平成 26 年度 修士論文

ロールベースの P2P ライブ
ストリーミングアーキテクチャの研究

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
1353015　鈴木 駿介

主任指導教員　末田 欣子　客員准教授
指導教員　多田 好克　教授
指導教員　古賀 久志　准教授

提出日　　平成 27 年 1 月 26 日

目次

図目次

表目次

第 1 章

はじめに

ネットワーク技術の発展に伴い, 近年ではリアルタイム動画配信サービスが人気となっている. このようなサービスはサーバから各クライアントに配信する, サーバ-クライアント方式が一般的であるが, 配信サーバのコスト削減などの理由で, サーバを介さずに直接クライアント同士で配信を行う P2P を利用したライブストリーミング配信が期待されている. しかし, 既存の P2P ライブストリーミングシステムでは, 途中参加したユーザはそれまでの配信内容を把握することが出来ないといった問題がある. そこで本研究では P2P ライブストリーミングにおいて, 途中参加したユーザがダイジェスト視聴可能な P2P ライブストリーミングシステムを提案する. 本研究では 2 つの段階を考えた. 1 つ目は P2P ネットワーク内でダイジェストを生成することである. 2 つ目は作成したダイジェストを P2P ネットワーク内で保持し, 広めるためのトポロジ設計を行うことである.

1 つ目の段階ではダイジェスト生成方式を提案した. 具体的には「増減率に基づくダイジェスト生成方式」, 「前後比較に基づくダイジェスト生成方式」, 「最小二乗法に基づくダイジェスト生成方式」の 3 つを提案した. 提案するシステムは参加ユーザが自由にコメント投稿出来ることを想定し, 3 つの方式ともコメントをした参加ユーザ数を利用した. 「増減率に基づくダイジェスト生成方式」ではユーザ数が増加している時を始点, 減少している時を終点とし, 始点から終点で最もユーザ数の多い点をダイジェストとした. 「前後比較に基づくダイジェスト生成方式」ではある点についてユーザ数が前後で急激に増減している点をダイジェストとした. 「最小二乗法に基づくダイジェスト生成方式」では最小二乗法を適用して 2 直線の重なりが鋭角な点をダイジェストとした. 実験では, ある動画に対して予め正解シーンを用意し, 提案方式がダイジェストとして選出したシーンとその正解シーンとの適合率で評価を行った. その結果「最小二乗法に基づくダイジェスト生成方式」が最も適合率が高く, 適切な方式であることがわかった.

2 つ目の段階ではトポロジを提案した. 具体的には複数クラスタ型において, 各ノードに役割をもたせたトポロジを設計した. 役割として「ゲートノード」, 「セミゲートノード」, 「ダイジェストノード」を用意した. ダイジェスト生成方式ではコメントをしたユーザ数を利用したが, トポロジ設計では各ノードの役割を決定するのに各コメント数におけるユーザ数の割合を利用した. ゲートノードは配信者ノードから配信されたコンテン

ツのパケットをクラスタ内で一番最初に取得する役目を担う. 高帯域かつコメント数の多いノードを選出した. セミゲートノードはクラスタ内でゲートノードから受信したパケットをクラスタ内部に拡散する役目を担う. ゲートノードの次に高帯域でコメント数の多いノードを選出した. ダイジェストノードはダイジェストの作成, また作成したダイジェストを保有し, 新規参加ノードへ送信する役目を担う. 最もコメント数の多いノードを選出した. 実験では NS-2 というシミュレーションを使って評価を行った. 役割の適切性に対する評価では, 役割を与えない場合ではノード数が増えるに従ってスループットの値が下がっていくが, 役割を与えた場合ではスループットの値が下がること無く, ノード数が増えても継続的に高いスループットを示しており, 役割を与えることの有用性を確認することが出来た.

第 2 章

背景

ネットワーク技術の発達に伴い, ネットワークを用いた動画配信サービスが人気である. 特にリアルタイム動画配信サービスが人気であり, 日本ではニコニコ生放送 [1] や TwitCasting[2], アメリカでは Ustream[3], 韓国では AfeecaTV[4] と, 近年世界中で急速に普及している. このようなサービスの形態としては, サーバから配信された映像をクライアントが視聴する, サーバ-クライアント方式が一般的である. そこで, 配信サーバのコスト削減や配信者の負荷を軽減するため, サーバを介さず直接クライアント同士で配信を行う P2P（peer to peer）を利用したライブストリーミング配信が期待されている. しかし, 既存の P2P ライブストリーミングシステムでは, 途中参加したユーザはそれまでの配信内容を把握することが出来ないといった問題がある. そこで本研究では P2P ライブストリーミングにおいて, 途中参加したユーザがダイジェスト視聴可能な P2P ライブストリーミングシステムを提案する.

本研究では 2 つの段階を考えた. 1 つ目は P2P ネットワーク内でダイジェストを生成することである. 2 つ目は作成したダイジェストを P2P ネットワーク内で保持し, 広めるためのトポロジ設計を行うことである. まずはじめにダイジェスト生成方式について述べ, その次にトポロジ設計について述べる.

2.1 ダイジェスト生成方式

動画配信サービスにおいてダイジェストを見ることは, 動画全体の雰囲気を知るために有用である. リアルタイム動画配信であるライブストリーミングにおいても, ユーザが途中から配信に参加した場合にそれまでの配信の内容を把握出来るという点においてダイジェストを見ることは有用である. しかし, P2P のライブストリーミングにおいては, サーバ-クライアント方式と比べてダイジェストを保存しておくサーバを用意することが出来ないといった問題があり, 既存のサーバ-クライアント方式のためのダイジェスト生成方式が適応出来ない. 以下にダイジェスト生成方式の既存研究をあげる.

2.1.1 動画に対するダイジェスト生成方式

橋本らはスポーツ映像を対象として, 映像メタデータと利用者の嗜好情報を利用したパーソナルダイジェスト生成方式 PDMS（Personal Digest Making Scheme）[10] を提案している. PDMS は, 映像メタデータから発生事象の重要度を自動的に検出し, 複数のダイジェストを選択する. 野球の映像を対象とし, 事前に選んだ正解集合と比較して適合率によってダイジェスト配信システムの重要度算出アルゴリズムの評価を行った.

PDMS では一度映像を整理してシーン毎に分類し, その上でダイジェストシーンを選出している. そのためライブ映像に対しては PDMS の手法を適応することが出来ない.

2.1.2 ライブ映像に対するダイジェスト生成方式

熊野らは野球の実況中継映像を対象として, 自動的にインデックス情報を付与して, ハイライトシーンを検出する, リアルタイムダイジェスト生成システム [11] を提案している. システムでは特に野球の PC（Picher and Catcher）シーンを画像解析により検出し, さらに音声解析により特別なイベントと判断されたキーワードを含む区間をハイライトシーンとして生成している. 実験では予め正解であるシーンを用意し, システムを適応した際の適合率により評価を行った. 結果は最も高い適合率で 97.2% という結果であり, 有用であることを示している.

熊野らの研究ではライブ映像に対してダイジェストをリアルタイムに生成している点が優れている. しかし, 野球という特定の分野の映像を対象にしているため, 様々な内容の動画配信に対応させることは難しい.

2.2 P2P ライブストリーミングのトポロジ

P2P ライブストリーミングは主にアプリケーションレベルマルチキャスト（ALM: Application Level Multicast）によって行われる．ALM はアプリケーションによって実現されるため開発が容易で多くの研究がなされている．

ALM は主にツリー型とメッシュ型に分類される．ツリー型は遅延が少なく構築が容易であるというメリットがあるが, 耐故障性やノードの離脱に弱いといったデメリットがある．メッシュ型は耐故障性やノードの離脱に強いというメリットがあるが, 複数の経路を用いるため遅延が大きくなるといったデメリットがある．一方でこれらツリー型やメッシュ型の欠点を補うために複数クラスタ型 [8],[9] が提案されている．複数クラスタ型は複数のクラスタを構築し各サブストリームをそれぞれのクラスタで配信するといった方法である．複数クラスタ型ではツリー構造のように中継ノードにストリームを渡すため, 配信の負荷が軽減される．またクラスタ内部では各ノードが複数の経路を持つため耐故障性に優れる．次ページ以降に複数クラスタ型の既存研究をあげる．

2.2.1 階層型クラスタ構造

Yang Guo らは階層的クラスタ HCPS（Hierarchically Clustered P2P Video Streaming）を提案している [5]. 図 2.1 は階層型クラスタ HCPS のトポロジ設計である. HCPS はクラスタ間で帯域のバランスを取ることにより, より品質の高い映像を流すことを可能にしている. また, 隣のピアへのストリームの受け渡しの際に, アップロード帯域幅を有効活用出来るようなスケジューリングアルゴリズムを提案している. クラスタ内部は完全結合となっており, 各クラスタのアップロード容量が均一になるように構成されている. クラスタを形成する際にクラスタ内で一番最初に配信内容を受け取るノードをヘッドノードと定義し, そのアップロード容量の大きいものを選出している.

HCPS はクラスタの中身が完全結合であり, 各ノードへ多くの経路を辿ることになるため遅延が多くなってしまうという課題がある.

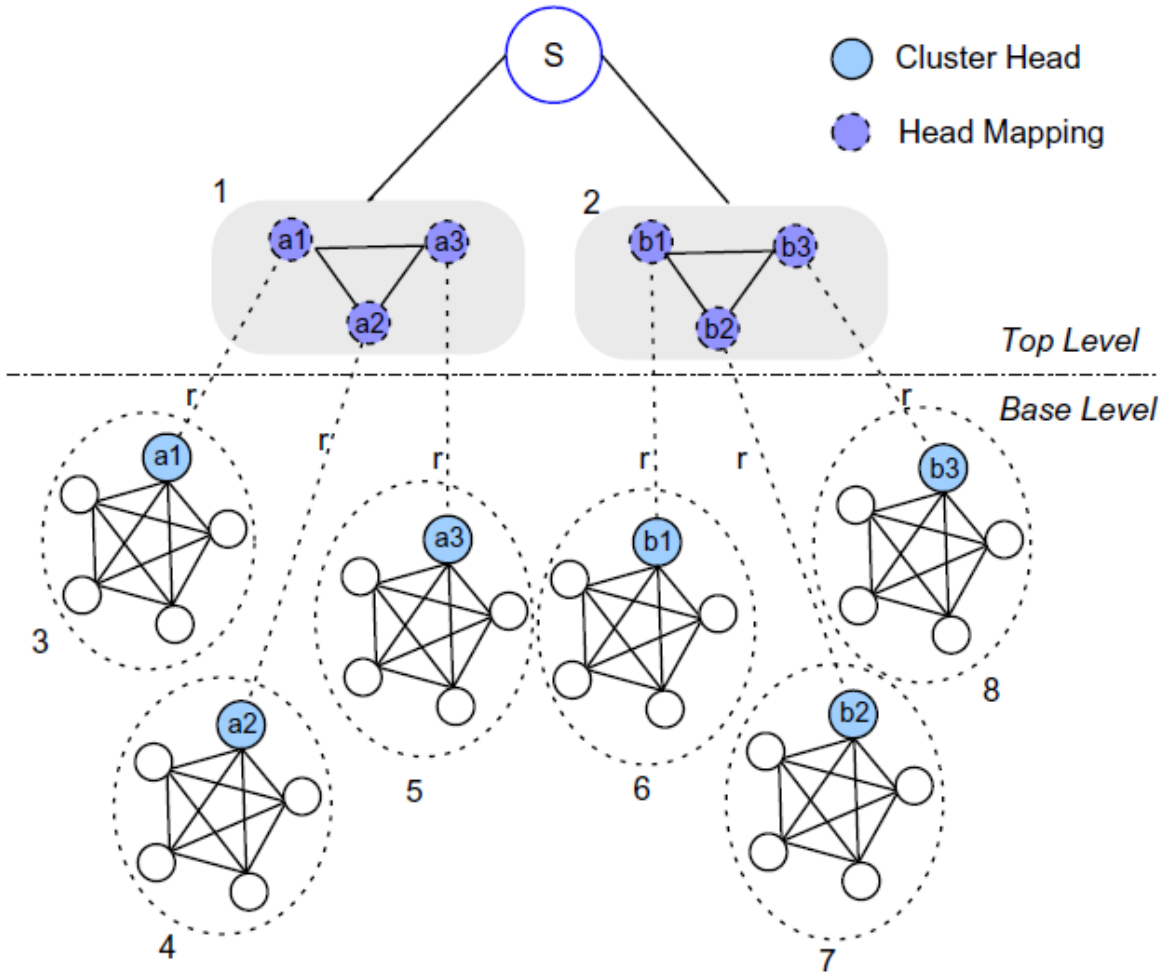

図 2.1 階層型クラスタ HCPS のトポロジ設計

2.2.2 重畳クラスタ木型動画配信システム

元橋らは重畳クラスタ木方式の動画配信システムを提案している [6]. 図 2.2 は重畳型クラスタ木型動画配信システムのトポロジ設計である. 階層的なクラスタ構造になっており, クラスタ間の配信木の中継ノードであるゲートノードが存在する. さらにゲートノードはクラスタ内で一番最初に配信内容を受け取り, それをクラスタ内に広める役割を担っている. ゲートノード選出方式として, P2P システム内での滞在時間の長さと RTT（Round Trip Time）を考慮した方法を提案している. 離脱耐性向上型, 配信遅延抑制型, ハイブリッド型の実験をしたところ, ハイブリッド型が最も性能が良いという結果が出ている.

