P2P 動画ストリーミングシステムの再設計と実装

2009SE073 稲山 裕太 2009SE198 中野 祐輔
指導教員：後藤　邦夫

1　はじめに

近年，YouTube や GyaO など動画を配信するストリーミング配信が広く普及しており，ユーザはいつでも動画を見ることが可能となった．今までのストリーミング配信ではクライアントサーバ配信を利用するため，ユーザからの大量アクセスによる負担に耐えることができる大規模サーバや広帯域回線が必要であり，多大なコストがかかる．そこで現在では P2P 技術の発達により，誰でも配信者となれる P2P ストリーミングが実現された．

P2P ストリーミングは，各ノードがストリームデータを受信すると同時にデータを複製し，下流のノードにデータを受け渡す方法を用いる．そのためサーバの負担を減らすことができ，ノードを減らすことができる．しかし，P2P ストリーミングではストリームデータを受け渡しているノードが接続を中断した場合，その下流のノードはストリームデータを受け取れなくなってしまう．また他ノードへの再接続までに時間が必要であり，その間ストリーミングが停止してしまう問題がある [2][3].

本研究は，過去の卒論 (発行 2011 年)[2] の引き継ぎであり，システムの改良を目的とする．[2] では，ノードが離脱した際にノードの再接続し，ストリーミングを再開するプログラムを実装し性能を評価したが，ノードの数が増えた場合にエラーや動画像の質の劣化が生じる．そこで本研究は，過去のプログラムを参考に問題点を解消した新しいルーティング方式を採用し，大多数ノードの時でもノード離脱処理をより効率よく行えるプログラムを再実装する．そして，ネットワークエミュレータを用いて仮想ネットワークを模倣し，複数のノードで P2P プログラムを動作し，改良後のシステムを評価する．また, 利便性を考慮するため, Qt4 を用いて GUI を作成し操作を行う．稲山裕太は主に P2P プログラムと実験環境の構築，中野祐輔は主に GUI 作成を担当した．

2　旧システムの概要

この節では，旧システムにおけるルーティング方式とノードの脱退処理について述べ，またそれぞれに対する問題点を挙げる．

2.1　旧システムのルーティング方式

旧システムのルーティング方式は以下 3 つの点を考慮して実装されている．

- 1 つのノードに対する下流ノードの数を至言し，利用可能帯域を越えないようにする．
- ストリーミングデータの中継回数を減らし，データの品質の劣化を防ぐ．
- IP ネットワークにおけるホップ数を減らし，ネットワークへのトラフィックを軽減する．

1 つのノードに対する下流ノードはストリーマの設定によって決定している．ストリーマはデータの配送を設定する際, データ送信に必要なバンド幅を入力する必要がある．この情報からその動画を中継するノードが一度に配送できるノードの数 (以下，配送人数) を決定する．

問題点

- ttl，IP ホップ数が考慮された処理が不完全であり，ネットワークへのトラフィック増加への対処が成されていない．

2.2　旧システムにおけるノードの脱退処理

旧システムでは，システムの安定性のため，ノード脱退に対する処理をしている．脱退を希望するノードは脱退の要請をストリーマに行い，処理が開始される．脱退を希望するノードの種類別で以下の処理を実行される．

1. ストリーマまたは視聴ノードの場合
   システム全体にストリーマが離脱し番組を中止することを伝える．システム内の番組表から番組を削除し，その番組を視聴していたノードの視聴状態を中止する．
2. ストリーミングを視聴，または中継していた場合
   ストリーマは下位ノード（一番最後に加わったノード）のアップストリーマ（送信元ノード）と配送先ノード情報を変更し、脱退するノードの代わりとして機能させる．

問題点

- ネットワーク障害などによるノードの予期せぬ脱退が起こった際の処理が考慮されていない．

3　新システムの実現

新システムでは旧システムで挙げられた問題点の解消とさらなるシステムの利便性向上を目標とする．

提案するシステムでは，3 つのプログラムによって構成される．

- Streamer: UDP でストリームを発生するプログラム．
- P2Pnode: Streamer から送られてきたデータを制御し受け渡すプログラム．
- Listner: P2Pnode から送られてきたデータを受け取るプログラム．

