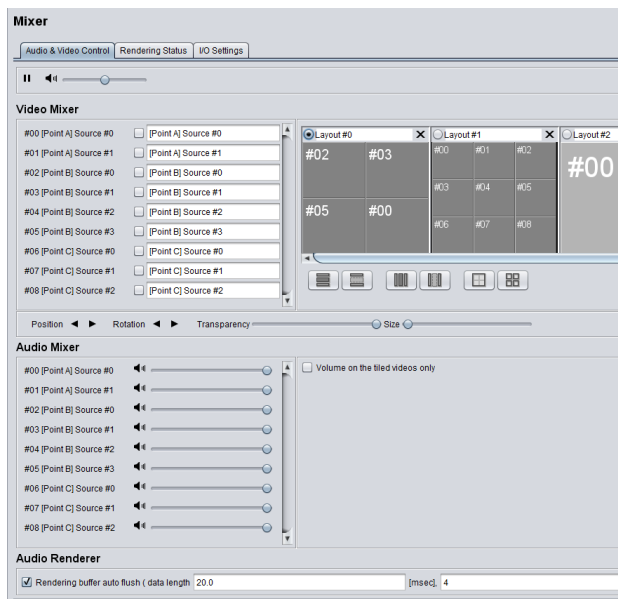


図 10 ミキシング操作 GUI

Fig. 10 GUI components for mixing operation.

および C++ 言語を用いて実装している。ここで、C/C++ 側のメモリ領域で処理しているビデオデータを Java 側の GUI コンポーネント上に表示するためには、通常、大量のメモリコピーが発生する。これを回避するために、プロトタイプシステムでは AWT Native Interface [19] を用いた。AWT Native Interface により Java の GUI コンポーネントに内包されている Windows のネイティブなウィンドウハンドラを C/C++ 側で扱うことが可能となる。プロトタイプシステムでは、C/C++ 側の DirectShow の画面描画用フィルタで処理されたビデオデータを、効率良く Java の GUI コンポーネント上に表示している。

5. 評価実験と考察

図 11 に示す実験環境を用意し、図 12 と図 13 に示す構成でライブ中継用ミキサの評価実験を行った。

5.1 実験環境とミキサの構成

図 12 の単一構成では、ミキサ端末 (M_0) がソースグループを生成し、そのソースグループに参加した送信端末 ($S_0 \sim S_9$) からのストリームを中継サーバ (R) 経由で M_0 が受信し、ミキシング処理を行った。また、ミキシングしたストリームデータをミキサ端末内で動作している Microsoft Expression Encoder 4 へ送り、最終的に利用端末 (U) 内の Windows Media Player で再生表示できることを確認した。この構成では、 $S_0 \sim S_9$ がそれぞれビデオストリームとオーディオストリームを 1 本ずつ送信した場合、 M_0 における CPU 利用率と出力フレームレート、および送信端末とミキサ端末間の遅延を測定した。

一方、図 13 のカスケード構成では M_0 がソースグループを生成し、そのソースグループに参加した前段のミキサ

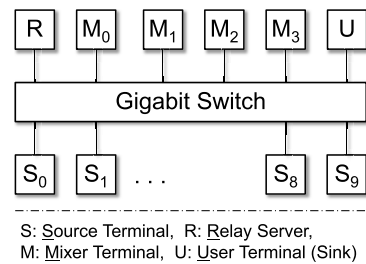


図 11 評価実験環境

Fig. 11 Experimental environments.

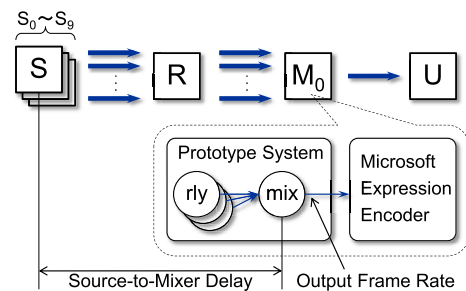


図 12 単一構成

Fig. 12 Single configuration.

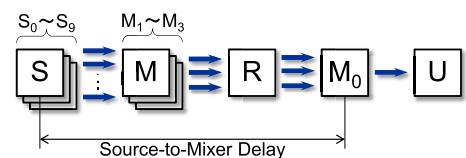


図 13 カスケード構成

Fig. 13 Cascade configuration.

端末 ($M_1 \sim M_3$) からのストリームを R 経由で M_0 が受信し、ミキシング処理を行った。 $M_1 \sim M_3$ は、それぞれ送信端末からのストリームを直接受信してミキシング処理を行った。この構成は、図 4 に示したカスケードミキサの構成における前段のミキサ端末を 3 台としたものである。実際のライブ中継としては、長い列を作り行進するような祭りにおいて、その行進開始地点と終了地点および中間地点を中継元とした場合の 3 地点を想定している。ここで、送信端末数は 10 台であるが、実験では送信端末から複数本のストリームを送出することで、 $M_1 \sim M_3$ への入力ストリーム数を確保した。1 台のミキサ端末で処理可能な入力ストリーム数は図 12 の構成における実験から明らかになるので、図 13 の構成における実験では、カスケードミキサの構成における、送信端末とミキサ端末間の遅延を測定した。

実験で用いた中継端末とミキサ端末のスペックとして、CPU と搭載メモリおよび NIC を表 1 に示す。プロトタイプシステムではミキシング処理を CPU で行っているため、ミキサの性能は CPU の性能に依存する。評価実験を行うにあたっては、専用機器としてのミキサのうち、比較的入手しやすい機器 [20] と同程度の価格で入手可能な PC を用いた。ミキサ端末はすべて同型の PC である。送信端末とし

表 1 中継端末とミキサ端末のスペック

Table 1 Specifications of relay and mixer terminals.