重畳クラスタ木方式は全体としてツリー構造のため下位のクラスタほど配信者からのホップ数が大きくなってしまうという課題がある.

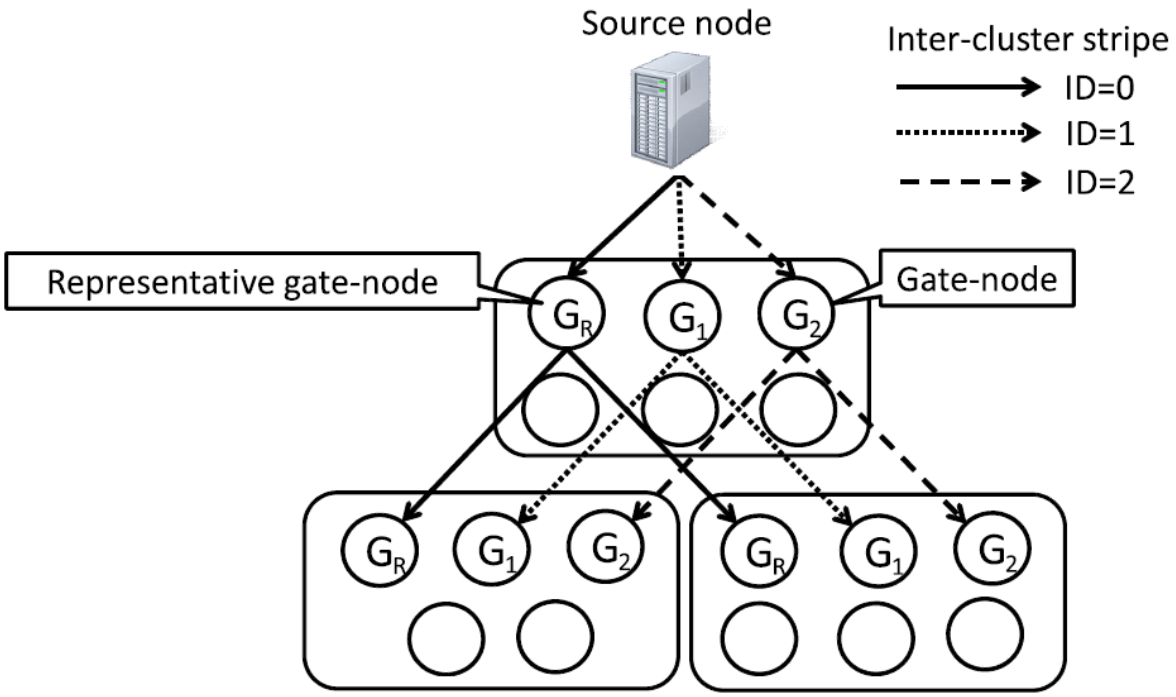

図 2.2 重畳型クラスタ木型動画配信システムのトポロジ設計

2.2.3 ハイブリッドアーキテクチャ構造

Huey-Ing Liu らは局所性と貢献度を考慮したハイブリッドアーキテクチャである MeTree を提案している [7]．図 2.3 は Metree のトポロジ設計である．MeTree は ISP（Internet Service Provider）ごとにクラスタリングを行いクラスタ内のそれぞれのノードをメッシュで接続し, 生成された各クラスタ同士をツリー構造で接続している．ツリー構造とメッシュ構造のハイブリッド構造となっている．配信内容をより多くのノードに広めるような貢献度の高いピアには質の高い映像を配信し, 逆に貢献度の低いピアには質の低い映像を配信する．異なる貢献度を持つピアに異なる QoE（Quality of Experience）を与えている．物理トポロジとオーバーレイを構築するためのピアの貢献度の両方を考慮することにより, 遅延を減少させている．

MeTree では貢献度を意識した設計のため貢献度の低いノードは良い映像が見られず, ネットワーク全体の QoE は低下してしまうという課題がある．

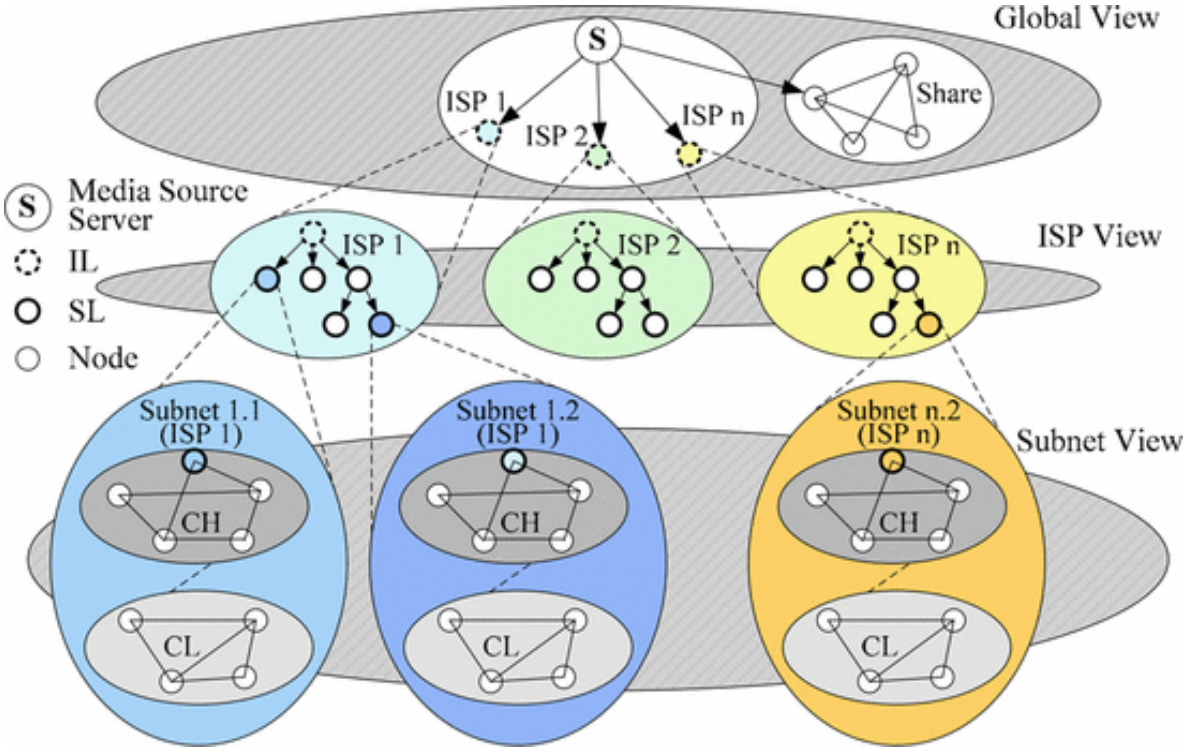

図 2.3　Metree のトポロジ設計

第 3 章

提案手法

本研究の目的は, 配信に途中参加したユーザがダイジェスト視聴可能な P2P ライブストリーミングシステムを提案することである．この目的を達成するために 2 つの段階を考える．1 つ目の段階として, まず P2P ネットワーク内でダイジェストを生成することを考える．動画コンテンツであったり, ライブ映像であっても, ダイジェストを生成するための専用の計算機やサーバを用意することが出来れば, 既存のダイジェスト生成方式を使える．しかし, 本研究では P2P ライブストリーミングにおいてダイジェストを生成することを考えるため既存の手法は使えない．そこで, P2P ライブストリーミングに特化したダイジェスト生成方式を提案する．2 つ目の段階として, P2P ネットワーク内で作成したダイジェストを保持し, 広めるためのトポロジ設計を行う．システムは P2P なのでダイジェストを管理する特別なサーバを用意することが出来ない．よって作成したダイジェストはサーバではなく各ピアが管理する．そのため各ピアが役割を持ったトポロジを設計する．

まず最初に 3.1 節でダイジェスト生成方式についての提案方式を述べ, 続く 3.2 節でトポロジ設計についての提案方式を述べる．

3.1 ダイジェスト生成方式

P2P ライブストリーミングを行う際に, 特別なサーバに頼ること無く参加している各ピアがダイジェストを生成出来るような方式を提案する. 本研究では 3 つの異なる手法に基づいたダイジェスト生成方式を提案する. 1 つ目は「増減率に基づくダイジェスト生成方式」(3.1.2 節), 2 つ目は「前後比較に基づくダイジェスト生成方式」(3.1.3 節), 3 つ目は「最小二乗法に基づくダイジェスト生成方式」(3.1.4 節) である.

3.1.1 ダイジェスト生成におけるユーザとコメント

本研究ではある特定の種類, 例えばスポーツなどの映像に特化したダイジェスト生成方式ではなく, どのような種類の映像にも適応可能な汎用的なダイジェスト生成方式を目指す. そのため, 本研究では P2P ライブストリーミングシステムに参加しているユーザとそのコメント数に着目した. 提案する P2P ライブストリーミングシステムでは, 参加ユーザが配信内容に対して自由にコメント投稿が出来ることを想定する. コメントは 1 ユーザにつき何回でも投稿出来るものとし, 各ユーザが投稿したコメント数はシステム内で管理しているものとする.

ダイジェスト生成においては単位時間当たりにコメントしたユーザ数を利用する. ユーザ数は 1 つのコンテンツを視聴しているユーザの数を意味する. ユーザ数はどの種類の映像のコンテンツであっても共通の指標として利用できるため, 汎用的なダイジェスト生成方式を目指すシステムに適している. ユーザ数はコンテンツの継続時間によって増減し, 特定の時刻のユーザ数が多いほど注目度が高いことを意味する. その注目度の高い特定の時刻のコンテンツの内容をダイジェストとして利用することを考える. 継続時間によって変化するユーザ数を用いて, 例えばユーザ数増減グラフ (図 3.1) を作成することが出来る. 提案する 3 つの方式では, ユーザ数増減グラフにおいて突出している部分「突出点」を発見することによりダイジェストを作成する.

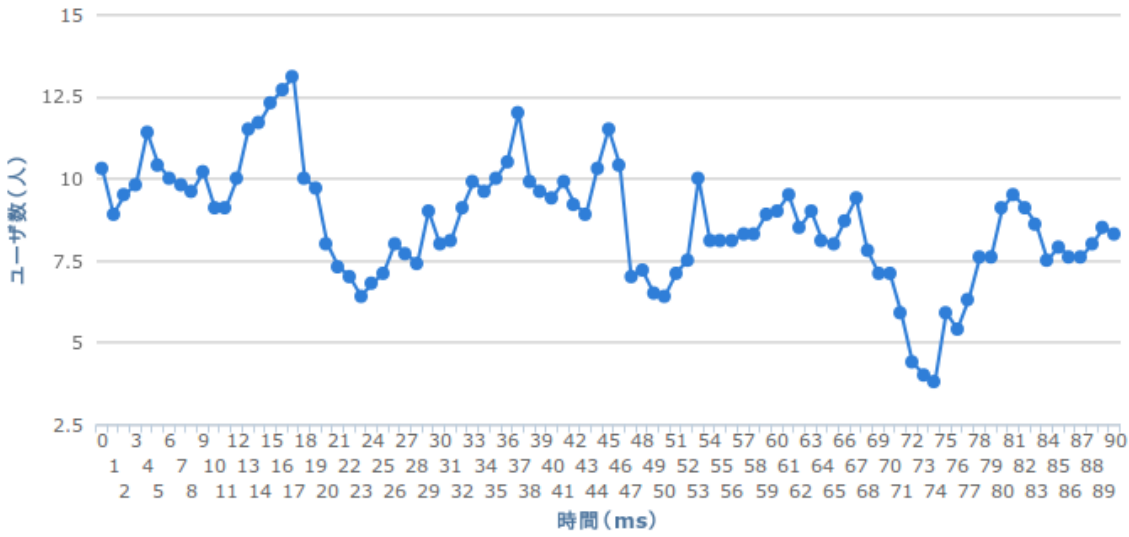

図 3.1　あるコンテンツのユーザ数増減グラフ

ダイジェスト生成方式を提案するにあたり, 達成すべき要求条件を以下にあげる.