Streamer，Listener は外部プログラムであり，動画や音声を直接送信・受信するプログラムである． P2Pnode は

メインプログラムであり，これらのプログラムがそれぞれつながることで P2P ストリーミングが可能となる．

3.1 新システムのルーティング方式

新規参加ノードに対する処理では，旧システムでの手法に加え，ttl，IP ホップ数を考慮した処理を実現させた．また，後述するノードの脱退処理を迅速に行うべく，新規参加ノードに対する処理が完了後，各ノードの持つ隣接ノード情報を増やす処理を追加した．旧システムでの手法を図 1 に示す．

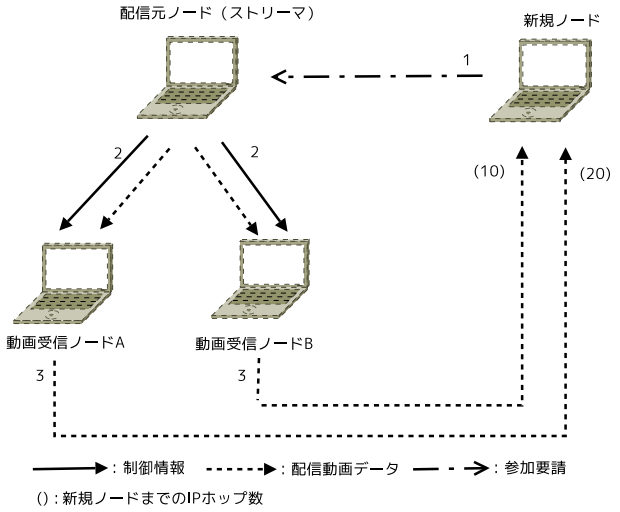

図 1 新規参加ノードのルーティングモデル

1. 新規ノードは視聴したい動画を配信しているストリーマに視聴要請を送信する．
2. 視聴申請を受信したストリーマは，配信人数を満たしていないノードの深さまで要請を中継する．図の場合，ストリーマは 2 つのノードにストリーミングデータを配信し 配送人数を満たしているため，中継ノードを持っていない参加ノード A とノード B に要請を送信する．
3. 要請を受信したノード A とノード B は，配送人数を満たしていないため，新規ノードにストリーミングデータを中継可能であることを送信する．
4. 新規ノードは中継可能の情報を受信した際の ttl か IP ホップ数の少ないノードに配信を要請し，ストリーミングデータを中継してもらう．

追加した処理の流れを図 2 に示す．

1. ノード A は中継可能であること意味する通知を受信した際の ttl か IP ホップ数の少ないノード B に配信を要請. ノード B は受信したノード A の情報（通信に必要となる IP アドレスやポート番号）を自らの隣接ノード情報として登録する．
2. ノード B は自分の配送先ノード C の情報を新規ノードに送信する．

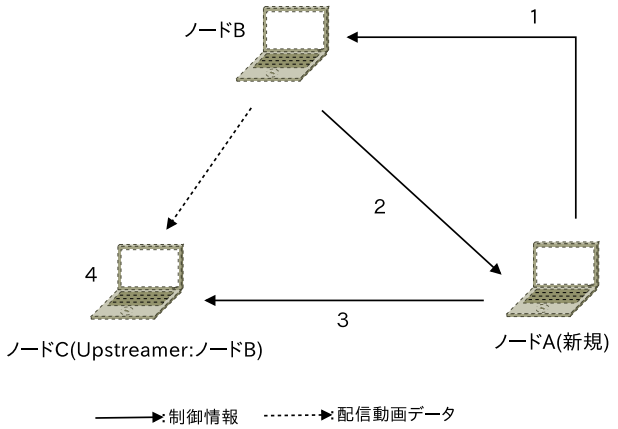

図 2 新規参加ノードのルーティングモデル

3. ノード A は受信したノード C の情報を自らの隣接ノード情報として登録し，ノード C に自分の情報を送信する．
4. ノード C は受信したノード A の情報を自らの隣接ノード情報として登録する．

