ライブ放送のための映像処理システムにおける負荷分散方式の設計と実装

川上　朋也[†,††]　　義久　智樹[††]　寺西　裕一[†††,††]

† 奈良先端科学技術大学院大学情報科学研究科　〒 630–0192 奈良県生駒市高山町 8916–5
†† 大阪大学サイバーメディアセンター　〒 567–0047 大阪府茨木市美穂ヶ丘 5–1
††† 国立研究開発法人情報通信研究開発機構　〒 184–8795 東京都小金井市貫井北町 4–2–1
E-mail: †kawakami@is.naist.jp

あらまし　著者らの研究グループでは，ライブ放送のための分散型映像処理システムを研究開発してきた．本システムでは，映像処理サーバが映像撮影端末から依頼された映像処理を行う．多数の映像撮影端末が映像処理サーバに映像処理を依頼すると通信負荷や処理負荷が大きくなって映像処理に時間がかかり，ライブ放送を円滑に行えない．そこで本研究では，映像処理を依頼する映像処理サーバを選択して映像処理サーバにかかる負荷を分散させる方式を提案，実装する．提案方式では，複数の映像処理サーバがある場合に，映像処理内容や負荷に応じて処理を依頼する映像処理サーバを，P2P 型モバイルエージェントシステムを用いて選択する．複数の映像処理サーバと映像撮影端末を用いて実験し，映像処理を完了するまでの時間への影響を確認した．

キーワード　ストリーミング配信，インターネット放送，ビデオオンデマンド，リアルタイム

1. はじめに

近年の映像配信技術の発達に伴い，USTREAM やツイキャスといった，個人がインターネットを介してリアルタイムな映像配信を行えるインターネットライブ放送サービスが普及している [1]．例えば，USTREAM では，個人のパソコンで USTREAM のホームページにブラウザでアクセスし，放送を開始するボタンをクリックすることで，パソコンに接続されたカメラで撮影された映像を配信できる．ツイキャスでは，主にスマートフォンからの配信を対象としており，専用アプリを用いてスマートフォンのカメラで撮影された映像を配信できる．視聴者は，インターネットブラウザや専用アプリの画面に列挙された配信中のインターネットライブ放送の中から興味のあるライブ放送を選択し，配信されている映像をブラウザや専用アプリで視聴する．

インターネットライブ放送サービスでは，放送者が重畳表示や画像処理といった映像効果を付加する場合がある．映像処理を短時間で行うことで，より多くの付加情報や重畳表示を行ってさらにライブ放送を盛り上げられたり，鮮明な画像処理を行ってライブ放送しやすくなったりする．しかし，現状では，個人が所有するパソコンやスマートフォンの計算能力に応じて，短時間で処理できる簡易な映像効果しか付加できなかった．例えば，物体検出に時間がかかるため，あらかじめ付加情報を表示させておいて，付加情報が表示されている位置に人や物が写るようにすることがあった．複雑な映像効果を付加すると，放送者の所望のタイミングに映像効果を付加できなかったり，映像配信のビットレートが下がるといった問題が発生する．計算能力の高い計算機を用いることで映像処理にかかる時間を短縮できるが，高価で個人で導入しにくい．

著者らの研究グループでは，ライブ放送のための分散型映像処理システムを研究開発してきた [2]．本システムでは，サービス提供者が映像処理サーバを用意し，映像撮影端末（PC や携帯情報端末）から依頼された映像処理を行う．また，映像撮影端末は，映像処理に必要なプログラムライブラリである映像効果ライブラリを映像処理サーバへ送信する．ライブ放送時には，映像撮影端末で撮影した映像を映像処理サーバに送信する．複雑な映像処理を計算能力の高い計算機で行うことで，映像処理にかかる時間を短縮しつつ，放送者は映像効果を付加できる．

映像撮影端末は，1 台の映像処理サーバを指定していたため，複数の映像撮影端末が同じ映像処理サーバを指定している場合，映像処理に関わる負荷が大きくなって映像処理に時間がかかる問題がある．映像処理サーバが複数ある場合には，負荷が小さい映像処理サーバを利用することで，短時間で映像処理を行える．これまでに処理負荷を分散させる多数の方式が提案されているが，あらかじめ負荷分散システムを構築するものがほとんどであった．インターネットライブ放送では，ライブ放送できるスマートフォンのような映像撮影端末が多数あり，あらかじめ負荷分散システムを構築しておくことは困難である．ライブ放送を始めようとしてから既存方式を用いた負荷分散システムを構築すると，ライブ放送開始までの時間が長くなる．このため，ライブ放送では，映像撮影端末を含めた負荷分散システムをあらかじめ構築することなく（ゼロコンフィグ），負荷を分散できる映像処理システムを用いることでライブ放送開始をすぐに行える．そこで本研究では，映像処理を依頼する映像処理サーバを選択して映像処理サーバにかかる負荷を分散させる方式を提案，実装する．提案方式では，あらかじめ映像撮影端末を含めずに負荷分散システムを構築できるように P2P 型の負荷分散システムを構築する．P2P 型の負荷分散システムでは，