- 特定の種類のコンテンツに依存しないダイジェスト生成方式であること
- P2P ライブストリーミングに特化したダイジェスト生成方式であること
- 生成されたダイジェストの映像によってコンテンツの内容の全体把握が出来ること

3.1.2　増減率に基づくダイジェスト生成方式

1 つ目は「増減率に基づくダイジェスト生成方式」である. これはユーザ数が増加している時を始点, ユーザ数が減少している時を終点とし, 始点から終点までで最もユーザ数の多い点を「突出点」として抽出する方式である. ユーザ数の増加率と減少率をそれぞれ閾値として決定する. 増加率を超えた時の値を始点とし, その後減少率を超えた値が出現するまで探してその値を終点とする. これは以下の式 3.1 に従う.

$$\text{始点}: \frac{X_{t+T} - X_t}{X_t} \geq Th_{inc}$$
$$\text{終点}: \frac{X_{t+T} - X_t}{X_t} \leq Th_{dec}$$
$$\text{なお}, X_t\text{は始点または終点となる.}$$
$$(X_t\text{は時刻 } t \text{ におけるユーザ数}, T \text{ は一定期間}, Th_{inc}\text{と } Th_{dec}\text{は閾値}) \qquad (3.1)$$

図 3.2 は増減率に基づくダイジェスト生成方式を適用したグラフである. 縦に赤い線が始点であり, 縦に青い線が終点である. 始点は数分間のユーザ数の増加率を計算したのち,

その増加率が閾値を超えた点を示している．終点は数分間のユーザ数の減少率を計算したのち，その減少率が閾値を超えた点を示している．縦に緑の線は「突出点」を示しており，この部分をダイジェストとして抽出する．

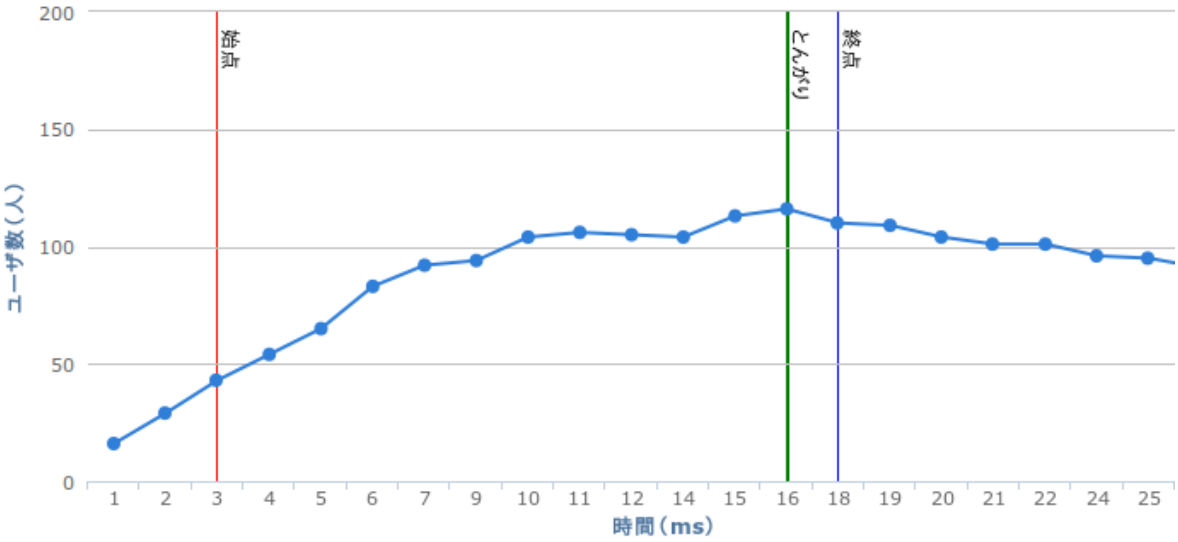

図 3.2　増減率に基づくダイジェスト生成方式を適用したグラフ

3.1.3　前後比較に基づくダイジェスト生成方式

2 つ目は「前後比較に基づくダイジェスト生成方式」である．これは，ある点についてユーザ数が前後で増加かつ減少している場合を「突出点」として抽出する方式である．この方式も 1 つ目の方式と同様にユーザ数の増加率と減少率をそれぞれ閾値として決定し，それぞれの値を超えた場合に「突出点」を決定する．これは以下の式 3.2 に従う．

$$\frac{X_t - X_{t-T}}{X_t} \geq Th_{inc} \wedge \frac{X_{t+T} - X_t}{X_t} \geq Th_{dec} \Rightarrow X_t \text{は突出点}$$
$$\frac{X_t - X_{t-T}}{X_t} \geq Th_{inc} \wedge \frac{X_{t+T} - X_t}{X_t} \geq Th_{dec} \Rightarrow X_t \text{は突出点}$$
$$(X_t \text{は時刻 } t \text{ におけるユーザ数}, T \text{ は一定期間}, Th_{inc} \text{と } Th_{dec} \text{は閾値}) \qquad (3.2)$$

図 3.3 は前後比較に基づくダイジェスト生成方式を適用したグラフである．縦に緑の線は「突出点」を示しており，この部分をダイジェストとして抽出する．1 つ目の方式との違いは，先に「突出点」となる候補を探し，その点が正しい「突出点」であるかを前後の増減率を計算して決定している点である．

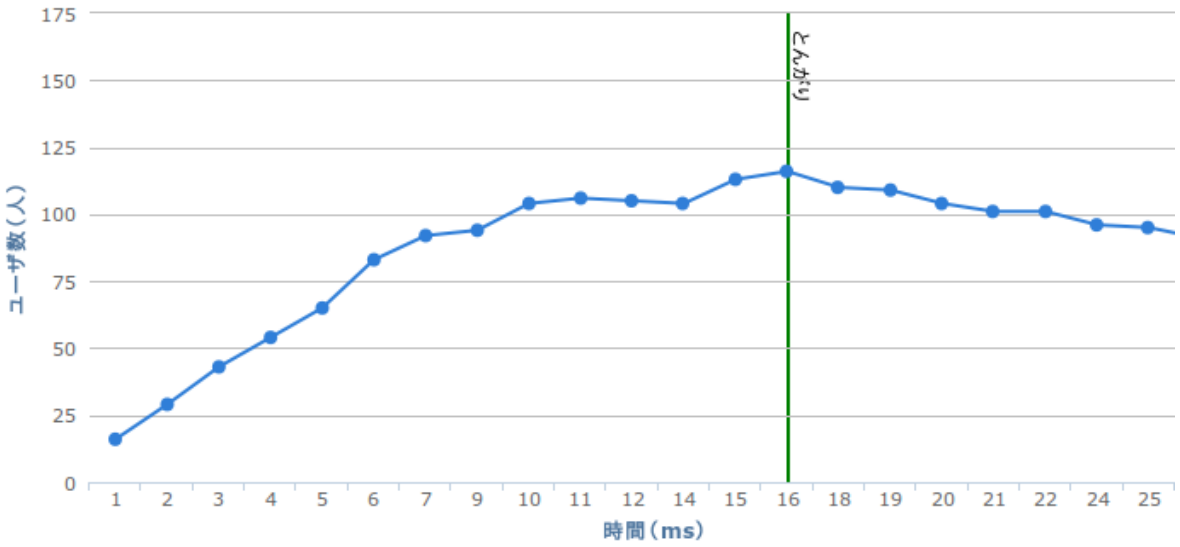

図 3.3　前後比較に基づくダイジェスト生成方式を適用したグラフ

3.1.4　最小二乗法に基づくダイジェスト生成方式

3 つ目は「最小二乗法に基づくダイジェスト生成方式」である．これは, ユーザ数増減グラフに数点間の最小二乗法を適用し, それによって近似された 2 直線の重なりが鋭角な点を「突出点」とし抽出する方式である．最小二乗法を適用した際の直線の重なりの角度を閾値として決定し, その値を超えた場合にその点を「突出点」として決定する．これは以下の式 3.3 に従う．

$$\arctan\left(\frac{slope_{line2}-slope_{line1}}{1+slope_{line2}*slope_{line1}}\right)\leq Th_{Angle}$$
$$\Rightarrow \text{直線の重なり部分が突出点}$$
$$(slope_{line1}\text{および}\ slope_{line2}\text{は直線の傾き}, Th_{Angle}\text{は閾値}) \tag{3.3}$$

図 3.4 は最小二乗法に基づくダイジェスト生成方式を適用したグラフである．点と点の間のカラフルな線が最小二乗法を適用した時の直線であり, 縦に緑の線が「突出点」を示しており, この部分をダイジェストとして抽出する．

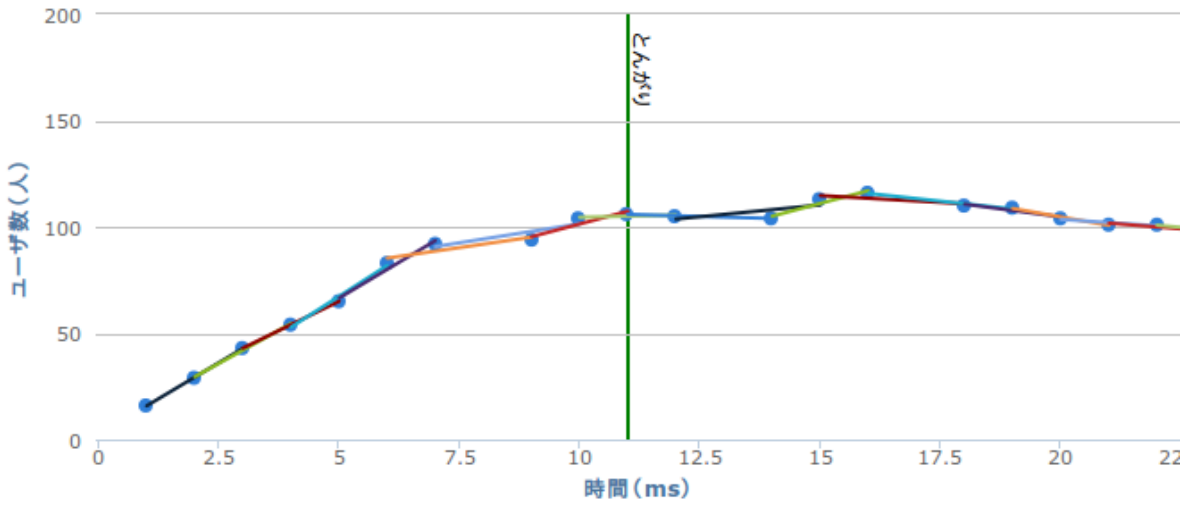

図 3.4　最小二乗法に基づくダイジェスト生成方式を適用したグラフ

3.1.5　既存のダイジェスト生成方式との比較

関連研究であげた既存のダイジェスト生成方式との比較の結果を表 3.1 に示す．橋本らの提案したスポーツ映像を対象とした方式を「方式 1」, 熊野らの提案した野球の実況中継映像を対象とした方式を「方式 2」, 本研究で提案する方式を「提案方式」とする．

表 3.1　既存のダイジェスト生成方式との比較

	方式 1	方式 2	提案方式
リアルタイム性	対応していない	対応している	対応している
P2P への対応	対応していない	対応していない	対応している
汎用性	スポーツに特化	野球に特化	様々な種類に適用可能

方式 1 や方式 2 に比べ, 提案方式では P2P ライブストリーミングにおいて汎用的なダイジェスト生成方式であることがわかる．

なお, 本節で提案した 3 つの方式は 4.1 節にて具体的なスレッショルド値等を与え, その優劣を評価する．

3.2 トポロジ設計

これまでは, P2P ライブストリーミングにおいて汎用的なダイジェスト生成方式について述べてきた. ここでは生成されたダイジェストを P2P ネットワーク内で保持し, それを配信するためのトポロジ設計を提案する.

トポロジの構成としては主にツリー構造とメッシュ構造が存在する. ツリー構造とは, 1 つのノードが複数の子を持ち, 1 つの子が複数の孫を持つという形になっており, 枝分かれをしながら階層が深くなっていく構造のことである. 木が幹から枝に, 枝から葉に分かれていく様子に似ていることからツリー構造と言われている. ツリー構造は構造が単純なため構築が容易であるが, その反面 1 つのノードが居なくなるとそれより下の子全員が切り離されてしまうので耐故障性に弱いという特徴がある. メッシュ構造とはツリー構造のように親と子のみと接続するのではなく, 複数のノードが接続し相互に通信を行う構造のことである. 複数のノードが接続しているので網の目（mesh）のように見えるためメッシュ構造と言われている. メッシュ構造はツリー構造のように特定のノードからのみ受信するのではなく, 複数のノードから受信できるので耐故障性に優れている. しかし必然的に各ノードの接続数が増え, 複数経路を通るため遅延が大きくなってしまうという問題がある. これらの問題を解決する構造として, 関連研究で紹介した複数クラスタ型が登場した. 複数クラスタ型はツリー構造のように中継ノードにストリームを渡し, 各中継ノードはそれぞれクラスタを作成してクラスタの中ではメッシュ構造のように複数のノード同士が接続している構造である. 複数クラスタはツリー構造とメッシュ構造の欠点を補っている. 本研究では複数クラスタ型のトポロジ設計にする.