	CPU	Memory	NIC
R	Intel Core i7 870 2.93GHz	8,192MB	Broadcom NetLink Gigabit Ethernet
M ₀ –M ₃	Intel Core i7 3770 3.40GHz	16,384MB	Intel Ethernet Connection I217-LM

では、今回の実験で利用するビデオとオーディオストリームの送信に十分対応でき、ビデオカメラとマイクロフォンを搭載している PC を用いた。また、送信端末とミキサ端末間の遅延を測定するにあたり、ネットワーク網内の遅延は無視できるよう、ギガビットスイッチングハブに各端末を接続して実験を行った。また、LAN を利用しているので、中継サーバを利用しなくてもストリームの中継は可能であるが、実際のライブ中継ではバックエンドの中継サーバが必要である場合が多いことを想定し、中継サーバを含めた構成としている。加えて、プロトタイプシステムのストリーミングプロトコルは TCP と UDP に対応しているが、インターネットの末端からバックエンドの中継サーバを利用することを想定し、実験では TCP を用いた。一方、中継サーバの有無による遅延の比較を行うため、図 12 の単一構成における送信端末とミキサ端末間の遅延の測定実験のみ、中継サーバを利用しなかった場合の実験も行った。

送信端末から送信するビデオストリームのフォーマットは、720 p (ピクセル解像度：1280×720 pixel, フレームレート：30 fps) および 360 p (ピクセル解像度：640×360 pixel, フレームレート：30 fps) を用いた。圧縮には WMV (Windows Media Video) の CBR (Constant Bit Rate) を用い、720 p の場合は 4 Mbps, 360 p の場合は 1 Mbps の設定を利用した。オーディオストリームのフォーマットは PCM (チャンネル数：1, サンプル周波数：44.1 kHz, 量子化ビット数：16 bit, キャプチャ間隔：1/25 sec.) を用いており、キャプチャした PCM データを圧縮せず 705.6 Kbps のオーディオストリームとして利用した。また、M₀ 内で動作させた Microsoft Expression Encoder 4 では、実際の配信を考慮し、720 p のビデオデータについては VC-1 アドバンストのエンコーダを用いて CBR 2 Mbps のストリームを生成し、360 p のビデオデータについては CBR 500 Kbps のストリームを生成した。オーディオデータについては、WMA のエンコーダを用いて CBR 64 Kbps のストリームを生成した。実際に配信をする場合は、通常、生成したビデオとオーディオのストリームを配信ポイントとなる Windows Media サーバか IIS スムーズストリーミングサーバーへ送信するが、実験では M₀ から U 上の Windows Media Player へ直接ストリーミング再生を行った。

また、プロトタイプシステムではビデオとオーディオデータにタイムスタンプが付加され、ストリームごとに同期をとって処理される実装となっている。送信端末ではビ

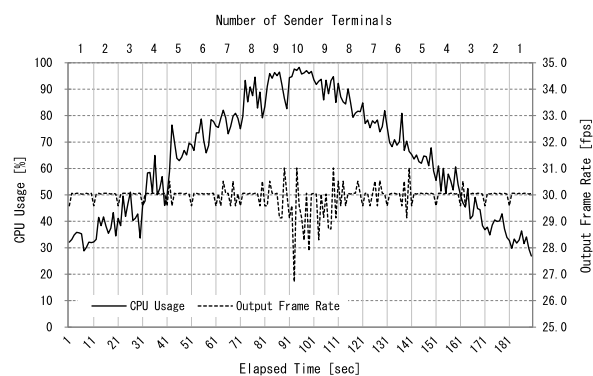


図 14 CPU 利用率と出力フレームレート (720 p)

Fig. 14 CPU usage and output frame rate (720 p).

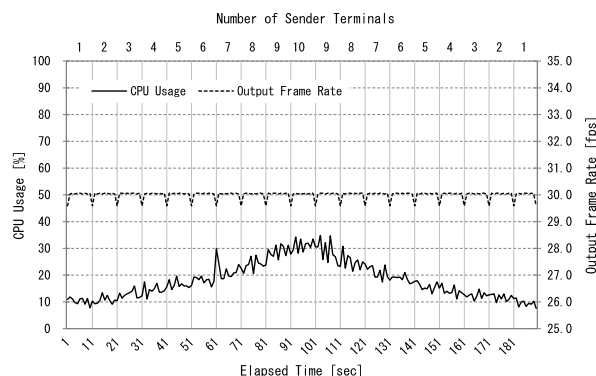


図 15 CPU 利用率と出力フレームレート (360 p)

Fig. 15 CPU usage and output frame rate (360 p).