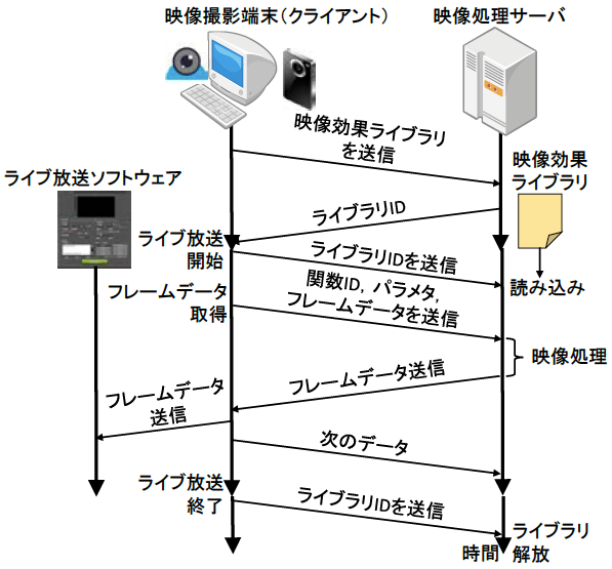

図 1　システム構成

負荷分散システムに接続するとシステムが負荷が小さい映像処理サーバを選択するため，映像撮影端末は負荷分散を意識することなくライブ放送を開始できる．提案方式では，複数の映像処理サーバがある場合に，映像処理内容や負荷に応じて処理を依頼する映像処理サーバを，P2P 型モバイルエージェントシステムを用いて選択する．

以下，2 章で想定する分散型映像処理システムについて述べる．3 章で負荷分散方式の提案と実装について述べ，4 章で実機を用いた実験について述べる．5 章で関連研究について説明し，最後に 6 章で本稿をまとめる．

2.　ライブ放送のための分散型映像処理システム

著者らの研究グループでは，ライブ放送のための分散型映像処理システムを研究開発してきた [2]．本章では，想定する分散型映像処理システムについて説明する．

2.1　システム構成

分散型映像処理システムのシステム構成を図 1 に示す．映像撮影端末は，インターネットライブ放送の放送者が映像を撮影する計算機である．映像撮影端末は，接続されたカメラで映像を撮影でき，映像のデータをインターネットを介して他の計算機に送信できる．映像処理サーバは，インターネットライブ放送サービス提供者が設置することを想定した高い計算能力をもつ計算機であり，インターネットを介して映像データおよび映像効果ライブラリの送受信を行える．映像効果ライブラリは，放送者が用意した任意の映像処理を行えるプログラムライブラリであり，映像処理サーバのプログラムで読み込んで実装された関数を呼び出すことで実行できる．詳細は次節で説明する．例えば，放送者の所望の物体検出や重畳表示，画像処理を行う映像効果を実装した映像効果ライブラリが考えられる．ライブ放送ソフトウェアは，インターネットライブ放送サービス提供者が公開するライブ放送のためのソフトウェアであり，映像撮影端末から実行できる．インターネットブラウザを介して実行するライブ放送ソフトウェアもある．

近年のクラウドコンピューティング環境の普及に伴い，インターネットライブ放送サービス提供者が高性能な計算機を設置し，放送者が利用する上記のようなシステム構成は現実的な構成といえる．

2.2　要 求 機 能

本研究では，以下をインターネットライブ放送のための分散型映像処理システムへの機能要求と考える．

- 放送者の任意のタイミングに映像効果を付加できる

1 章で述べた例のように，ライブ放送中に一貫して付加される映像効果とは別に，付加情報を表示したり，放送を盛り上げたりするために，放送者の任意のタイミングで映像効果することがある．例えば，放送者の知人が写っている間，名前を表示させたり，放送者が楽しいと思ったときに音符マークを表示したりすることが考えられる．これらを実現するためには，放送者の任意のタイミングに映像効果を付加できる必要がある．