複数クラスタ型では中継ノードを選ぶ必要がある. また, 本研究では P2P のライブストリーミングシステムを対象としているため, ダイジェストを管理するためのサーバを用意することが出来ない. そのため, 生成されたダイジェストは特定のノードに管理させることにする. さらに, 中継ノードへの負担を軽減するためにクラスタ内へストリームを広めるためのノードも必要である. このように, 各ノードに役割を持たせたトポロジ設計にする. 以下では, 中継ノードを「ゲートノード」, ダイジェストを管理をするノードを「ダイジェストノード」, クラスタ内へストリームを広めるためのノードを「セミゲートノード」と呼ぶこととする.

トポロジを提案するにあたり, 達成すべき要求条件を以下にあげる.

- ライブストリーミングの映像を見られるだけの性能があること
- 少ない遅延でパケットが届くこと
- ダイジェストを P2P ネットワーク内に確保できること

3.2.1 トポロジ設計におけるユーザとコメント

ダイジェスト生成においては単位時間当たりにコメントしたユーザ数を利用した．トポロジ設計においてもユーザ数とそのコメント数を利用する．あるコンテンツを視聴しているユーザのコメント数が多いということは, そのユーザが, 視聴しているコンテンツに対してアクティブであり, 配信から離脱する可能性が低いことを意味する．このような離脱可能性の低いノードを中継ノードである「ゲートノード」や, ダイジェストを管理する役割を担う「ダイジェストノード」に適用する．

ダイジェスト生成で紹介した図 3.1 は, 単位時間当たりにコメントしたユーザ数の増減グラフを表している．このグラフの値に 1 ユーザ当たりのコメント数を付与すると例えば以下のような表 3.2 が出来上がる．

表 3.2　単位時間における各ユーザ毎のコメント判別グラフの例

	時刻（ms）				合計（ユーザ毎のコメント数）
ユーザ	0	1	2	···	
ユーザ 1	○	○	×	···	15
ユーザ 2	○	○	○	···	5
ユーザ 3	×	×	×	···	8
⋮	⋮	⋮	⋮		
合計（時刻におけるコメントしたユーザ数）	4	6	10		

この表は単位時間における各ユーザ毎のコメント判別グラフの例である．コメントをした場合は○, コメントをしなかった場合は×が書かれている．この表において, n 番目の時間コメントしたユーザ数の合計をグラフにすると図 3.1 が出来上がる．ユーザ毎のコメント数の合計を見ると以下の図 3.5 のようになる．図 3.5 は各コメント数におけるユーザ数の割合を示しており, 離脱可能性の高いノードや, 逆に低いノードの割合を知ることが出来る．各ノードに役割を割り当てる際に, このグラフを利用する．

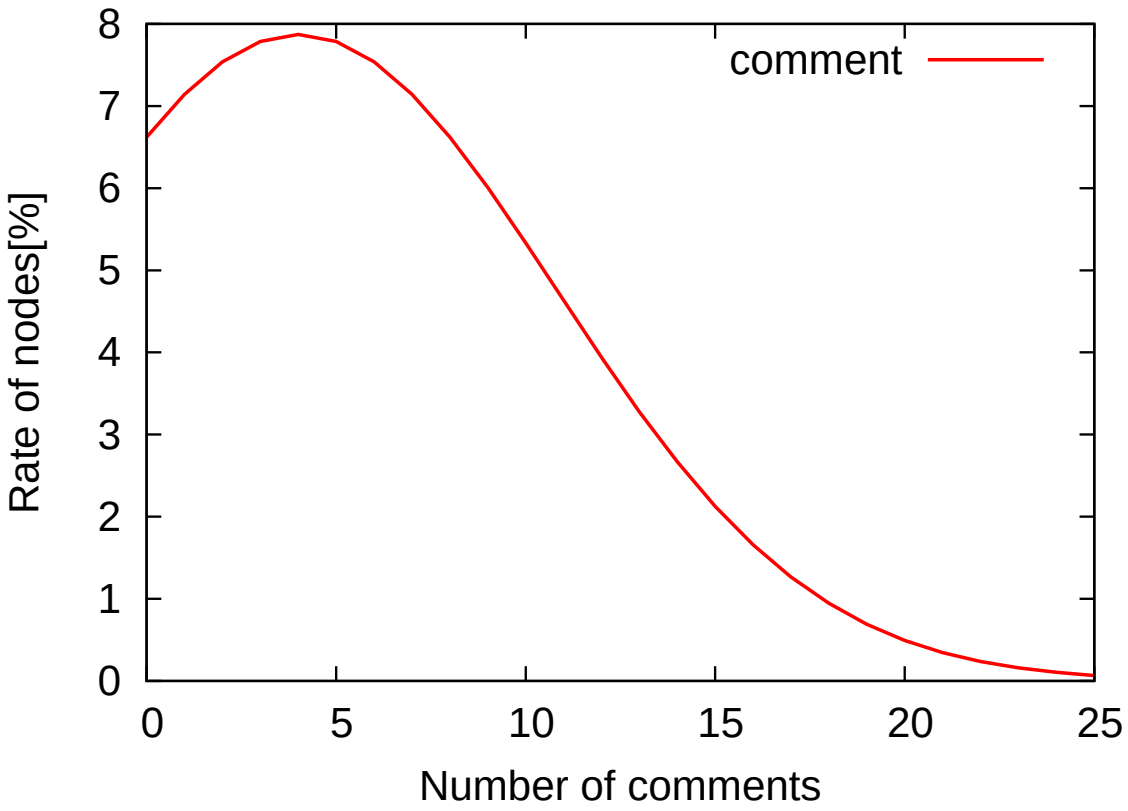

図 3.5　各コメント数におけるユーザ数

3.2.2　ノードの役割

各ノードにはそれぞれ役割を割り当てる．役割は「ゲートノード」，「セミゲートノード」，「ダイジェストノード」，「その他のノード」に分けられる．それぞれについて解説していく．

ゲートノード

配信者ノードから配信されたコンテンツのパケットをクラスタ内で一番最初に取得する．また, 他のクラスタのゲートノードと繋ぎコンテンツのパケットを送受信する役割を担う．

セミゲートノード

クラスタ内でゲートノードから受信したパケットをクラスタ内部に拡散する役割を担う．

ダイジェストノード

ダイジェストの作成を行う, また作成したダイジェストを保有し, 新規参加ノードへ送信する役割を担う.

その他のノード

その他のノードに, トラッカーサーバとノーマルノードが存在する. トラッカーサーバは全ノードのコメント数, 帯域幅, ホップ数を常に計算する. トラッカーサーバには全てのノードが接続し, 全てのノードの情報を管理する.

ノーマルノードはどの役割にも属さないノードのことである. ノーマルノードの中にはダイジェストを取得し終えたノードと, 新規参加したばかりでまだダイジェストを取得し終えていないノードが存在する.

3.2.3 ノードの役割決定方法

ノードの役割を決定する方法について述べる. 図 3.6 に図解したものを載せる.

各ノードは固有の帯域幅, コメント数を持っており, この値を元に役割を決定する.

ノードの役割は, ダイジェストノード, ゲートノード, セミゲートノード, ノーマルノードの順番で決定する. まず, 各ノードをコメント数の降順に並び替える. 並び替えたノードのうち, 上位のいくつかのノードをダイジェストノードとして選出する. 残ったノードのうちゲートノードとセミゲートノードの分のノードを取り出し, 帯域幅の降順に並び替える. 並び替えたノードのうち, 上位いくつかのノードをゲートノードとして選出する. さらにそのゲートノードを除いた上位いくつかのノードをセミゲートノードとして選出する. ゲートノードとセミゲートノードは同じ数だけ選出する. 残りのノードをノーマルノードとする.

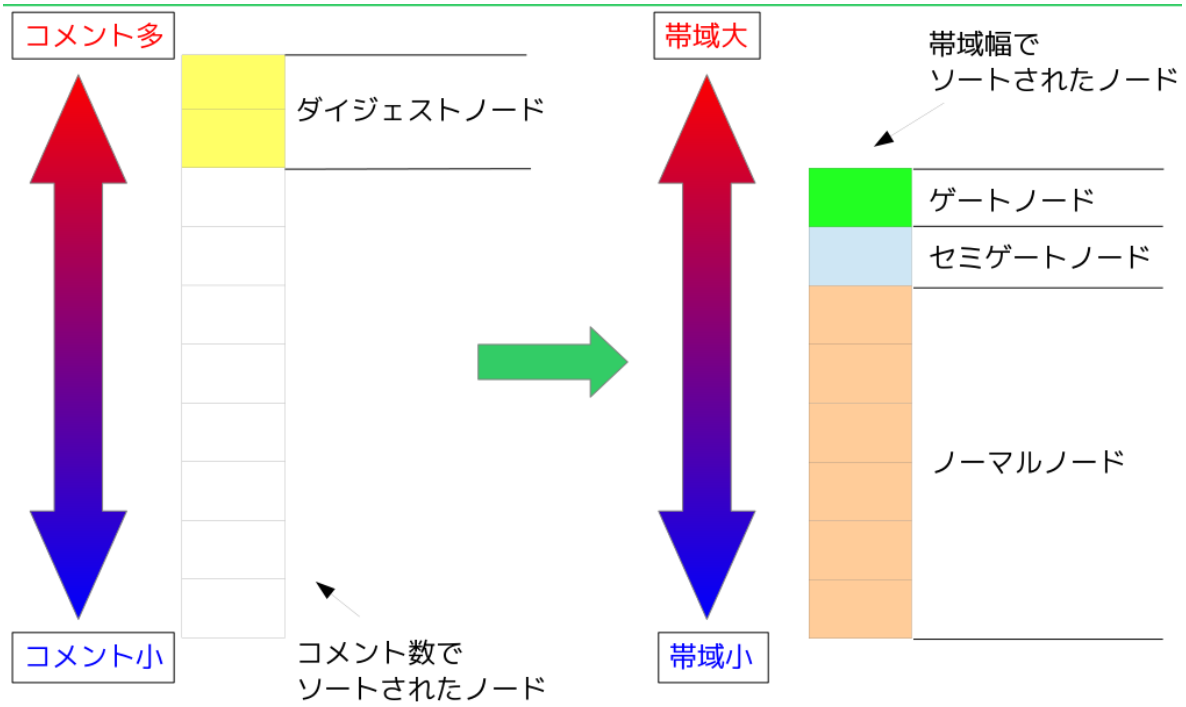

図 3.6　ノードの役割を決定する方法

3.2.4　ノード間の接続方法

図 3.7 は提案システムのトポロジの様子である．2 つのクラスタがあり，その外側の接続と，クラスタ内の接続について示している．

まず，クラスタ外部間での接続方法を示す．配信者ノードは全てのゲートノードと接続する．1 つのゲートノードは他のクラスタのゲートノード 1 つずつと接続する．

次にクラスタ内部での接続方法を示す．1 つのゲートノードは他の全てのゲートノードと接続する．また，全てのゲートノードは 1 対 1 の関係となるセミゲートノードが存在し，1 つのゲートノードはその対応するセミゲートノード 1 つずつと接続する．1 つのセミゲートノードはいくつかのダイジェストノードと接続する．またノーマルノードいくつかと接続する．1 つのダイジェストノードはダイジェスト未取得ノーマルノード全てと接続する．ノーマルノードは他のノーマルノードのいくつかと接続する．