デオとオーディオデータのキャプチャ時にタイムスタンプを付加しており、ミキサ端末ではビデオとオーディオデータをそれぞれミキシングした後にタイムスタンプを付加している。一方、現在のプロトタイプシステムでは送信端末間の同期は考慮していない。

5.2 プロトタイプシステムにおけるミキサの性能

プロトタイプシステムとして実装したミキサの性能評価として、M₀ における CPU 利用率と出力フレームレート、および送信端末とミキサ端末間の遅延を測定した。

(1) 入力の動的な追加/削除機能

まず、ミキサ端末に対して入力を動的に追加/削除した場合の実験結果をまとめる。実験では、M₀ が生成したソースグループへ送信端末 S₀～S₉ が参加し、各送信端末からのビデオとオーディオストリームを 10 秒ごとに追加した。S₀～S₉ すべての送信端末からのストリームをミキサの入力として 10 秒間処理した後、同じく 10 秒ごとにビデオとオーディオストリームを削除した。この間 1 秒間隔で CPU 利用率とミキサの出力フレームレートを測定し、720 p と 360 p に分けてグラフにしたのが図 14 と図 15 である。グラフ左の縦軸は M₀ の CPU 利用率を、グラフ右の縦軸は出力フレームレートを示している。横軸は経過時間であり、グラフの上部には送信端末数を示している。前述した

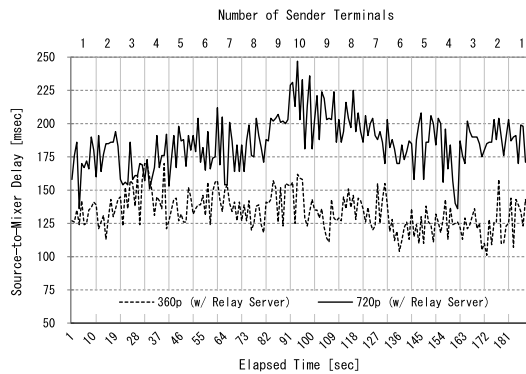


図 16 送信端末・ミキサ端末間遅延 (中継サーバ経由)

Fig. 16 Source-to-mixer delay (with relay server).

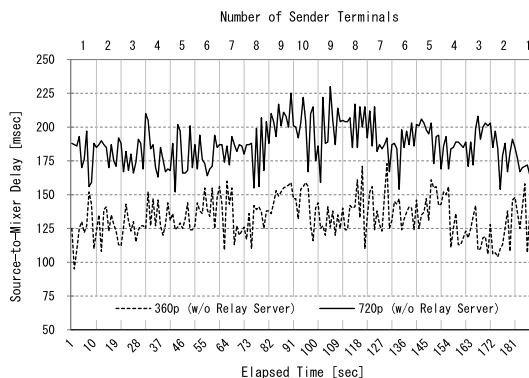


図 17 送信端末・ミキサ端末間遅延 (中継サーバなし)

Fig. 17 Source-to-mixer delay (without relay server).

とおり、この出力フレームレートは図 12 におけるミキサストリームノードから、他プロセスとして動作している Microsoft Expression Encoder 4 へのフレームレートである。また、送信端末とミキサ端末間の遅延は、送信端末においてビデオカメラからキャプチャしたビデオフレームのキャプチャ時刻に対して、そのビデオフレームがミキサ端末上のミキシングビューワで表示されるまでの時間である。図 16 は中継サーバを経由した場合の遅延を、図 17 は中継サーバを利用しなかった場合の遅延を示している。

図 14 と図 15 のグラフからは、入力追加/削除に対する CPU 利用率と出力フレームレートの変化の概要が分かる。CPU 利用率は測定値をそのままグラフにしているのバラつきはあるが、入力数に応じて増減していることが分かる。また、図 14 からは、追加のタイミングで CPU 利用率が大きく変化する傾向も見取れる。これは、資源の確保と追加した入力に対する処理の開始による変化と考えられる。削除のタイミングでもその傾向は見受けられるが、追加のタイミングに比べると変化は少ない。一方、出力フレームレートの測定結果を見てみると、追加/削除のタイミングでフレームレートは一瞬下がる。プロトタイプシステムにおけるミキサの実装では、図 7 における Audio Video Mixer へ Ring Buffer を追加/削除する際、Relay Stream Node と Mixer Stream Node それぞれのスレッド間で処理

の同期をとっている。その同期処理が、出力フレームレートに影響しているものと考えられる。また、図 14 のグラフから 720p の場合は、入力数が 7~10 の区間でフレームレートにバラつきが見られる。後の実験結果として示すが、入力数 7 までは平均的には十分なフレームレートを確保することができる。しかし、入力数が 8~10 の区間は CPU 利用率も高く、プロトタイプシステムにおけるミキサ端末の処理能力がフレームレートに影響していると考えられる。