- 放送者が所望の映像効果を付加できる

ロゴを表示したり時計を表示するといった簡易な映像効果ではなく，放送者の所望の任意の映像効果を付加することで，よりライブ放送を盛り上げたり，ライブ放送しやすくなる．例えば，放送者独自の画像アイコンを表示させてライブ放送を盛り上げたり，放送者の所望の明るさに映像を調整したりすることが考えられる．これらを実現するためには，放送者が所望の映像効果を付加できる必要がある．

2.3　映像効果ライブラリ

本研究で提案する分散型映像処理システムでは，放送者が所望の映像効果を付加できるように，放送者が映像効果を実装した映像効果ライブラリを用意する．映像効果ライブラリはプログラムライブラリであり，様々な映像効果を実装できる．ライブ放送中に映像効果ライブラリを映像処理サーバに送信すると，通信帯域が低下するため，ライブ放送前に映像処理サーバに送信する．複数の放送者が送信した映像効果ライブラリを識別するため，ライブラリ ID を設定する．放送者は，映像効果を付加したいタイミングに，ライブラリ ID で指す映像効果ライブラリに実装されている複数の関数の中から，付加したい映像効果を示す関数 ID を選択する．関数 ID が示す関数の引数として渡すパラメタも渡せる．

映像処理サーバは，放送者からあらかじめ映像効果ライブラリを受信して保存しておく．映像効果ライブラリの読み込み時間を削減するため，映像処理サーバは，ライブ放送が開始されると，プログラムで放送者が送信した映像効果ライブラリを読み込む．映像撮影端末からフレームデータを受信時にライブラリ ID が指定されていると，映像効果ライブラリに実装されている関数の中から対応する関数呼び出し，放送者の所望の映像効果を付加する．ライブ放送が終了すると，読み込んでいた映像効果ライブラリを開放する．

2.4　映像撮影端末

インターネットライブ放送では，通信量を削減するために映像の幾つかのフレームのデータをまとめて他の計算機に送信することがあるが，送信するフレームの取得を待つため，遅延

時間が長くなる．遅延時間が長くなると，放送者の任意のタイミングに映像効果を付加しにくくなる．そこで，提案する分散型映像処理システムでは，1 フレーム毎に映像処理サーバにフレームデータを送信する．映像効果を付加する場合には，同時にライブラリ ID とパラメタも送信する．

フレームデータの送信後、映像処理サーバで映像処理されたフレームデータを受信すると，ライブ放送ソフトウェアに送信し，インターネットで映像配信する．続けて次のフレームデータを映像処理サーバに送信する．ライブ放送が終了すると，映像処理サーバに通知する．

2.5　映像処理サーバ

インターネットライブ放送サービスのほとんどでは，放送者を識別するために，ライブ放送開始前にサービスにログインすることが多い．映像処理サーバが直接インターネットで映像配信すると，これらのログイン情報を映像処理サーバが保持する必要があり，セキュリティ等の問題が発生する可能性がある．そこで，提案する分散型映像処理サーバでは，映像処理サーバは映像処理を施したフレームデータを映像撮影端末に返信し，映像撮影端末がライブ放送ソフトウェアを用いて映像配信する．ログイン等の配信上の制約がなければ映像処理サーバから直接ライブ放送を行うことも考えられる．

映像処理サーバは，映像撮影端末からフレームデータを受信すると，関数 ID が指定されている場合には映像効果を施して映像撮影端末に返信する．ライブ放送が終了すると，読み込んでいた映像効果ライブラリを解放する．

3.　負荷分散方式の提案および実装

3.1　概　　要

多数の映像撮影端末が映像処理サーバに映像処理を依頼すると通信負荷や処理負荷が大きくなって映像処理に時間がかかり，ライブ放送を円滑に行えない．そこで本研究では，映像処理を依頼する映像処理サーバを選択して映像処理サーバにかかる負荷を分散させる方式を提案し，負荷分散機構として実装する．

負荷分散機構の実装では，既存のシステムの変更を最小限にとどめることを基本方針とする．これは，既に映像撮影クライアントと映像処理サーバの各システムが存在することから，それらのシステムの大幅な改修は容易ではないためである．このため，原則として既存のシステムとは別プロセスで稼動させ，それぞれが独立して動作する疎結合な設計とする．最終的に負荷分散機構への適合のために既存システムに要した変更点は，映像処理サーバに負荷の通知機能を追加するのみとなった．

図 2 に本研究のシステムが想定している配信環境を示す．想定環境では映像撮影端末，映像処理サーバ，映像受信端末の 3 種の端末・サーバがそれぞれ複数台存在する．映像撮影端末は，必要な映像処理を行える映像処理サーバを選択して，映像効果ライブラリや撮影した映像を送信する．映像処理サーバでは，映像撮影端末から送られた映像を映像撮影端末からの指示に従い画像加工やエフェクト処理を施す．映像受信端末は，各々が受信したい映像を扱っている映像処理サーバに接続し，処理済みの映像を随時受け取ることとなる．