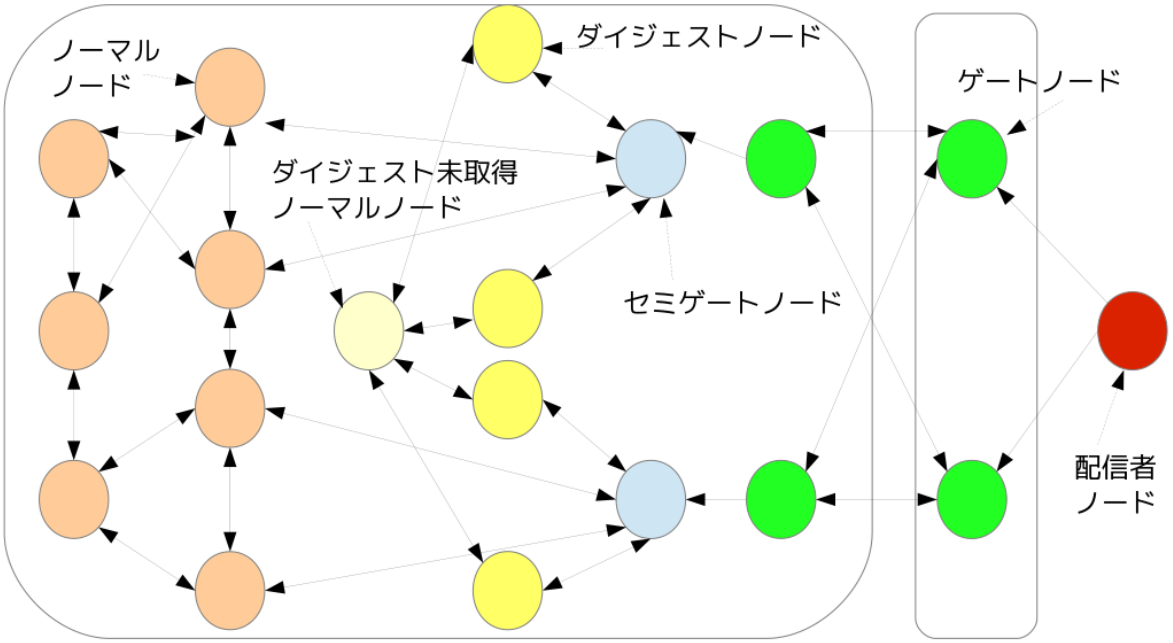

図 3.7　提案システムのトポロジ

3.2.5　新規参加ノードについて

新規ノードの参加手順について述べる．図 3.8 は新規ノードの参加手順を示したシーケンス図である．

新規参加ノードはネットワーク接続時に配信者ノードにつなぐ．その後, 配信者ノードはトラッカーサーバへ新規参加ノードの存在を伝える．トラッカーサーバは配信者ノードから受け取った新規参加ノードの情報を元に, 帯域や各クラスタへの論理ホップ数を計算し, 最も論理ホップ数の小さいクラスタに配置される．新規参加ノードはクラスタに配置された後, クラスタ内のダイジェストノードからダイジェストを受け取り, ダイジェストを視聴する．

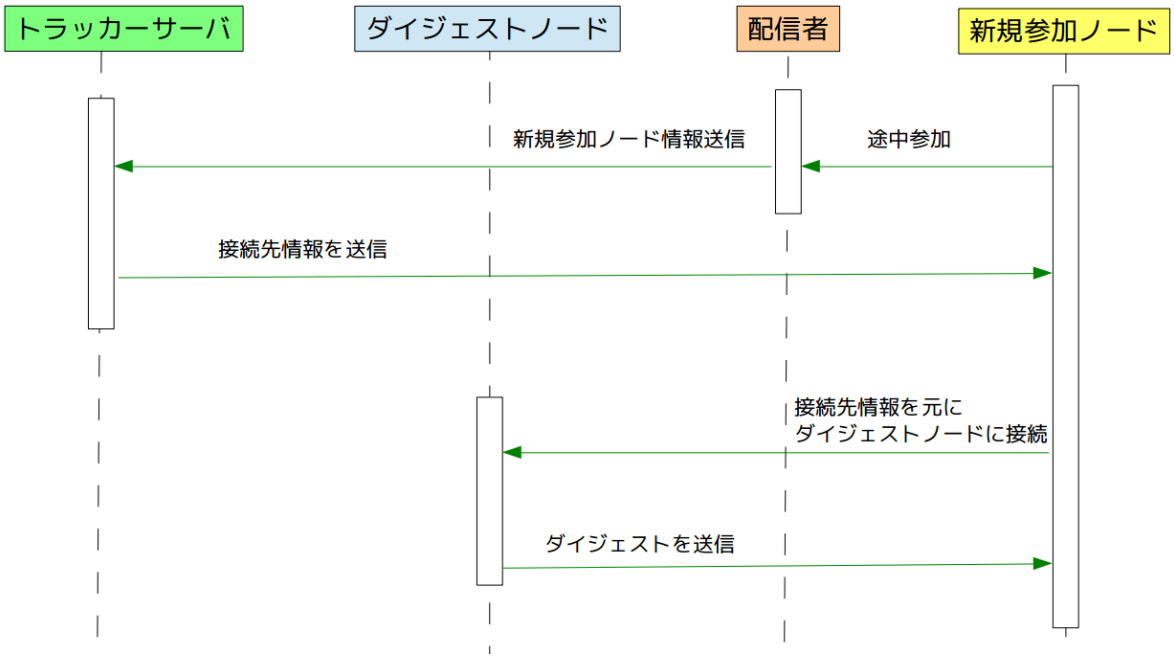

図 3.8　新規ノードの参加手順シーケンス図

3.2.6　再構築のタイミング

再構築のタイミングは 3 つ存在する．1 つ目は新規参加ノードがダイジェストを取得し終わった時である．ダイジェストノードとの接続を切断したあと, 帯域に応じて接続数を決定する．帯域が十分に高い場合にはセミゲートノードになる．2 つ目はノーマルノードのコメント数が一定数を超えた時である．この場合, 対象のノードはコメント数が多く, 離脱可能性が低いノードであると判断される．そしてその対象のノードはダイジェストノードになり, ダイジェストを保有し, 管理する役割を担う．3 つ目はノードが離脱した時である．この場合には, 離脱ノードに接続していたノードは接続数を維持するように他のノードと再接続する．特にゲートノードが離脱した場合にはセミゲートノードが即座にゲートノードになる．また, 特にセミゲートノードが離脱した場合には, クラスタの中で最も帯域が高いノードがセミゲートノードになる．

3.2.7 既存のトポロジ設計との比較

関連研究であげた既存のトポロジ設計との比較の結果を表 3.3 に示す. 比較する全てのトポロジは複数クラスタ型である. Yang Guo の提案した HCPS を「HCPS」, 元橋らの提案した重畳クラスタ木方式の動画配信システムを「重畳型」, Huey-Ing Liu らの提案した MeTree を「MeTree」, 本研究で提案する方式を「提案方式」とする.

表 3.3 既存のトポロジ設計との比較

	HCPS	重畳	MeTree	提案
ダイジェスト	なし	なし	なし	あり
クラスタ	完全結合	一部メッシュ	一部メッシュ	一部メッシュ
ゲートノード	高帯域ノード	滞在時間の長いノード	高帯域ノード	高帯域 + コメント数の多いノード

提案システムは, P2P のライブストリーミングにおいてダイジェストノードという役割を配置するなど, 生成したダイジェストを保持出来る構造になっている. また, クラスタの中身は一定の接続数を満たしながら遅延の少ないメッシュ構造になっている. さらにゲートノードの選出方法として高帯域のノードを選ぶことはもちろんだが, さらに離脱可能性を考えてコメント数の多いノードを選出している.

第 4 章

評価

本研究では, 配信に途中参加したユーザがダイジェスト視聴可能な P2P ライブストリーミングシステムを提案してきた. そのために 2 つの段階を考え, 1 つ目の段階では, P2P ネットワーク内で特別なサーバに頼らない, 汎用的なダイジェスト生成方式を提案した. 2 つ目の段階では, 1 つ目の段階において生成されたダイジェストを P2P ネットワーク内で保持し, それを拡散させるためのトポロジ設計を提案した.

この章では, まず 1 段階目のダイジェスト生成方式に対する評価を行い, その次に 2 段階目のトポロジ設計に対する評価を行う.

ダイジェスト生成方式では, まず 3 つの提案方式それぞれに設定されている閾値の適切な値を決めるための評価を行う. その次に実際の動画コンテンツを対象に提案システムを適用し, ダイジェスト生成に最も適切な方式を決定する. トポロジ設計では, まず各役割ノードの適切な割合を決めるための評価を行う. その次にその役割ノードが適切な役目を果たしているかの評価を行う. さらに生成されたダイジェストが適切に P2P ネットワーク内で保持され, 視聴出来たかの評価を行う. なお, 評価はシミュレーションで行う.

表 4.1 に評価実験を行った環境を示す.

表 4.1　実行環境

環境名	規格	バージョン
OS	Ubuntu	12.04 64-bit
CPU	Intel Core i7 2.10 GHz	
メモリ	8 GB	
シミュレータ	NS2	2.35
言語	OTcl	1.14
	TK	8.5.10
可視化ツール	nam	1.15

4.1 ダイジェスト生成方式に対する評価

4.1.1 評価の準備

ダイジェスト生成方式に対する評価を行う際の準備について述べる．評価の 1 つ目は, 3 つの提案方式の適切な閾値を決めることである．2 つ目は 3 つの提案方式のうち最も適切な方式を決定することである．

2 つの評価に共通するものとして, 対象となる映像がある．まず配信時間の長さの異なる 3 つ映像に対して, 時間の 10 分の 1 となる 1 つ 2 分間の正解シーンを用意した．つまり, 例えば 100 分の映像を扱う際には 5 個の正解シーンが存在するということである．

4.1.2 適切な閾値の決定

3 つの提案方式の適切な閾値を決定するための評価方法について述べる．まず, 提案システムがダイジェストであると選出したシーンの数が, 元々用意しておいた正解シーンよりも半分以上であるものを集計する．なお, 正解シーンの前後 2 分間をダイジェストと判断した場合を選出出来たと判断して集計する．集計した後, 最も多く正解シーンを選出した閾値を適切な値とする．例えば 100 分の映像ならば, 正解シーンが 5 つ存在する．そのうち提案システムが半分以上選出出来たらならばプラス 1 カウントする．これを 3 つの映像それぞれについて行い, 最も多く選出出来た（映像は 3 種類なので最大で 3）閾値を適切な値とする．

増減率に基づくダイジェスト生成方式の閾値

「増減率に基づくダイジェスト生成方式」では式 3.2 において閾値は, 増加率（Th_{inc}）, 減少率（Th_{dec}）で, パラメータは一定期間（T）であった．図 4.1 は増減率に基づくダイジェスト生成方式の抽出例である．時刻に付いている赤い丸が正解シーンであり, 縦に緑の線がシステムがダイジェストであると判断して選出した箇所である．図を見ると A と記した 1 箇所で正解していることがわかる．

表 4.2 に増加率 0.05 の時の正解シーン選出の結果を, 表 4.3 に増加率 0.10 の時の正解シーン選出の結果を示す．実験の結果, 増加率が 0.1, 減少率が 0.03, 一定期間が 3 である時が一番正解数が多く, 適切な値であることがわかった．

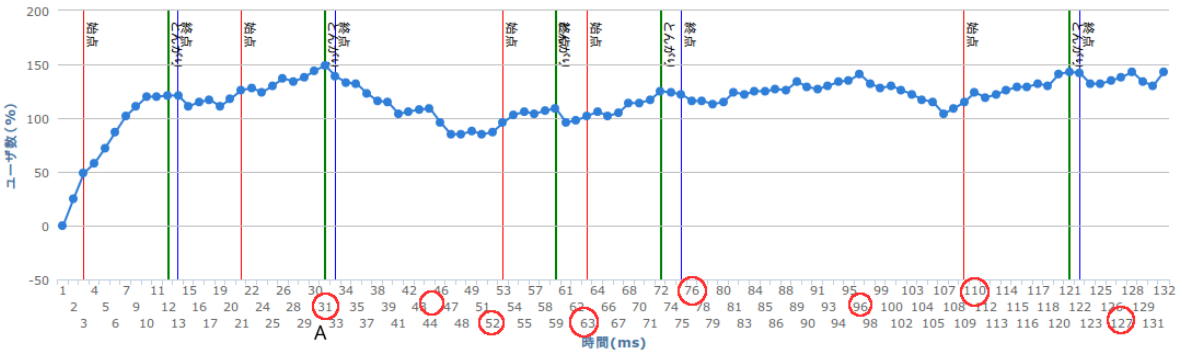

図 4.1　増減率に基づくダイジェスト生成方式の抽出例

表 4.2　増加率 0.05 の時の正解シーン選出の結果

増加率（Th_{inc}）0.05				
		減少率（Th_{dec}）		
		0.03	0.05	0.10
一定期間（T）	1	0	1	1
	2	0	0	2
	3	2	2	1
	4	1	1	1
	5	2	2	1
	6	2	2	1
	7	1	1	1

表 4.3　増加率 0.10 の時の正解シーン選出の結果

増加率（Th_{inc}）0.10				
		減少率（Th_{dec}）		
		0.03	0.05	0.10
一定期間（T）	1	1	1	0
	2	2	2	1
	3	3	2	1
	4	2	2	1
	5	2	2	1
	6	2	2	1
	7	1	1	1

前後比較に基づくダイジェスト生成方式の閾値

「前後比較に基づくダイジェスト生成方式」では式 3.3 において閾値は, 増加率（Th_{inc}）, 減少率（Th_{dec}）で, パラメータは一定期間（T）であった. 図 4.2 は前後比較に基づくダイジェスト生成方式の抽出例である. 時刻に付いている赤い丸が正解シーンであり, 縦に緑の線がシステムがダイジェストであると判断して選出した箇所である. 図を見ると A と記した 1 箇所で正解していることがわかる.