次に、図 16 と図 17 のグラフからは、入力追加/削除に対する送信端末とミキサ端末間の遅延の変化の概要が分かる。中継サーバの有無によらず 720p の場合は、入力数が増えると遅延も増加傾向にあることが見て取れる。一方、360p の場合、入力数の増加による遅延の増加は顕著には現れていない。ここで、遅延の測定については中継サーバを経由する場合としない場合の実験を行ったので、その結果を比較する。入力を追加/削除した全 19 区間ごとの平均遅延の差を求めると、360p を用いた実験では、11 区間で中継サーバなしの方が遅延は少なかった。720p を用いた実験では、8 区間で中継サーバなしの方が遅延は少なかった。逆に、中継サーバ経由の方が平均遅延は少ない区間がある。中継サーバの有無による、区間ごとの平均遅延の差は ± 20 msec の範囲であった。遅延の測定には、前述したとおりビデオカメラからキャプチャしたビデオフレームのキャプチャ時刻と、そのビデオフレームがミキサ端末上のミキシングビューワで表示される時刻との差を求めている。したがって 30fps のビデオフレームを 1 枚表示している時間、約 33 msec 分の誤差が含まれる。よって、この実験における遅延測定方法では、中継サーバの有無による遅延の差は測定誤差の範囲に収まる程度と考えられる。一方、全サンプルデータの標準偏差を算出したところ、360p の場合はほとんど変化がない (中継サーバ経由: 14.1 msec, 中継サーバなし: 14.8 msec) が、720p の場合は中継サーバ経由の方が若干大きい (中継サーバ経由: 19.0 msec, 中継サーバなし: 15.7 msec) 結果となった。720p の場合は、中継サーバを経由することで若干遅延のバラつきが増えたとも考えられるが、結果としては、中継サーバの有無による遅延には顕著な差は見られない。

以上、入力の動的な追加/削除機能の動作に合わせて、CPU 利用率と出力フレームレート、および送信端末とミキサ端末間の遅延に関する実験結果をまとめた。動的な追加/削除機能を試すため、比較的短い間隔でストリームの追加と削除を行ったので、そのタイミングにおける各測定値の変化は確認できた。一方、ミキサの入力数に対する平均的な性能を確認するためには、入力数ごとの測定時間を増やし、各測定値のサンプル数を増やす必要がある。引き続き、ミキサの入力数に対する評価実験結果をまとめる。

(2) 入力数に対するミキサの性能評価

先の実験同様、 M_0 が生成したソースグループへ送信端

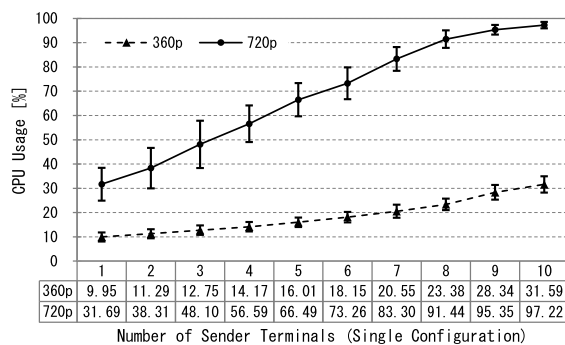


図 18 CPU 利用率 (単一構成)

Fig. 18 CPU usage (single configuration).

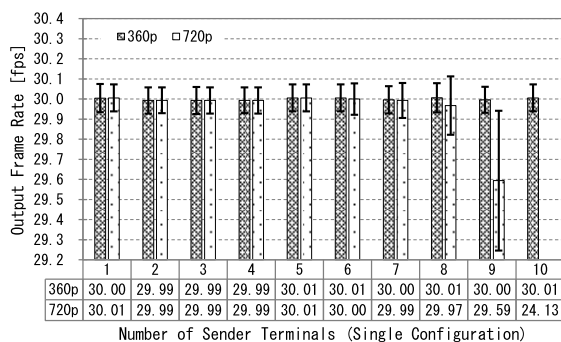


図 19 出力フレームレート (単一構成)

Fig. 19 Output frame rate (single configuration).

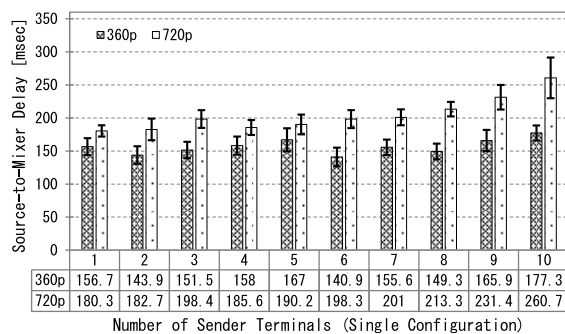


図 20 送信端末・ミキサ端末間遅延 (単一構成)

Fig. 20 Source-to-mixer delay (single configuration).