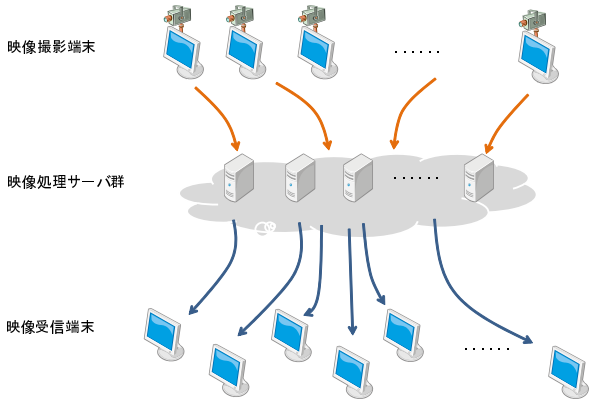

図 2　映像処理配信システムの配信環境

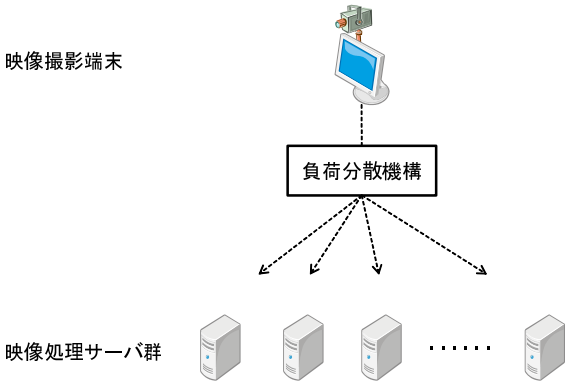

図 3　負荷分散の実現方式

図 4　映像撮影端末と映像処理サーバの内部構成

本研究では，映像撮影端末が映像処理サーバを選択する部分に着目して映像処理サーバの負荷分散を実現する．既存のシステムでは，映像撮影端末が映像を送信する時点において，映像撮影端末ソフトウェア上の設定により固定的に映像処理サーバを選択している．このため，映像処理サーバが既に高負荷状態であっても映像撮影端末からのさらなる接続が行われ，結果として過負荷による遅延の増大や処理落ちが生じることとなる．本研究では，映像撮影端末が映像処理サーバに接続する際に，その接続に割り込み，その接続先を複数台の映像処理サーバからランダムに選択することで負荷分散を行う（図 3）．

3.2　負荷分散機構の実装

図 4 に映像撮影端末と映像処理サーバの内部構成を示す．映像撮影端末には映像撮影端末ソフトウェアと PIAX プロセス，映像処理サーバには映像処理サーバソフトウェアと PIAX プロセスを配置する．この中で，映像撮影端末ソフトウェアと映像

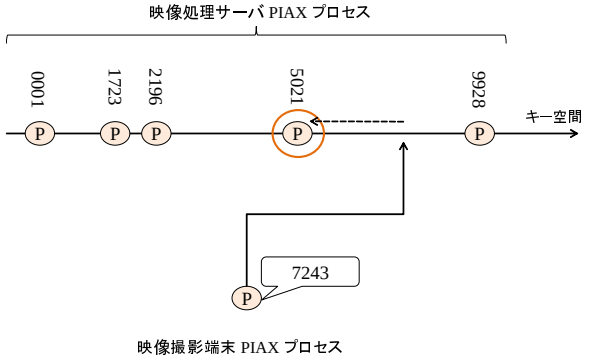

図 5　MaxLessThan 検索による映像処理サーバの選択

処理サーバソフトウェアは既存のシステムのものとなる．PIAX プロセスは，PIAX のオーバレイネットワーク探索機能を用いて，複数ある映像処理サーバから 1 台をランダムに検索する機能を実現する．