表 4.4 に増加率 0.05 の時の正解シーン選出の結果を, 表 4.5 に増加率 0.10 の時の正解シーン選出の結果を示す. 実験の結果, 増加率が 0.1, 減少率が 0.03, 一定期間が 4 である時が一番正解数が多く, 適切な値であることがわかった.

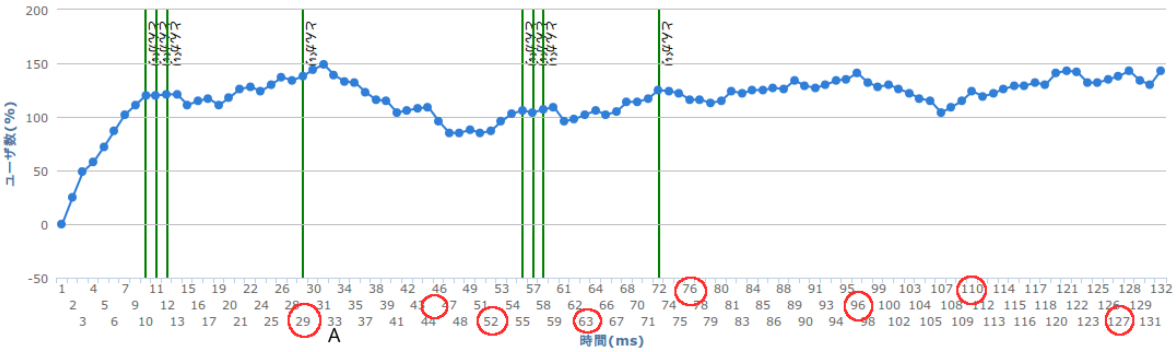

図 4.2　前後比較に基づくダイジェスト生成方式の抽出例

表 4.4　増加率 0.05 の時の正解シーン選出の結果

増加率（Th_{inc}）0.05				
		減少率（Th_{dec}）		
		0.03	0.05	0.10
一定期間（T）	1	0	0	0
	2	1	1	1
	3	0	1	0
	4	0	0	1
	5	0	0	0
	6	0	0	0
	7	0	0	1

表 4.5　増加率 0.10 の時の正解シーン選出の結果

増加率（Th_{inc}）0.10				
		減少率（Th_{dec}）		
		0.03	0.05	0.10
一定期間（T）	1	0	0	0
	2	1	1	0
	3	1	1	0
	4	2	1	0
	5	0	1	1
	6	0	0	0
	7	0	0	1

最小二乗法に基づくダイジェスト生成方式

「最小二乗法に基づくダイジェスト生成方式」では式 3.4 においてパラメータは, 角度（度）, データ数（個）であった. 図 4.3 は最小二乗法に基づくダイジェスト生成方式の抽出例である. 時刻に付いている赤い丸が正解シーンであり, 縦に緑の線がシステムがダイジェストであると判断して選出した箇所である. 図を見ると A, B, C, D, E と記した 5 箇所で正解していることがわかる.

表 4.6 に正解シーン選出の結果を示す. 実験の結果, 角度（度）55, データ数（個）3 である時が一番正解数が多く, 適切な値であることがわかった.

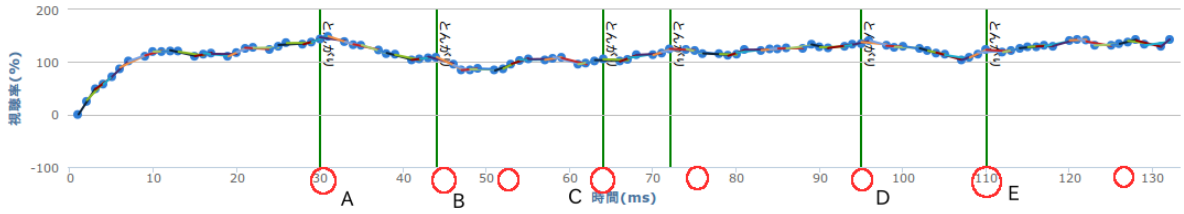

図 4.3　最小二乗法に基づくダイジェスト生成方式の抽出例

表 4.6　正解シーン選出の結果

		角度（度）											
		25	30	35	40	45	50	55	60	65	70	75	80
データ数（個）	2	0	0	0	2	3	3	2	1	1	0	0	0
	1	1	1	1	1	1	3	3	2	2	2	2	2
	0	0	0	0	0	0	0	2	2	2	2	2	2
	0	0	0	0	0	0	1	1	1	1	1	1	1

4.1.3 提案方式の評価

今までの実験により, それぞれの提案方式の適切な閾値がわかった. 次に, その適切な閾値を適用したそれぞれの提案方式を比較し, P2P ライブストリーミングにおいてダイジェストを生成する際に最も適当なダイジェスト生成方式を決定する.

3 つの動画に対して, それぞれの提案方式の適合率を計算することによって評価する. 適合率は式 4.1 に従う. これは, 提案方式がダイジェストとして選出したもののうち, 元々用意しておいた正解シーンが含まれている確率を意味する. 適合率の値が高いほど, より適切なダイジェスト生成方式であると言える. 表 4.7 に実験の結果を示す. 実験の結果, 最小二乗法に基づくダイジェスト生成方式の適合率が最も高く, 平均で 66% を超えるという結果になった.

$$\text{適合率} = \frac{\text{正解シーン数}}{\text{方式による抽出シーン数}} \tag{4.1}$$

表 4.7 適合率の結果

	適合率		
生成方式	増減率に基づく方式	前後比較に基づく方式	最小二乗法に基づく方式
動画 1	12.5%	50.0%	62.5%
動画 2	37.5%	50.0%	75.0%
動画 3	50.0%	50.0%	62.5%

4.1.4 ダイジェスト生成方式に対する評価の考察

ダイジェスト生成方式の実験では, まずそれぞれの提案方式の適切な閾値を決定した. 次にその閾値を適用した提案方式の適合率を計算することによって最も適切なダイジェスト生成方式を決定した.

適切な閾値の決定では,「増減率に基づくダイジェスト生成方式」,「前後比較に基づくダイジェスト生成方式」,「最小二乗法に基づくダイジェスト生成方式」のそれぞれで閾値を決定した. それぞれについて考察していく. 次に適合率の結果についての考察を行う.

増減率に基づくダイジェスト生成方式に対する考察

実験的に求めた望ましい閾値は増加率 0.1, 減少率 0.03, 一定時間 3（ms）という結果であった. 増加率は減少率よりも高い値であった. これは, 瞬間的に多くのユーザがコメントをして, その瞬間に対して反応したことを意味している. 瞬間的に多くの反応があっ

たということは, とても注目度が高いことを意味しており, ダイジェストにすべき瞬間であったことがわかる. まだ, 一定時間は 3 という結果であり, 長くもなく短くもない値であった. これは, 短すぎると雑音となる部分まで余計に反応してしまうためであり, 逆に長すぎると瞬間的な変化に反応出来ないからであると考えられる.

前後比較に基づくダイジェスト生成方式に対する考察

実験的に求めた望ましい閾値は増加率 0.1, 減少率 0.03, 一定時間 4（ms）という結果であった. この閾値は「増減率に基づくダイジェスト生成方式」の結果とほぼ同じであり, 基本的には同様の考え方が出来る. しかし, こちらの方式では増加率が 0.05 の場合だとほとんど正解シーンを選出出来なかったことがわかる. これは「前後比較に基づくダイジェスト生成方式」の方では特に増加率の値に敏感であり, 細かい値を設定することが望ましいことがわかった.

最小二乗法に基づくダイジェスト生成方式に対する考察

実験的に求めた望ましい閾値は角度 55（度）, データ数 3（個）という結果であった. 全体的に角度は急なほど正解シーンを多く選出していることがわかる. これは, より瞬間的な状態を検出していることがわかる. データ数は少ない方が多く正解シーンを選出していることがわかる. これは, データ数が多いほどグラフをなだらかにしてしまい, 瞬間的な状態を検出出来なかったからではないかと考えられる.

適合率の結果に対する考察

適合率は「最小二乗法に基づくダイジェスト生成方式」が最も適切であることがわかった. これは, その他の方式では雑音となる突発的な瞬間を誤検知してしまったのに対し, こちらの方式では最小二乗法を適用していることで, より自然な盛り上がり部分をダイジェストとして捉え, 検出出来ているからではないかと考えられる.

ダイジェスト生成方式の要求条件に対する考察

ダイジェスト生成方式の要求条件に対する考察を行う. 各要求条件を示し, それについての考察を行う.

- 特定の種類のコンテンツに依存しないダイジェスト生成方式であること
  本研究では, 単位時間当たりにコメントしたユーザ数を利用したダイジェスト生成方式を提案し, これは特定のコンテンツに依存しない, 汎用的なものである.

- P2P ライブストリーミングに特化したダイジェスト生成方式であること
  既存手法では, ダイジェストを生成するための専用のサーバを利用する必要があったが, 本研究では各ノードがダイジェストを生成するため P2P に特化したものである. また, 上記と同様に参加ユーザが配信内容に対して自由にコメント投稿が出来るシステムにおいて, 単位時間当たりにコメントしたユーザ数を利用したことは, ライブストリーミングに特化したものである.
- 生成されたダイジェストの映像によってコンテンツの内容の全体把握が出来ること
  4.1.3 節の結果, 平均で 66% の適合率を記録した. 提案したダイジェスト生成方式によって, コンテンツの内容の全体を把握することが出来る.

4.2 トポロジ設計に対する評価

トポロジ設計に対する評価を行う．評価はシミュレーションで行った．シミュレーションには NS-2 を用いた．提案システムでは各ノードに役割を与えたトポロジを設計した．その各役割の適切な割合についての評価を行う．また, 役割ノードが適切に機能しているかの評価を行う．そして, 配信者ノードから各ノードへ送られるストリームの遅延に対する評価を行う．さらに, P2P ネットワーク内に十分なダイジェストを保有できているかを評価する．

4.2.1 前提条件

シミュレーションを行う上でいくつかの前提となる条件について述べる．前提となる項目に「帯域幅分布」,「コメント数分布」,「ノード数とクラスタ」がある．それぞれについての前提条件を決定する．

全ノードには固有の帯域幅を割り当てる．「帯域幅分布」は, 他の P2P ライブストリーミング研究 [12] で使用されている値を参考にする．この研究ではインターネットユーザの帯域幅分布を解析した 2 つの研究 [13][14] を参考にしている．帯域幅は全 10 種類あり, 平均は 540kbps である．表 4.8 のように決定する．

表 4.8　帯域幅分布

帯域幅 (kbps)	256	320	384	448	512	640	768	1024	1500	3000
割合 (%)	10.0	14.3	8.6	12.5	2.2	1.4	6.6	28.1	1.4	14.9

また, 全ノードには固有のコメント数を割り当てる．サンプルとして 1 つの動画に対して分析を行った結果を利用する [16]．1 つ 24 分で 2013 年 3 月 20 日時点での動画を参考にしている．1 人当たりの平均コメント数は 4 で標準偏差は 6.8 である．