末 $S_0 \sim S_9$ が参加し、各送信端末はビデオとオーディオストリームを送信した。各送信端末からのストリームを M_0 が受信し、ミキシング処理を開始した後 3 分間、 M_0 の CPU 利用率とフレームレート、および送信端末とミキサ端末間の遅延を測定した。CPU 利用率の平均値をグラフにしたのが図 18 である。図 18 の縦軸およびグラフの下表は M_0 の CPU 利用率を示しており、横軸は送信端末数を示している。また、グラフには送信端末数ごとの標準偏差も合わせて示してある。同様に図 19 は、出力フレームレートの平均値と標準偏差を、図 20 は送信端末とミキサ端末間の遅延の平均値と標準偏差を示している。

720p のミキシング処理を行った場合、図 18 より、送信端末数が 1 台から 8 台目までは CPU 利用率がほぼ単調増加している。これはデコード処理の増加によるもので、

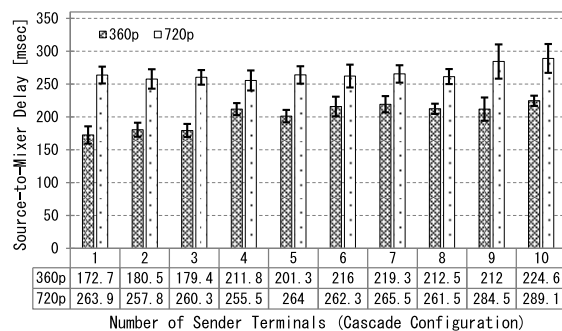


図 21 送信端末・ミキサ端末間遅延 (カスケード構成)

Fig. 21 Source-to-mixer delay (cascade configuration).

プロトタイプシステムでは、送信端末 1 台につき約 7%～10% の CPU 資源を必要とする。一方、8 台目から 9 台目に送信端末が増加した際には CPU 利用率の平均で 3.91% 増加しているが、これは、9 台目の送信端末から受信したストリームのデコード処理を十分に行えていないことを示している。また、送信端末数 7 台目以降、CPU 利用率の標準偏差の幅が小さくなることも図 18 から確認できる。これは、 M_0 における処理に余裕がなくなっていることを示しており、CPU 利用率がほぼ単調増加していた 8 台目でもミキシング処理の結果である出力フレームレートに影響していることが図 19 から見て取れる。図 19 における 720p の場合の平均出力フレームレートは、送信端末数 7 台目までは、ほぼ 30 fps であるが、送信端末 8 台目は 29.97 fps となる。標準偏差の幅も送信端末 8 台目から明らかに増え、一定のフレームレートによる処理ができない状態となっている。図 20 における 720p の場合の送信端末・ミキサ端末間遅延を見ても、送信端末 7 台目までは多少のばらつきはあるものの、ほぼ一定であるが、送信端末 8 台目以降は増加していることが分かる。以上より、この実験で用いたコーデックの設定において、プロトタイプシステムのミキサ端末では、720p の場合、送信端末 7 台までミキシング処理できたことが分かる。一方、360p のミキシング処理を行った場合、送信端末 10 台までのミキシング処理は十分可能であることが図 18～図 20 より確認できる。

引き続き、カスケード構成の場合の、送信端末とミキサ端末間における遅延の測定結果を図 21 に示す。図 13 のカスケード構成において、後段のミキサ端末 (M_0) への入力ストリームは前段のミキサ端末 ($M_1 \sim M_3$) 3 台分となる。これは図 18～図 20 における送信端末 3 台の場合と同じである。一方、カスケード構成により送信端末と M_0 間の遅延は単一構成の場合より増加する。どの程度の遅延が発生するかを確認するため、前段のミキサの 1 つである M_1 に対して送信端末の数を 1 台～10 台まで増加させ、遅延の測定を行った。ここで、図 7 におけるミキサの機能モジュールのうち、カスケード構成の前段のミキサでは、Audio Video Mixer の出力を Encoder モジュールでエ

ンコードした後、パケット化し中継サーバ経由で後段のミキサ宛に送信する。単一構成の場合同様、送信端末から受信したストリームのデコード処理とミキシング操作後のエンコード処理は CPU で行っている。前段のミキサから後段のミキサへ送信されるストリームのフォーマットは、送信端末の送信ストリームフォーマットと同じ設定としている。前段のミキサ端末における処理と後段のミキサ端末における処理はほぼ同じであり、その結果として、前段のミキサ M_1 における CPU 負荷が、送信端末と後段のミキサ M_0 間の遅延にも反映される。720p の場合、図 21 のグラフから送信端末 8 台目まではほぼ変化しないが、送信端末 9 台目と 10 台目では、8 台目までより遅延が増加していることが分かる。360p の場合は、送信端末 3 台目までに比べて 4 台目以降で遅延が若干増加するが、実験した送信端末 10 台までの範囲では、顕著な遅延の増加は発生していない。

以上、プロトタイプシステムのミキサの性能評価として、 M_0 における CPU 利用率と出力フレームレート、および送信端末とミキサ端末間の遅延の測定結果を示した。測定結果から、実験で用いた 720p のフォーマットであれば、最大で送信端末 7 台分のストリームを 1 台のミキサ端末で処理できることが確認できた。360p のフォーマットについては、少なくとも送信端末 10 台分のストリームまで処理できることも確認した。

また、送信端末とミキサ端末間の遅延については、単一構成の場合とカスケード構成の場合の測定結果を示した。1 章で述べたとおり、中継元におけるビデオ撮影者とミキサ操作者との間でタイミングをとれる程度のエンド間遅延内でストリーミングとミキシング処理が行われる必要がある。ここで、過去に実施された遠隔会議における伝播遅延の影響の実験 [21] を参照すると、往復遅延 250 ms ではほとんどの会議場面が許容されるが、500 ms では一部適さない会議場面（速くてひん度の高いやりとりが要求される会議場面）が生ずる、とする一方で、討論、意見交換が主体である一般の会議では 500 ms でもほとんど支障がない、と報告されている。この文献の実験結果を参照し、中継元におけるビデオ撮影者とミキサ操作者間のやりとりが、速くてひん度の高いやりとりではない場合を想定し、ここでは、往復遅延 500 ms を許容値の目安と考える。