PIAX [3] とは， Java ベースのプラットフォームミドルウェアで，オープンソースソフトウェアとして公開されている(注1)．PIAX では，構造化オーバレイネットワークの一種である SkipGraph を用いることで，P2P ネットワークにおいて MaxLessThan 検索を実現している．MaxLessThan 検索とは，検索クエリに指定された値を超えない最大の値を持つ要素を抽出する検索で，例えば検索対象に 1, 6, 15, 24 という値がある場合に，10 を検索すると 6 が抽出されるという検索となる．本実装では，この MaxLessThan 検索により，複数ある映像処理サーバから自律的に 1 台のみを選択する．図 5 に検索の模式図を示す．まず，映像処理サーバの PIAX プロセスでは，起動時にランダム値を生成し，その値を SkipGraph オーバレイネットワーク上に公開する．この値は SkipGraph の特性により，オーバレイネットワークのキー空間に大小の従い順に整列する．映像撮影端末の PIAX プロセスでは，映像処理サーバを検索する際に，ランダムな値を生成し，この値を検索クエリとして MaxLessThan 検索を行う．図では，映像撮影端末が 7243 を生成し，MaxLessThan 検索をした結果，7243 を超えない最大の値である 5021 を持つ映像処理サーバの PIAX プロセスが発見されている様子を示している．実際の検索クエリは，SkipGraph の MaxLessThan 検索アルゴリズムに準じて PIAX プロセス間を転送されることで，最終的に 5021 を持つ PIAX プロセスに到達することとなるが，その詳細については本稿では触れないこととする．SkipGraph 上での MaxLessThan 検索の実現方法については，公開されている PIAX の実装を参照いただきたい．

3.3　実装システムの動作

初期設定として，各 PIAX プロセスには初期接続先として，PIAX プロセスが動作する任意の端末・サーバの IP アドレスと PIAX プロセス待ち受けポート番号を与える．映像撮影端末ソフトウェアは，事前にその接続先を localhost の PIAX プロセスの映像待ち受けポート番号に設定しておく．また，映像処理サーバでは，映像処理サーバソフトウェアの待ち受けポート番号と映像処理サーバの IP アドレスを映像処理サーバの PIAX プロセスに与える．それぞれのプロセスを起動すると，各 PIAX プロセスは，その初期接続先情報に従って接続を試み，SkipGraph をはじめとするオーバレイネットワークを構築し，相互に接続する．

(注1)：PIAX: P2P Interactive Agent eXtensions - http://www.piax.org/

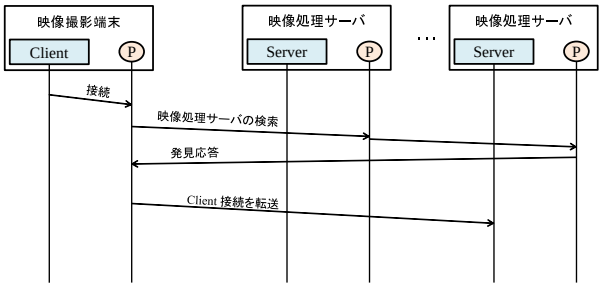

図 6　動作シーケンス

起動が完了した後，映像撮影端末より配信を開始する際の動作シーケンスを図 6 に示す．

まず，映像撮影端末ソフトウェアがユーザの配信開始の指示により，配信先に接続する．この接続先は localhost の PIAX プロセスとなっているため，図のように localhost 内での接続となる．映像撮影端末の PIAX プロセスは，この接続をきっかけとして，先述した MaxLessThan 検索により映像処理サーバの 1 台に検索要求が届く．映像処理サーバの PIAX プロセスはこの検索要求に対し，初期設定で与えられた映像処理サーバソフトウェアの待ち受けポート番号と映像処理サーバの IP アドレスを応答する．その応答を元に，映像撮影端末の PIAX プロセスは映像処理サーバソフトウェアとの間に接続を確立し，映像撮影端末ソフトウェアと映像処理サーバソフトウェア間の通信を中継を始める．

映像処理サーバが高負荷となった場合に生じる動作を図 7 に示す．まず，自身が高負荷状態と判断した映像処理サーバソフトウェアは localhost の PIAX プロセスの制御ポートに接続し，検索からの離脱要求コマンドを送る．この検索からの離脱要求コマンドを受けた PIAX プロセスは，図 5 に示したキー空間より自身の値を削除する．これにより，それ以降の映像撮影端末からの検索の対象から外れ，現在以上の接続は生じないこととなる．映像処理サーバソフトウェアの負荷が下がり，再び新たな接続を許容できる状態となると再び localhost の PIAX プロセスの制御ポートに接続し，検索からの離脱解除要求コマンドを送る．このコマンドを受けた PIAX プロセスは，再びキー空間上に自身の値を公開し，映像撮影端末からの検索を受け付ける状態となる．