図 4.4 は, 動画を分析した結果である．横軸が 1 人当たりのコメント数, 縦軸がそのコメント数をしている人の割合である．

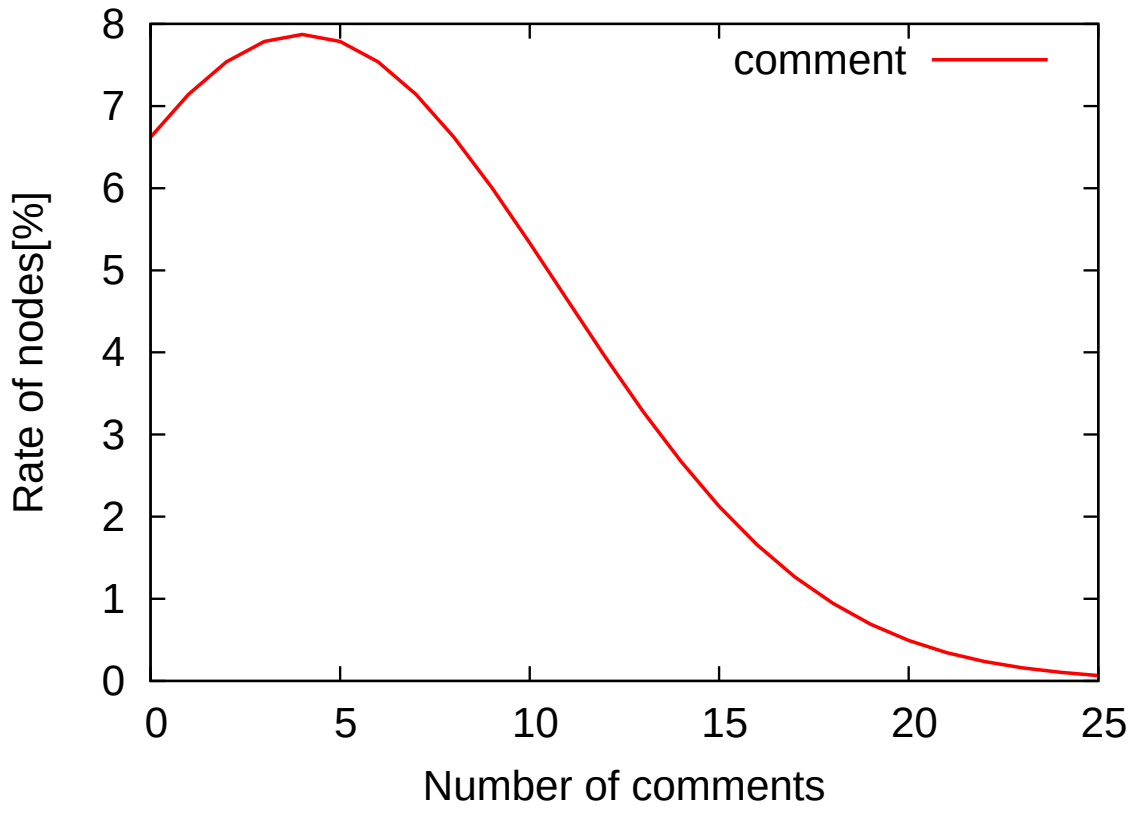

図 4.4　コメント数分析の結果

図 4.4 を 10 種類の具体的な数字に表すと表 4.9 が出来上がる．この表を元に, 各ノードに対して固有のコメント数を割り当てる．

表 4.9　コメント数分布

コメント数	2	5	7	10	12	15	17	20	22	25
割合 (%)	15	16	16	15	13	11	10	2	1	1

また, 全ノード数に応じたクラスタの数を予め設定しておく．設定する値は [15] の研究で示されている線形関数を利用する．図 4.5 に参加ノード数に応じたクラスタ数の変化を示す．このグラフを元に表 4.10 を作成した．こちらの表の通り全ノード数に対するクラスタ数を決定する．全ノード数は,「ゲートノード」,「セミゲートノード」,「ダイジェストノード」,「ノーマルノード」の 4 種類を対象とする．

表 4.10　ノード数とクラスタ数対応表

ノード数	200	400	600	800
クラスタ数	7	10	14	18

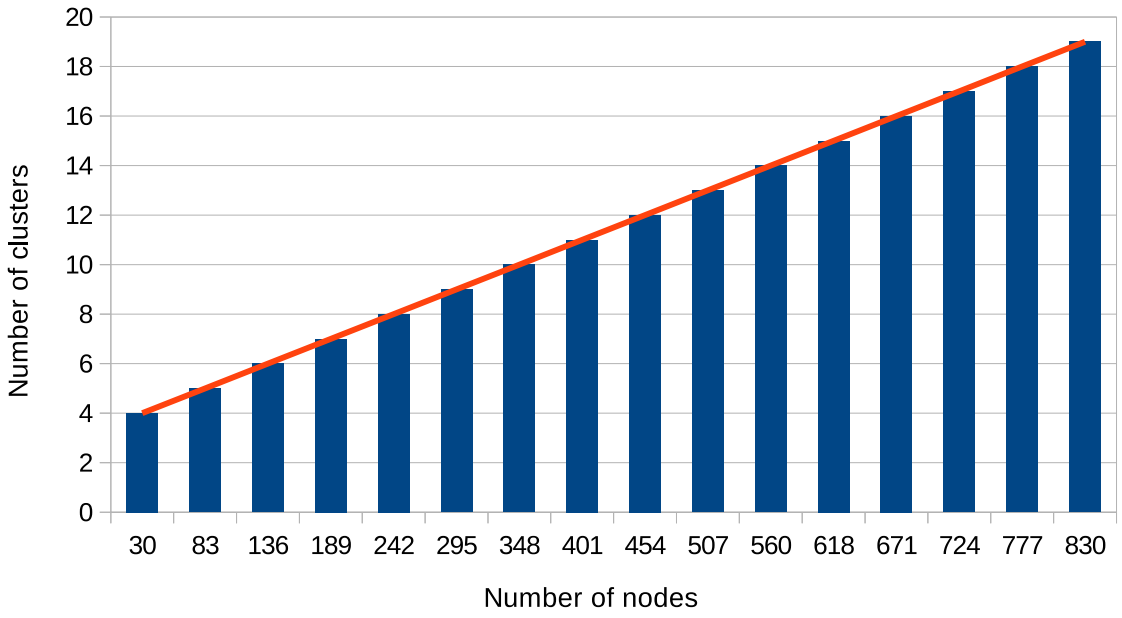

図 4.5　コメント数分析の結果

その他の前提条件として, ノード間遅延を全てのノード間で 100ms に設定している．またパケットロス率は 0% として考慮しないこととする．また, 参加離脱を考慮した実験においては, 10 秒に 1 回参加と離脱を行うこととする．参加はランダムであり, 最初はダイジェスト未取得のノーマルノードとなる．ダイジェスト未取得のノーマルノードは 10 秒経つとダイジェスト取得済みのダイジェストノードとなる．また, シミュレーションにおいての参加離脱はノードにパケットが流れたかどうかで判断する．

ノード数が 200 の時の、役割を与えた場合のトポロジ図を図 4.6 に示す．図ではゲートノードが全体の 10%, セミゲートノードが全体の 10%, ダイジェストノードが全体の 20% となるように構成されている．具体的な構成を表 4.11 に示す．

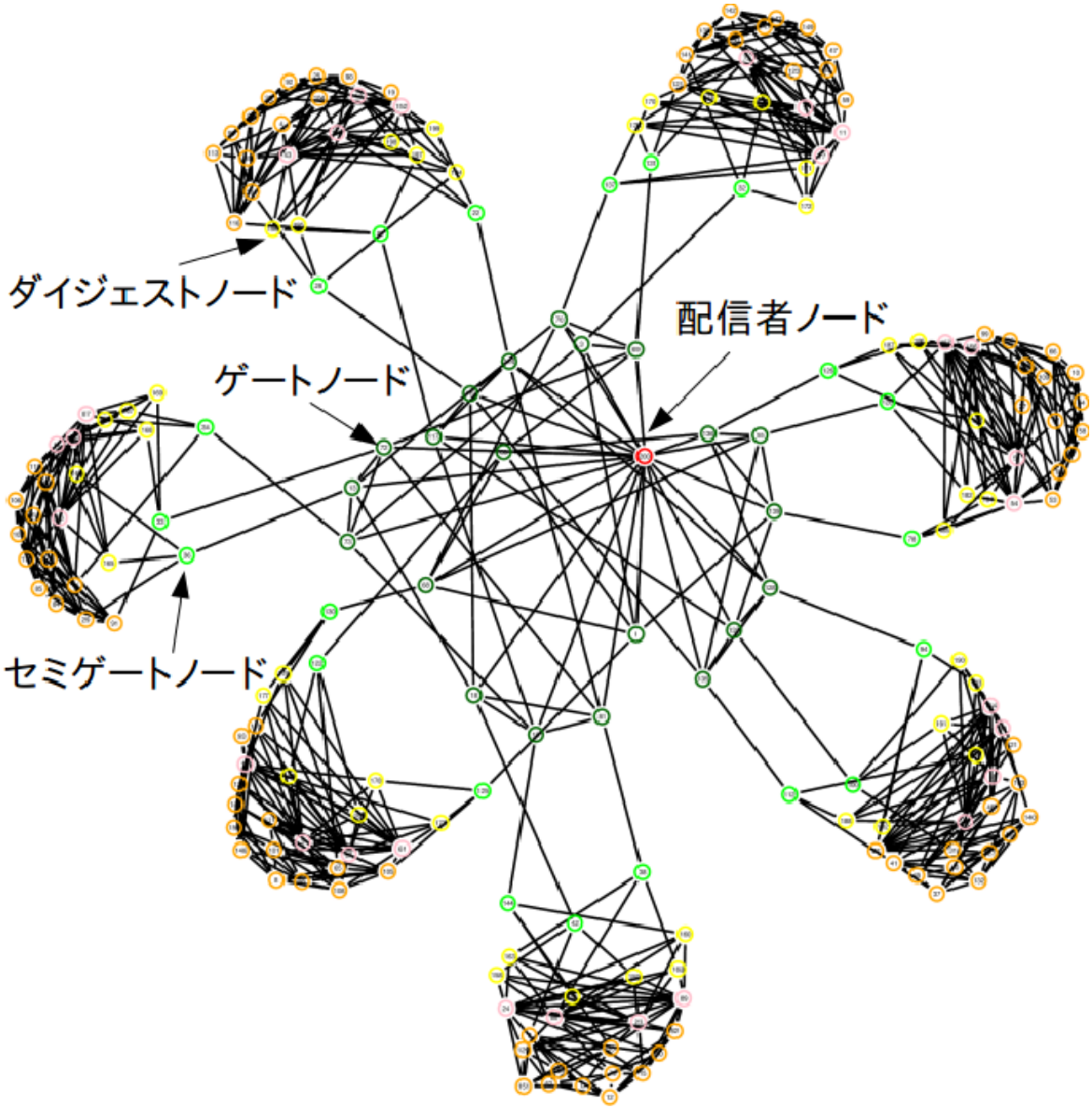

図 4.6　役割を与えた場合のトポロジ図

表 4.11　ノード数が 200 時の役割を与えた時の役割構成

役割	ノード数
配信者ノード	1
ゲートノード	21
セミゲートノード	21
ダイジェストノード	42
ノーマルノード	112

ノード数が 200 の時の, 役割を与えない場合のトポロジ図を図 4.7 に示す. 図では役割が無く, 全てがノーマルノードになっている. 具体的な構成を表 4.12 に示す.

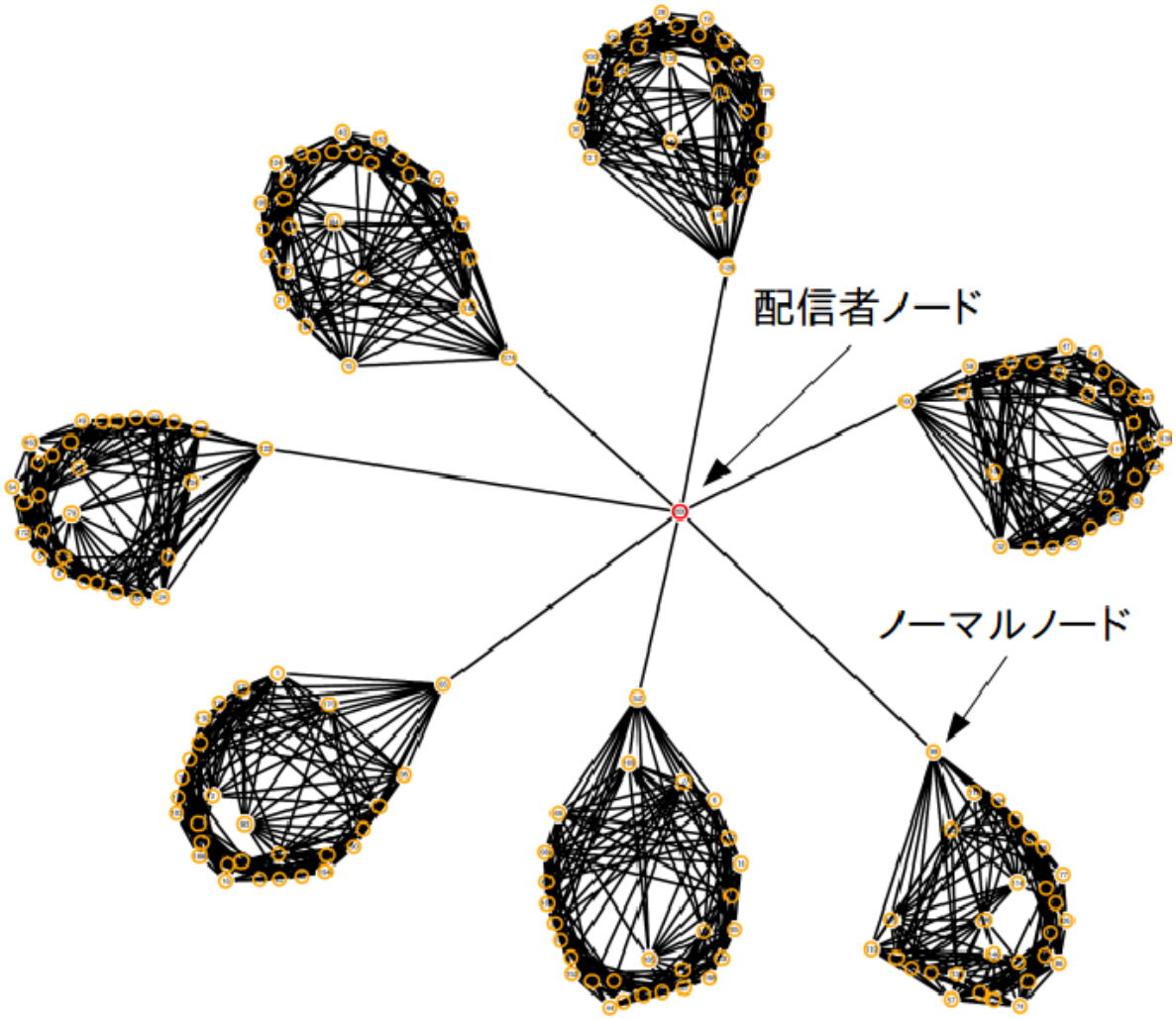

図 4.7　役割を与えない場合のトポロジ図

表 4.12　ノード数が 200 時の役割を与えない時の役割構成

役割	ノード数
配信者ノード	1
ノーマルノード	199

以上のトポロジ構成にしたがって, 役割を与えた場合と与えない場合での比較を行う.