プロトタイプシステムを用いた実験では、送信端末からミキサ端末までの片方向の遅延を測定しているため、単純に 500 ms の半分の 250 ms を許容値とすると、図 20 から、ミキサで処理可能なストリーム数の範囲ではこの値以下の遅延であることが分かる。しかし、カスケード構成の場合、720p のストリームではすべて 250 ms を超えている。カスケード構成の場合でも、中継元におけるビデオ撮影者と前段のミキサ操作者間のやりとりには支障がないと考えられるが、中継元におけるビデオ撮影者と後段のミキサ操作

者間でタイミングをとるためには、より遅延を少なくする工夫が必要となる。プロトタイプシステムでは時間的な揺らぎを吸収するため、リングバッファにてバッファリング（ビデオは 2/30 sec., オーディオは 2/25 sec.）を行っているため、カスケード構成の場合はミキサストリームノードにおいて少なくとも 4/25 sec. (= 160 ms) 分の遅延が発生する。片方向の遅延を 250 ms 内に抑えるためには、ビデオ 1 フレーム分（対応するオーディオデータを含む）のエンド間におけるデータ処理を平均 90 ms 以内で行う必要があることになる。実験環境はネットワーク網の遅延をほぼ無視できる環境であることも考慮すると、遅延は可能な限り抑えられることが望ましい。4 章でも述べたとおり、プロトタイプシステムではミキサの入力となる中継ストリームノードとミキサストリームノードにおけるミキシング処理を異なるスレッドに分けることにより、処理の並列性を高めているが、より遅延を抑えるためには今後検討の余地がある。

5.3 機能的な考察

評価実験の結果をふまえ、ライブ配信ソフトウェア [4], [5], [6], および専用機器として比較的入手しやすい AV ミキサ [20] とその上位機種 [22] の機能を参照し、提案システムの機能的な考察を以下に述べる。

(1) 入力の動的な追加/削除機能と入力数

提案したライブ中継用ミキサにおいて実現した、ミキサへの入力となるビデオソースの動的な追加/削除機能は、既存のライブ配信ソフトウェアや専用機器では実現されておらず、提案システムにおける機能の新規性を示すものである。入力として扱えるビデオソースの数は、ライブ配信ソフトウェアでは PC の構成や性能および利用するビデオとオーディオのフォーマットに依存する一方、参照した専用機器では 4 系統のビデオ入力を扱え、その他の専用機器における主流は 4 系統または 8 系統である。プロトタイプシステムの評価結果から、提案システムでは 720p の場合で 7 入力、360p では少なくとも 10 入力のミキシング処理が可能であることを示した。プロトタイプシステムは、ビデオソースの動的な追加/削除に対応し、既存の専用機器と同等の入力数を処理することができる。また、ソフトウェアとしてライブ中継用ミキサを実現しているため、必要に応じて高性能な PC を使用することにより、操作や設定の大きな変更なしに処理可能な入力数を増やすことも可能である。これは、入力数が固定である専用機器に対し、ソフトウェアとして動作するライブ中継用ミキサの利点であると考えている。

(2) カスケードミキサの動的構成機能

専用機器をカスケード接続し、カスケードミキサとして利用することは以前より可能であるが、物理的にケーブルを用いた接続が必要であり、比較的広い範囲を想定すると

カスケード接続は困難である。専用機器のミキサ機能とライブ配信ソフトウェアのミキサ機能を組み合わせて、あらかじめ物理的または論理的にカスケードミキサを構成することも可能である。しかし、ネットワーク経由でミキサを動的に追加/削除できる機能は、ライブ配信ソフトウェアや専用機器では実現されていない。本論文で対象としている複数の会場で同時に実施されるスポーツイベントや、神輿/山車/踊り手等が長い列を作る祭りのライブ中継を容易に実現することを考慮した場合、複数の送信端末とミキサ端末を1つの送信グループとして扱い、それを動的に追加/削除できる機能は実用性があると考えている。

(3) 既存ストリーミングサービスを利用した配信機能

ライブ配信ソフトウェアは、既存のストリーミングサービスを利用した配信機能を搭載している。専用機器を用いる場合は、通常、専用機器をPCへ接続し、ライブ配信ソフトウェアを用いて既存のストリーミングサービスを利用する。通常、既存のストリーミングサービスを利用するためには、そのサービスに適合しているライブ配信ソフトウェアを用いる。提案したライブ中継用ミキサは、ミキシング処理後のデータをプロセス間通信により既存のライブ配信ソフトウェアへ渡す機能を実現した。プロトタイプシステムではMicrosoft Expression Encoder 4の入力として利用可能なDirectShowのソースフィルタを実装し、接続できることを確認した。同様の仕組みにより、ビデオカメラやマイクロフォン用のデバイスドライバとしてソフトウェアモジュールを実装することで、Microsoft Expression Encoder 4以外の既存システムへもビデオとオーディオデータを渡すことは可能であると考えられる。実装したプロトタイプシステムにより、その実現可能性を確認できた。様々な既存のライブ配信ソフトウェアから見て、本システムの出力がビデオカメラとマイクロフォンとして扱える機能の実装を今後も継続して行う。