以上の動作により，映像撮影端末からの新規接続先が映像処理サーバソフトウェア負荷状態に応じて制御されることとなる

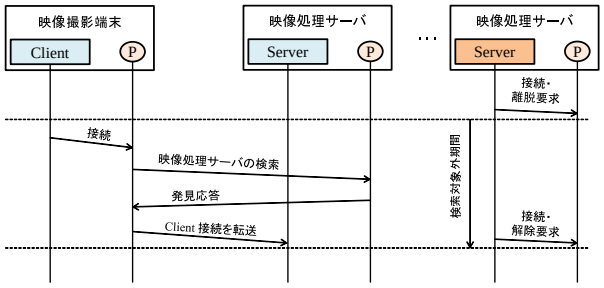

図 7　映像処理サーバ高負荷時の動作シーケンス

ため，映像処理サーバソフトウェアの過負荷を避けることができる．また，全てのサーバが検索の対象から外れている場合は，映像撮影端末ソフトウェアは単純に接続に失敗する．これにより，ユーザは映像処理サーバ群の余剰リソースが枯渇していることを知ることができる．

本負荷分散機構において，既存ソフトウェアに必要となる修正は，映像処理サーバソフトウェアの負荷計測と PIAX プロセスへの通知部のみにとどまり，映像撮影端末ソフトウェアは接続先を localhost に変更するのみで従来と透過的に扱う事ができる．一方で，負荷状況の計測と検索からの離脱・離脱解除の判断を映像処理サーバの PIAX プロセスが行えば，映像処理サーバソフトウェアに修正は不要となる．しかしながら，その場合は CPU の負荷から間接的に映像処理サーバソフトウェアの負荷を判断することになり，映像加工の推移を見越した負荷判断はできなくなるため，高負荷が続く状況にあるにも関わらず，一時的な負荷の現象により高負荷状態を脱したと誤判断し，新たな接続を受けてしまう場合が生じる．現時点以降の負荷予想ができるのは，処理を行っている映像処理サーバソフトウェアのみであるため，精度よく負荷判断を行うためには，その判断を映像処理サーバソフトウェア自身がすることが望ましいと考える．

また，本研究に類似する負荷分散手法として，リバースプロキシによる負荷分散が Web サーバの負荷分散などに一般的に用いられているが，本研究では P2P 技術を用いたことで，それぞれの端末・サーバが自律的に動作し，負荷分散先の端末の状態管理を集中して行うこと無く，任意の時点での映像処理サーバの追加や削減を容易に実現できることとなった．

本研究では実装負荷を軽減するため，負荷分散のアルゴリズムとして実質的にランダムとなる手法を採用したが，映像処理サーバ群全体が均等な負荷となるように制御したい場合には不十分と考えられる．その場合は，映像撮影端末からの接続先を選択する際に，映像撮影端末が初期に送出する制御フレームの情報を用いるとともに，映像処理サーバの負荷状況を収集し，これらの情報に基づき現時点で負荷が低い映像処理サーバに接続を中継するという手法も考えられるため，前提条件も含めて検討を続けたい．

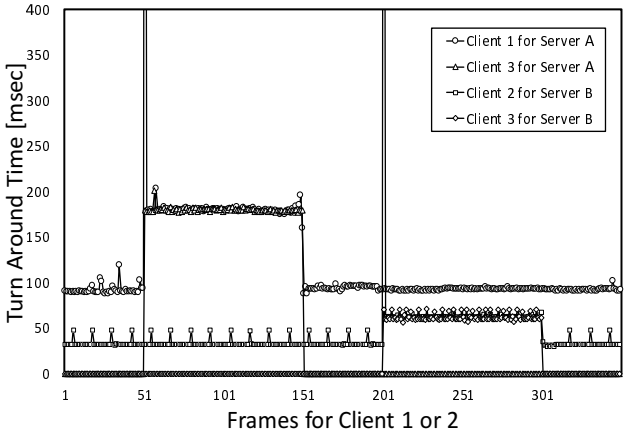

図 8　実 験 結 果

4. 実　　験

本研究では，提案する負荷分散方式を用いて負荷を分散できるか実験を行った．

4.1 実 験 環 境

2 コアの CPU を搭載したサーバ A と 4 コアの CPU を搭載したサーバ B の 2 台を用いる．3 台のクライアントには Web カメラが接続されており，取得した画像をサーバに送信して処理する．クライアント 1 はサーバ A を利用しており，クライアント 2 はサーバ B を利用している．各サーバを用いた場合の負荷を調査するため，クライアント 3 はまずサーバ A を利用し，その後サーバ B を用いる．これらの計算機は同一の無線 LAN に接続されている．

フレームあたりの映像処理にかかる時間を短縮することで高いフレームレートを実現できるため，映像処理システムでは，映像処理にかかる時間を短縮するために負荷を分散させることが考えられる．そこで，サーバに 1 フレームの画像を送信してから映像処理を完了するまでの時間（ターンアラウンドタイム）を負荷と捉え，ターンアラウンドタイムを計測した．