3.2 新システムにおけるノードの脱退処理

新システムでは，動画視聴ノード数が多数の場合を考えたノード脱退処理システムを提案する．また，旧システムでの脱退処理に加え，ネットワーク障害などによる意図しないノードの脱退が発生した場合の処理を実現させる．

処理方法は，2 つの手法を用意しそれぞれ『末端ノード引き継ぎ法』と『直下ノード引き継ぎ法』とする．ノードより脱退の要請を受信したストリーマは前者の処理を行う．後者の処理は，ノードが一切の連絡なしに離脱をした際に，それを発見した周辺ノードが行う．動画を視聴する各ノードは一定時間毎にアップストリーマ（動画の送信元ノード）に対して Ping を送信し，存在の有無を確認する．

- 末端ノード引き継ぎ法

  ストリーマは末端ノード（最も新しく追加されたノード）を選択する．アップストリーマ（送信元ノード）情報， 送信宛先ノード情報，周辺ノード情報の 3 つを脱退するノードの持っていた各情報に変更し，脱退するノードの代わりとして機能させ，通信を続行させる．

  末端ノード引き継ぎ法を図 3 に示す．図中で使用する各変数について以下に示す．

  - stream：配信可能なノード数を表す．
  - limit：木の深さに対する配送可能な残りのノードを表す．ストリーマ側で使用される．
  - treesize：ノードの種類によって表す内容が異なる．

    リスナー：自分のいる位置 (木の深さ)

    ストリーマ：ストリーミングツリーのサイズ

- 直下ノード引き継ぎ法

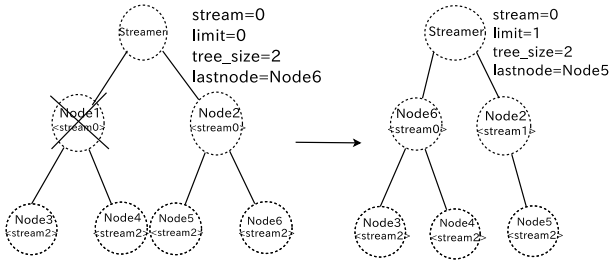

図 3 末端ノード引き継ぎ

直下ノード引き継ぎ法を図 4 に示す.

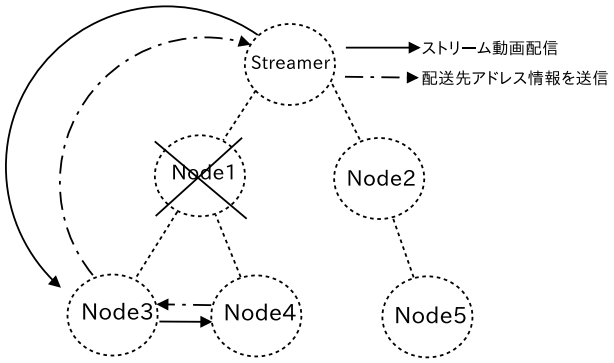

図 4 直下ノード引き継ぎ

ネットワーク障害などで，ストリーマに対し一切の連絡なしに脱退したノードを発見したノードが行う．脱退したノードの送信宛先ノード情報を引き継ぎ，通信を継続させる．

3.3 GUI の設計

新システムでは，よりシステムの利便性を向上させるために Qt4 を用いた GUI（Graphical User Interface）を設計する．

ストリーマーの IP アドレスやポート番号の記入など記入は端末で行い, リスナーの参加と脱退を GUI を用いて操作する．

4 実験

我々が提案するシステムは 100 ノード程度が P2P ストリーミングを用いて，動画や音楽データを送受信する．また， 先行研究 [2] より大規模システムを想定する．しかし実際に想定人数分の P2P ノードを起動し実験するのは困難である．そのため， 本研究では Goto‘s IP Network Emulator(以下 GINE[1]) を用いて適切な仮想ネットワークを構築し実験を行う．