以上、機能的な考察を述べた。LAN上での実験結果から、要件1と要件2に対する機能は実現できたと考えている。一方、要件3の遅延に関しては、送信端末やミキサ端末として利用するPCの性能にも依存するものの、より遅延を抑えるための実装やシステムの構成方法についての検討を続ける。

(4) 機能的な検討課題

本研究開発では、1章で示した要件を満たすライブ中継用ミキサを、時間や場所を問わずに利用できる仕組みの実現を目指しており、その一環として汎用PC上で動作するライブ中継用ミキサを開発した。しかし現状では、ミキサ端末としての汎用PCを準備して中継地点に設置するという手間が残っており、多くのスマートデバイスを送信端末として扱うことに対しても検討すべき課題が残っている。

まず、時間や場所を問わずミキサの機能を使ったライブ中継を容易に実施するためには、ミキサ端末の設置を不要

とすることが重要であると考えている。そこで、ミキサ端末におけるミキシング操作と処理を切り離し、ネットワークのバックエンドでミキシング処理を実行し、バックエンドのミキサを遠隔から操作できる機能を今後検討する。これにより、現在のミキサ端末をスマートデバイスに置き換え、スマートデバイスからもオンデマンドでミキサの機能が利用できる仕組みの実現を目指す。

また、現在のプロトタイプでは送信端末間の時刻同期を考慮していないが、これについても検討する必要があると考えている。本論文におけるLAN上の評価実験では、送信端末間の同期ズレは体感できなかったが、無線LANや携帯電話網を利用して多くのスマートデバイスが送信端末となりうることを想定すると、遅延の増加に加えて送信端末間の同期ズレが生じる。送信端末が送信するビデオストリーム間の時間的な関係がどの程度厳密に処理されるべきかは中継されるコンテンツによって異なり、厳密性が高いほどバッファリングによる遅延が増加する。同期の厳密性とバッファリングの遅延はトレードオフの関係にあり、中継するコンテンツに応じて適応可能な同期の仕組みは重要な検討課題と考えている。

6. まとめ

本論文では、複数のビデオソースを扱うライブ中継を支援するために必要となる機能を整理し、複数ビデオソースの動的な構成を可能とするライブ中継用ミキサを提案した。ライブ中継用ミキサの要件としては、(1) ミキサへの入力となるビデオソースの数の動的な増減に対応できること、(2) ミキサのカスケード構成が可能であること、(3) ビデオソースからミキサまでの遅延がタイミングをとれる程度に小さいことがあげられる。これらの要件を満たすためには、送信側の端末が任意の時間に送信するストリームを自動的にミキサ端末の入力とする機能と、その機能をミキサ間でも利用できる仕組みが必要となる。そこで、ミキサ端末ではソースグループを生成し、送信端末はそのソースグループに参加したうえでストリーム送信を行うためのプロトコルを設計・実装した。ソースグループに参加している送信端末からのストリームは、ミキサ端末へ自動的に中継されプレビュー表示される。ミキサ操作者は必要に応じて自動的に中継されるビデオソースに対してミキシング操作をすることが可能となる。また、既存のライブ配信ソフトウェアを利用してミキシング処理後のストリームを配信するための仕組みも実装した。そしてプロトタイプシステムを用いた評価実験では、単一構成とカスケード構成のミキサに対してビデオソースの動的な追加機能を確認するとともに、既存の専用機器と同程度の入力数を処理できることを確認した。送信端末とミキサ端末間の遅延については、実装上の検討の余地も残るが、機能的な要件を満たすライブ中継用ミキサをPC上で実現できることを示した。

今後は、プロトタイプシステムの実装における性能と機能の向上を図るとともに、実際のライブ中継イベントでの利用を通し、より効果的にライブ中継を支援できる実用性のあるライブ中継用ミキサを実現する。また、考察として述べた検討課題に対する機能の設計と実装を含め、引き続き、多様なライブ中継を支援できる高機能なライブ中継用システムに関する研究開発を進める。

謝辞 本研究の一部は、科学研究費補助金（課題番号：15K00427）、および、独立行政法人情報通信研究機構のプロジェクト（プロジェクト番号：JGNX-A11040）の一環として実施されました。ここに記して謝意を表します。