4.2 実 験 結 果

実験結果を図 8 に示す．横軸はクライアント 1 またはクライアント 2 が送信したフレームの数であり，縦軸はターンアラウンドタイムである．実験中，クライアント 1 はサーバ A，クライアント 2 はサーバ B を常に利用している．クライアント 3 は 50 番目のフレームから 150 番目のフレームが送信されている間サーバ A を利用し，200 番目のフレームが送信されるまで待機した後，300 番目のフレームが送信されるまでサーバ B を利用する．

クライアント 1 のターンアラウンドタイムは，50 番目のフレームを送信するまで約 90msec だが，その後，約 180msec になっている．これは，クライアント 3 がサーバ A を利用することでサーバ A の負荷が上昇し，映像処理に時間がかかったためである．この間，クライアント 3 のターンアラウンドタイムも約 180msec になっている．また，クライアント 2 のターンアラウンドタイムは，200 番目のフレームを送信するまで約 32msec だが，その後，約 65msec になっている．これは同様

に，クライアント 3 がサーバ B を利用することでサーバ B の負荷が上昇し，映像処理に時間がかかったためである．これより，サーバ A にクライアント 1 が接続した時点でサーバ A は高負荷と判断し，提案方式を用いて新たな接続を受け付けないようにすることで，クライアント 3 はサーバ B に接続することになってサーバ A に接続するよりもターンアラウンドタイムを短くできることが分かる．クライアント 2 のターンアラウンドタイムが 32msec 程度長くなるが，サーバ A に接続してクライアント 1 のターンアラウンドタイムが 90msec 程度長くなる場合に比べて影響を少なくできる．

クライアント 3 が接続した時点でクライアント 1 やクライアント 2 のターンアラウンドタイムが非常に長くなっているが，これは，クライアント 3 の接続開始処理に時間がかかるためである．また，クライアント 2 のターンアラウンドタイムが周期的に約 50msec になっている．これは，カメラから映像の取得に時間がかかっているためであり，ハードウェアの都合によるものと考えられる．

5. 関 連 研 究

本章では，本研究に関連する研究について説明する．

映像撮影端末とは別の計算機で映像処理を行う分散型映像処理システムに関する研究が幾つか行われている．MediaPaaS では，ライブ放送のための PaaS (Platform as a Service) 形式のクラウドコンピューティングサービスを目的とし，映像の符号化，再符号化，配信および画像処理をサービス提供者側の計算機で行えるシステムを提案している [4]．画像処理の内容は時刻を表示する，ロゴを表示するといったインターネットライブ放送中に一貫して行われる内容であり，放送中に放送者の要求に応じて行えない．文献 [5] では，P2P ネットワークを用いてライブ放送を行う場合に，通信状況に応じて映像の再符号化を行う手法を提案しているが，放送者が明示的に再符号化を指定できない．また，静止画ではあるが，PictuAR では，撮影した静止画を動画サーバに転送して解析し，関連ある動画を検索している [6]．本研究で提案するシステムでは，放送者がプログラムした任意の映像処理を，要求に応じて行える点が異なる．

また，インターネットライブ放送のための映像配信システムに関する研究も行われている．文献 [7] では，パノラマ撮影された映像から，あたかもカメラでズームやパンを行ったような映像効果を与えて配信できるシステムを実装している．文献 [8] では P2P ネットワークを用いてライブ放送を行う場合に，少し前から視聴（追っかけ再生）できるシステムを実装している．これらの映像配信システムでは，放送者が任意の映像処理を行えない点が本研究で提案するシステムとは異なる．

インターネットライブ放送において，映像配信の遅延時間を削減する幾つかの手法が提案されている．SmoothCache 2.0 では，P2P ネットワークを用いてライブ放送を行う場合に，他のピアに映像データをキャッシュし，キャッシュしたピアから配信することで映像撮影端末にかかる通信負荷を低減させて遅延時間を削減する手法を提案している [9]．文献 [10] では，遅延時間を理論上最小にする P2P ネットワーク上の配信経路決定手法を提案している．また，HD 法では，1 対 1 の通信と同時に，1 対多の放送型配信を用いて同時に複数の視聴端末に映像データを送信することで，通信量を削減している [11]．提案システムにおいても，映像配信時にこれらの遅延時間短縮手法を適用できるが，本研究とは映像処理システムを対象としている点が異なる．

さらに，保存された映像データに対して映像処理を行うシステムに関する研究が行われている．文献 [12] では，カメラで撮影した映像データを，高い計算能力がある計算機に転送して映像処理を行うシステムを提案している．文献 [13] では，スマートフォン等の計算能力が低い映像撮影端末で撮影した映像を映像撮影端末に保存せずに直接，クラウドストレージ等の外部記憶装置に保存するシステムを提案している．これらのシステムは，保存された映像データを対象としており，インターネットライブ放送では利用できない．