GINE[1] とは，多数のルータやリンクで構成される広域ネットワークを模倣することができるネットワークエミュレータである．また，少数実機での実験も実施し，ノードの脱退処理の正確性を確認する．実験環境は，OS に Ubuntu10.04 を使用し，仮想ネットワーク上で expect スクリプトを用いて P2P プログラムを自動的に実行する．

また，後藤研究室内の PC を使用し，実機においてもシステムが正常に動作するかを確認する．そして最後に自作 GUI の動作確認実験を実施する．

4.1 ネットワークモデル

GINE で構築するネットワークのモデルを図 5 で示す.

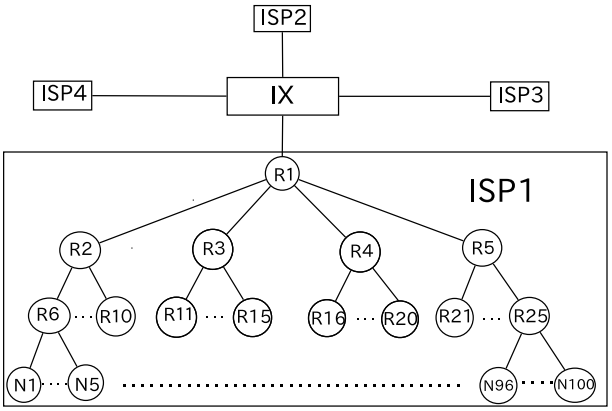

図 5 NetworkModel

一つの ISP につきノード 100 個で構成される IP ネットワークを構成し，提案した方式に基づき P2P ルーティングを行う．

現実のネットワークでは，ISP をまたがって Tree が構成され，ノード間の通信が異なる ISP 間をまたがって行われてしまうため，各ノードが，同一 ISP 内に存在しているという物理的ネットワークを構築して実験を行う．

4.2 expect スクリプト

仮想ネットワーク内で P2P プログラムを自動的に実行するために expect スクリプトを用いる．expect コマンドは対話的に使用するように作られたプログラムをスクリプト中に組み込み，定型的な作業を自動実行させることができる．P2P プログラムではストリーマとリスナーごとに expect ファイルを用意し，実行時に IP アドレスとポート番号を指定し起動する．

4.3 実験手順

GINE を用いた実験では，多数ノードでの各処理の安定性を確認する．

各ノードの expect スクリプトにツリーを形成するための処理を書き込み実行する. ストリーマは一つ. リスナー数は 99 ノードとする．実験のためストリーマが入力するデータサイズを調整し，ツリーを形成する．また，ノードが途中で脱退した場合，脱退処理が行われ，再接続が完了するまで，ストリーミングデータの中継は途切れ，下流ノードへのデータ転送が一時停止し通信に遅延が生じる．接続完了後，tcpdump コマンドを使用し，ノード間のパケットの流れを観察し，本研究であげた二種類のノード脱退処理をそれぞれ行った際の通信の遅延時間を計測する．

実機を用いた実験では，ノード離脱時の処理の正確性を

確認する．実験は後藤研究室内の PC を使用し，ノード数は 7 ノードで実施する．

各ノードの接続完了後，ひとつの PC に接続された LAN ケーブルを抜き，強制的にネットワークから離脱させ，適切な処理が行われるか確認する．

4.4 実験結果

100 ノードのツリーの形成は成功した．また脱退処理など各々の処理も問題なく実行することができた．脱退処理により生じる通信遅延時間は，末端ノード引き継ぎ法が平均 342msec．直下ノード引き継ぎ法が平均 170msec と測定することができた．これより，前者よりも後者の方が，システムに対する影響が少ないことが確認できた．個々のノードが持つべき情報は増えてしまうが，配信動画の劣化防止や，視聴者の満足度を第一に考えた場合，ユーザからの脱退申請を受け付けた際も，後者の処理を採用すべきだという結論に至った．