参考文献

- [1] Ustream – The leading HD streaming video platform, available from <http://www.ustream.tv> (accessed 2015-10-13).
- [2] Live – YouTube, available from <https://www.youtube.com/live> available from (2015-10-13).
- [3] ニコニコ生放送, 入手先 (<http://live.nicovideo.jp>) (参照 2015-10-13).
- [4] media encoder, video capture software, audio capture software | Adobe Flash Media Live Encoder, available from <http://www.adobe.com/products/flash-media-encoder.html> (accessed 2015-10-13).
- [5] Live Webcasting Software | Telestream Wirecast | Overview, available from <http://www.telestream.net/wirecast> (accessed 2015-10-13).
- [6] Niconico Live Encoder – ニコニコ生放送, 入手先 (<http://live.nicovideo.jp/s/encoder>) (参照 2015-10-13).
- [7] Chan, B.G. and Yuen, P.: MixNStream: multi-source video distribution with stream mixers, *Proc. ACM Workshop on Advanced Video Streaming Techniques for Peer-to-Peer Networks and Social Networking*, pp.77–82 (2010).
- [8] Cheung, E. and Karam, G.: A novel implementation of very large teleconferences, *Proc. ACM IPTComm '10 Principles, Systems and Applications of IP Telecommunications*, pp.84–90 (2010).
- [9] Schulzrinne, H., Casner, S., Frederick, R. and Jacobson, V.: RTP: A Transport Protocol for Real-Time Applications, RFC3550 (2003).
- [10] Ooi, W.T. and Renesse, V.R.: Distributing Media Transformation Over Multiple Media Gateways, *Proc. 9th ACM International Multimedia Conference* (2001).
- [11] 橋本浩二, 柴田義孝: 利用者環境を考慮した相互通信のためのミドルウェア, 情報処理学会論文誌, Vol.46, No.2, pp.403–417 (2005).
- [12] Zhao, H., Smilkov, D., Dettori, P., Nogima, J., Schaffa, F.A., Westerink, P. and Wah, C.: A Feasibility Study of Collaborative Stream Routing in Peer-to-Peer Multiparty Video Conferencing, *Proc. IEEE International Symposium on Multimedia*, pp.233–240 (2011).
- [13] Wang, Z., Zhao, J., Xi, W. and Jiang, Z.: A Scalable P2P Video Conferencing System Based on VCStream Model, *Proc. IEEE/ACIS 11th International Conference on Computer and Information Science*, pp.77–82 (2012).
- [14] Guo, H., Lo, K.-T., Qian, Y. and Li, J.: Peer-to-Peer Live Video Distribution under Heterogeneous Bandwidth Constraints, *IEEE Trans. Parallel and Distributed Systems*, Vol.20, No.2, pp.233–245 (2009).
- [15] OpenLDAP Foundation: OpenLDAP, Main Page, available from <http://www.openldap.org> (accessed 2015-10-13).
- [16] Hashimoto, K. and Shibata, Y.: MidField: An Adaptive Middleware System for Multipoint Digital Video Communication, *Digital Video*, De Rango, F. (Ed.), ISBN: 978-953-7619-70-1, InTech, DOI: 10.5772/8032 (2010). available from <http://www.intechopen.com/books/digital-video/midfield-an-adaptive-middleware-system-for-multipoint-digital-video-communication> (accessed 2015-10-13).
- [17] MSDN Library: DirectShow, available from [https://msdn.microsoft.com/en-us/library/windows/desktop/dd375454\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/windows/desktop/dd375454(v=vs.85).aspx) (accessed 2015-10-13).
- [18] Download Microsoft Expression Encoder 4 with Service Pack 2 (SP2) from Official Microsoft Download Center, available from <http://www.microsoft.com/en-us/download/details.aspx?id=27870> (accessed 2015-10-13).
- [19] AWT Native Interface, available from http://docs.oracle.com/javase/7/docs/technotes/guides/awt/AWT_Native_Interface.html (accessed 2015-10-13).
- [20] Roland VR-3EX | AV Mixer, available from <http://www.roland.co.jp/products/vr-3ex> (accessed 2015-10-13).
- [21] 鑑沢 勇, 滝川 啓, 大久保栄, 渡辺義郎: 衛星通信を利用した画像会議におけるエコー及び伝搬遅延の影響, 電子情報通信学会論文誌 B, Vol.J64-B, No.11, pp.1281–1288 (1981).
- [22] Roland VR-50HD | Multi-Format AV Mixer, available from <http://www.roland.co.jp/products/vr-50hd> (accessed 2015-10-13).



橋本 浩二 (正会員)

1996 年東洋大学大学院工学研究科電気工学科専攻博士前期課程修了。同年 (株) CSK 総合研究所入社。1998 年岩手県立大学ソフトウェア情報学部助手。2001 年東北大学大学院情報科学研究科博士後期課程修了。2002 年岩手県立大学ソフトウェア情報学部講師。2005 年同大学助教授 (2007 年より准教授), 現在に至る。博士 (情報科学)。映像通信および分散システムの研究に従事。電子情報通信学会, IEEE 各会員。



柴田 義孝 (正会員)

1985 年 UCLA 大学院コンピュータサイエンス学専攻博士課程修了. Ph.D. in Computer Science. 1985 年から 1988 年まで Bellcore (旧 AT & T ベル通信研究所) にて専任研究員としてマルチメディア情報ネットワークの研

究に従事. 1989 年より東洋大学工学部情報工学科助教授. 1997 年同大学教授. 1998 年より岩手県立大学ソフトウェア情報学部教授, 同大学メディアセンター長. 2005 年より同大学大学院ソフトウェア情報学専攻主任, 2012 年より同大学理事副学長, 現在に至る. 電子情報通信学会, 感性工学会, IEEE, ACM 各会員.