6. ま と め

本研究では，ライブ放送のための分散型映像処理システムにおいて，映像処理サーバの負荷を分散させるため，複数の映像処理サーバから選択する方式を提案した．また，提案方式は P2P 型エージェントプラットフォームである PIAX を用いて実装した．実装システムでは，映像撮影端末が映像処理サーバに接続する際に，その接続に割り込み，その接続先を複数台の映像処理サーバからランダムに選択することで負荷を分散する．さらに，複数の映像処理サーバと映像撮影端末を用いて実験し，映像処理を完了するまでの時間への影響を確認した．

今後の課題としては，各映像処理サーバの負荷を収集し，負荷の低い映像処理サーバへ優先的に中継する手法の検討が考えられる．

謝　　辞

本研究の一部は，科学研究費補助金（基盤研究 A）「ユビキタス環境のためのトポロジコーディングによる全体プログラミング」（課題番号：23240010），および，NICT・大阪大学共同研究「大規模分散コンピューティングのための高機能ネットワークプラットフォーム技術の研究開発」，総務省戦略的情報通信研究開発推進事業（SCOPE）「放送通信融合環境による次世代モバイルビデオオンデマンド配信の研究開発」による成果である．

文　　献

[1] I. Sodagar, “The MPEG-DASH standard for multimedia streaming over the Internet,” IEEE MultiMedia, vol.18, no.4, pp.62–67, April 2011.

[2] 義久智樹，川上朋也，石　芳正，寺西裕一，“ライブ放送のための分散型映像処理システムの設計と評価,” 情報処理学会研究報告，第 2015-DCC-11 巻，pp.1–7，Nov. 2015.

[3] 吉田　幹，奥田　剛，寺西裕一，春本　要，下條真司，“マルチオーバレイと分散エージェントの機構を統合した P2P プラットフォーム PIAX,” 情報処理学会論文誌，vol.49，no.1，pp.402–413，Jan. 2008.

[4] B. Cheng, “MediaPaaS: A cloud-based media processing platform for elastic live broadcasting,” Proceedings of the 7th IEEE International Conference on Cloud Computing

(CLOUD 2014), pp.713–720, June 2014.
[5] J.-C. Wu, P. Huang, J.J. Yao, and H.H. Chen, “A collaborative transcoding strategy for live broadcasting over peer-to-peer IPTV networks,” IEEE Transactions on Circuits and Systems for Video Technology, pp.220–224, Feb. 2011.
[6] “スマートフォン向け AR（拡張現実）サービス”. Available at http://www.nttcom.co.jp/ar-saas/.
[7] V.R. Gaddam, R. Langseth, H.K. Stensland, P. Gurdjos, V. Charvillat, C. Griwodz, D. Johansen, and P. Halvorsen, “Be your own cameraman: Real-time support for zooming and panning into stored and live panoramic video,” Proceedings of the 5th ACM Multimedia Systems Conference (MMSys 2014), pp.168–171, March 2014.
[8] Y. Gotoh, T. Yoshihisa, H. Taniguchi, and M. Kanazawa, “Brossom: a P2P streaming system for webcast,” Journal of Networking Technology, vol.2, no.4, pp.169–181, Dec. 2011.
[9] R. Roverso, R. Reale, S. El-Ansary, and S. Haridi, “SmoothCache 2.0: CDN-quality adaptive HTTP live streaming on peer-to-peer overlays,” Proceedings of the 6th ACM Multimedia Systems Conference (MMSys 2015), pp.61–72, March 2015.
[10] J. Dai, Z. Chang, and G.S.H. Chan, “Delay optimization for multi-source multi-channel overlay live streaming,” Proceedings of the IEEE International Conference on Communications (ICC 2015), pp.6959–6964, June 2015.
[11] T. Yoshihisa and S. Nishio, “A division-based broadcasting method considering channel bandwidths for NVoD services,” IEEE Transactions on Broadcasting, vol.59, no.1, pp.62–71, March 2013.
[12] D. Gibbon and L. Begaja, “Distributed processing for big data video analytics,” IEEE ComSoc MMTC E-Letter, vol.9, no.3, pp.29–31, 2014.
[13] W.-C. Ting, K.-H. Lu, C.-W. Lo, S.-H. Chang, and P.-C. Liu, “Smart video hosting and processing platform for Internet-of-Things,” Proceedings of the 2014 IEEE International Conference on Internet of Things (iThings 2014), pp.169–176, Sept. 2014.