実機での実験では，離脱したノードは下位ノードにより即座に認識され，ネットワーク障害などによる脱退に対する処理として，直下ノード引き継ぎ処理が実行された．また，処理完了後ストリーミングデータも中継され，ノード間の再接続も完了したことも確認できた．

4.5 GUI 動作確認

自作 GUI を実際に動作させた結果を記す．GUI 機動後の MainWindow 画面を図 6 に示す．

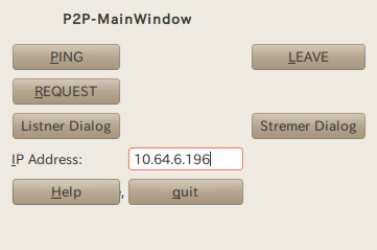

図 6 GUI 起動後の画面

各ボタンの説明を以下に示す．

- REQUEST:中央の空欄に配信元ノードのアドレスを入力することで，P2P ノードの設置され，新規参加の要請が送られ，リスナーポートの設置まで完了する．
- PING:中央の空欄に IP アドレスを入力すること，そのアドレスを持つノードへ ping を送信する．
- LEAVE:中央の空欄に配信元ノードのアドレスを入力することで，配信元ノードへ脱退申請が送信される．
- Listener Dialog:新たにリスナー設定を行う際に使用．
- Streamer Dialog:新たにストリーマ設定を行う際に使用．
- Help:GUI の操作についてのヘルプを表示する．
- quit:GUI 画面を閉じる．

実際に中央の空欄にに配信元ノードのアドレスを入力し“REQUEST ”ボタンを押した際の動作結果を図 7 に示す．

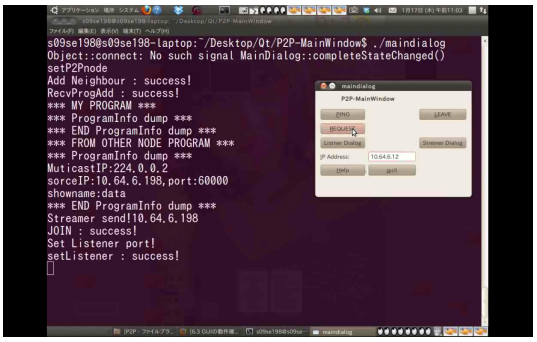

図 7 動作結果

5 おわりに

本研究では， システムの安定性向上を目的にシステムを再設計し，その性能を評価した．多数ノードでの実行も成功し，ノードの離脱によるノード間の再接続処理も完了した．また，離脱要請なしに離脱するノードに対する処理を追加し，ネットワーク障害などによるユーザの意図しない事態にも対応できるシステムを実現させた．また，自作 GUI の構築により，ユーザの利便性向上にも貢献した．

今後の課題として，以下 2 つがあげられる．

1. GUI の機能追加
2. 配信データの品質計測

プログラムの起動と配信元ノードへの脱退申請、アップストリーマの存在確認のための ping 送信は今回作成した GUI で実現したが, まだ受信したデータを表示，再生する機能などはないため追加する必要がある。また、動画を実際に送受信し品質を確認するまでは至っていないため、品質を計測し、品質確保のためのシステム改良も今後の課題となると考えられる。以上を実現することで、より利便性の高いシステムが完成する．

参考文献

[1] Goto, K.: Network Emulator with Virtual Host and Packet Diversion, *Cyber Journals: Multidisciplinary Journals in Science and Technology, Journal of Selected Areas in Telecommunications (JSAT), Vol. 3, No. 3*, pp. 13–20 (2012).

[2] 牧　祐希，長江　翔：Ｐ２Ｐストリーミング放送システムの改良，南山大学 情報通信学科 2010 年度卒業論文 (2011).

[3] 後藤祐輝，田中達也：Ｐ２Ｐストリーミングのためのルーティングの提案とそのエミュミレーション，南山大学 情報通信学科 2008 年度卒業論文 (2009).