4.2.2　適切な役割の割合の決定

提案するトポロジに存在するノードの役割のうち「ゲートノード」, 「セミゲートノード」, 「ダイジェストノード」の役割の適切な割合を決定する. まず最初に「ダイジェストノード」の割合を, 次に「ゲートノード」と「セミゲートノード」の割合を決定する.

ダイジェストノードの割合

ダイジェストノードの割合をそれぞれ, 全ノードの 10%, 20%, 30% で割合を変え, スループットの値を比較した. スループットはネットワーク全体で単位時間当たりに受信したパケット数と定義する.

図 4.8 に結果を示す.

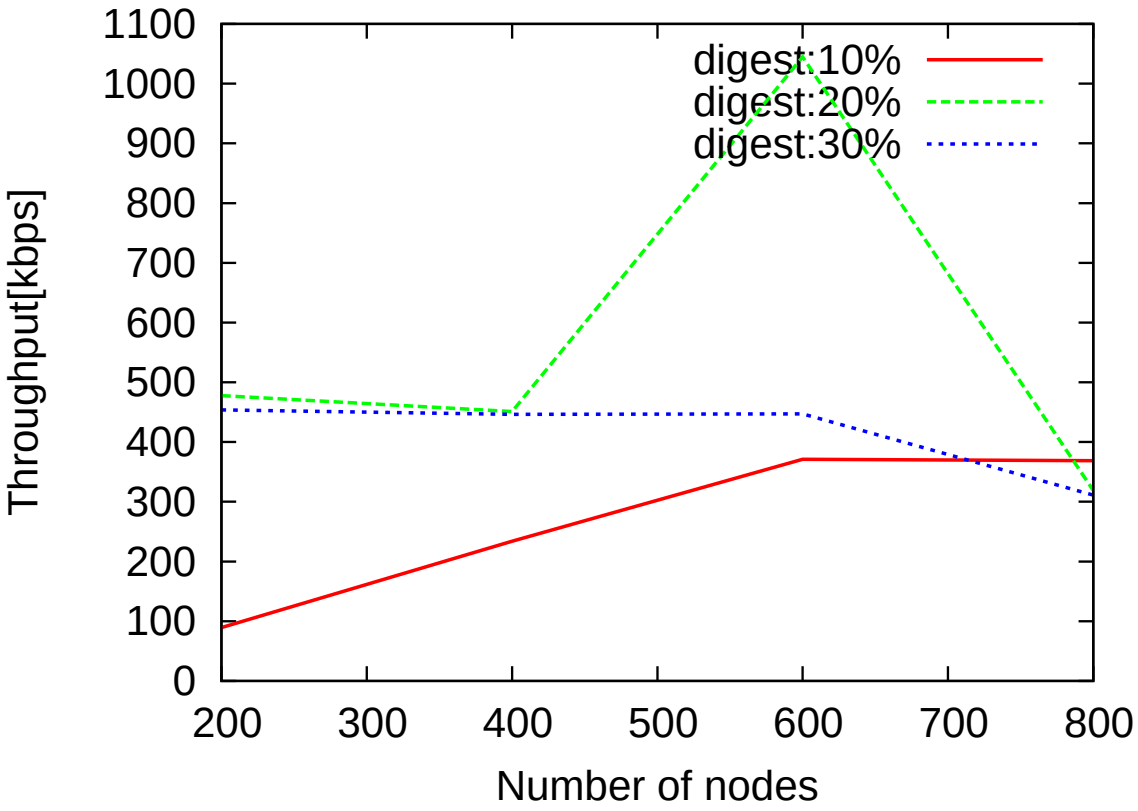

図 4.8　ダイジェストノードの割合を変化させた時の結果

実験の結果, 10% の時の平均スループットが 265.71kbps, 20% の時の平均スループットが 573.12kbps, 30% の時の平均スループットが 414.63kbps であった. 20% の時が最もスループットの値が大きく, 適切な値であることがわかった.

ゲートノードとセミゲートノードの割合

次にゲートノードとセミゲートノードの割合を決定する. ゲートノードとセミゲートノードは常に 1 対 1 の関係となるため, 一緒に決定する. 全ノードのそれぞれ 10% と 10%, 20% と 20% で割合を変え, スループットの値を比較した. 図 4.9 に結果を示す.

実験の結果, 10% と 10% の時の平均スループットが 446.33kbps, 20% と 20% の時の平均スループットが 413.48kbps であった. どちらもそれほど変わらないが, 10% と 10% の方がスループットの値が大きかった. しかし, グラフを見るとノード数が小さい時は 10% と 10% の時の方が値が高く, ノード数が大きい時は 20% と 20% の時の方が値が高かった.

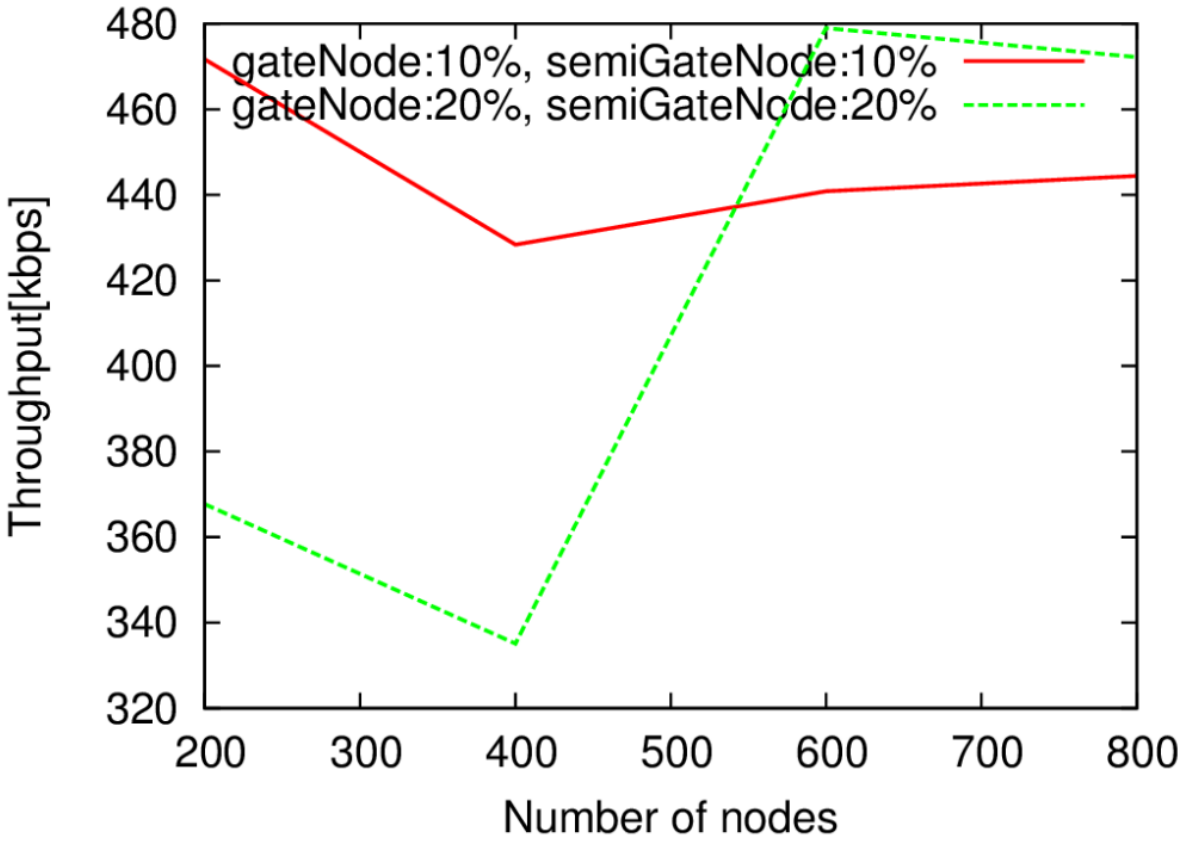

図 4.9　ゲートノードとセミゲートノードの割合を変化させた時の結果

4.2.3　役割の適切性に対する評価

次に役割を与えたことの適切性に対する評価を行う．図 4.10 に役割を与えた場合のネットワーク全体のスループットの推移の様子を示す．udp200 はノード数が 200 の時を意味する．図を見ると，ノード数が 200 の時が一番低い値を取っているが，それ以降ノード数が大きくなってもスループットの値が低下していないことがわかる．

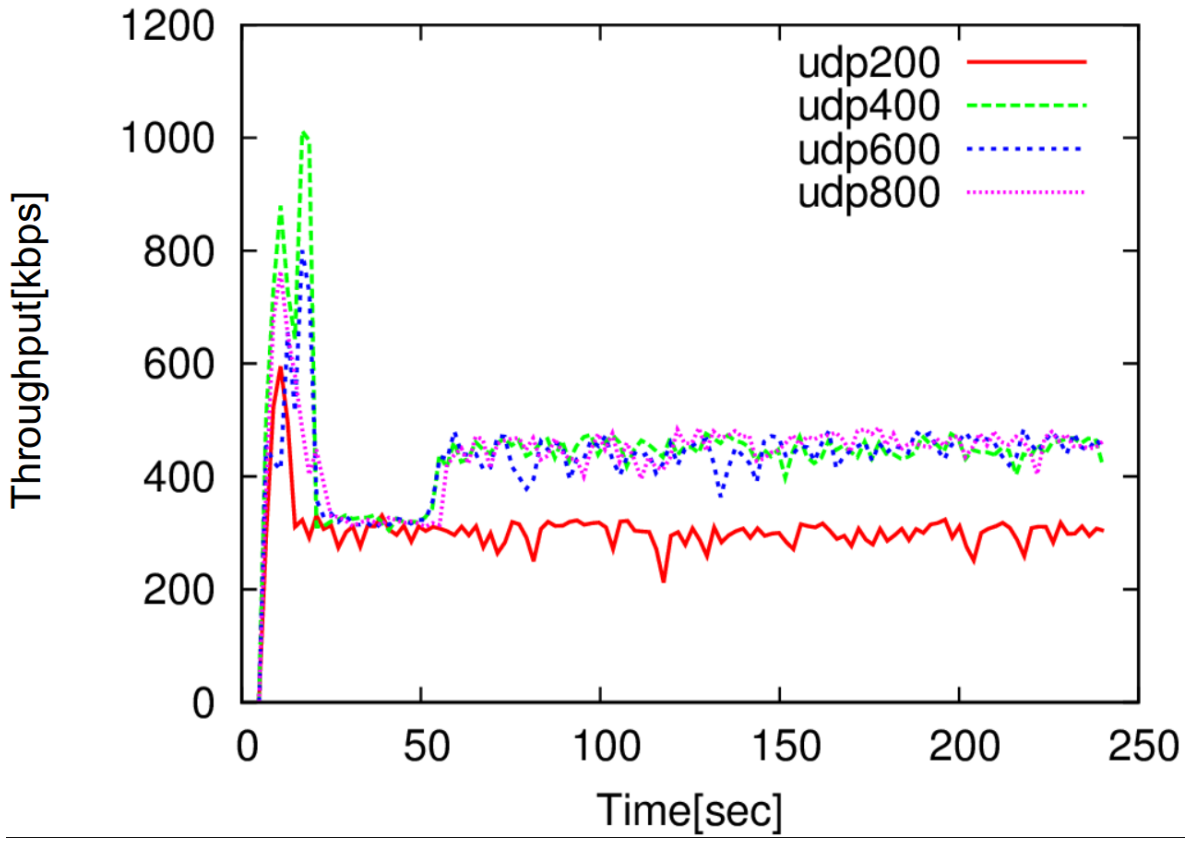

図 4.10　各ノード数におけるスループットの推移

図 4.11 に役割を与えない場合のスループットの推移の様子を示す．udp200-no-role はノード数が 200 の時を意味する．図を見ると，ノード数が 200 の時が一番高い値を取っているが，それ以降ノード数が大きくなるに連れてスループットの値が低下していることが分かる．

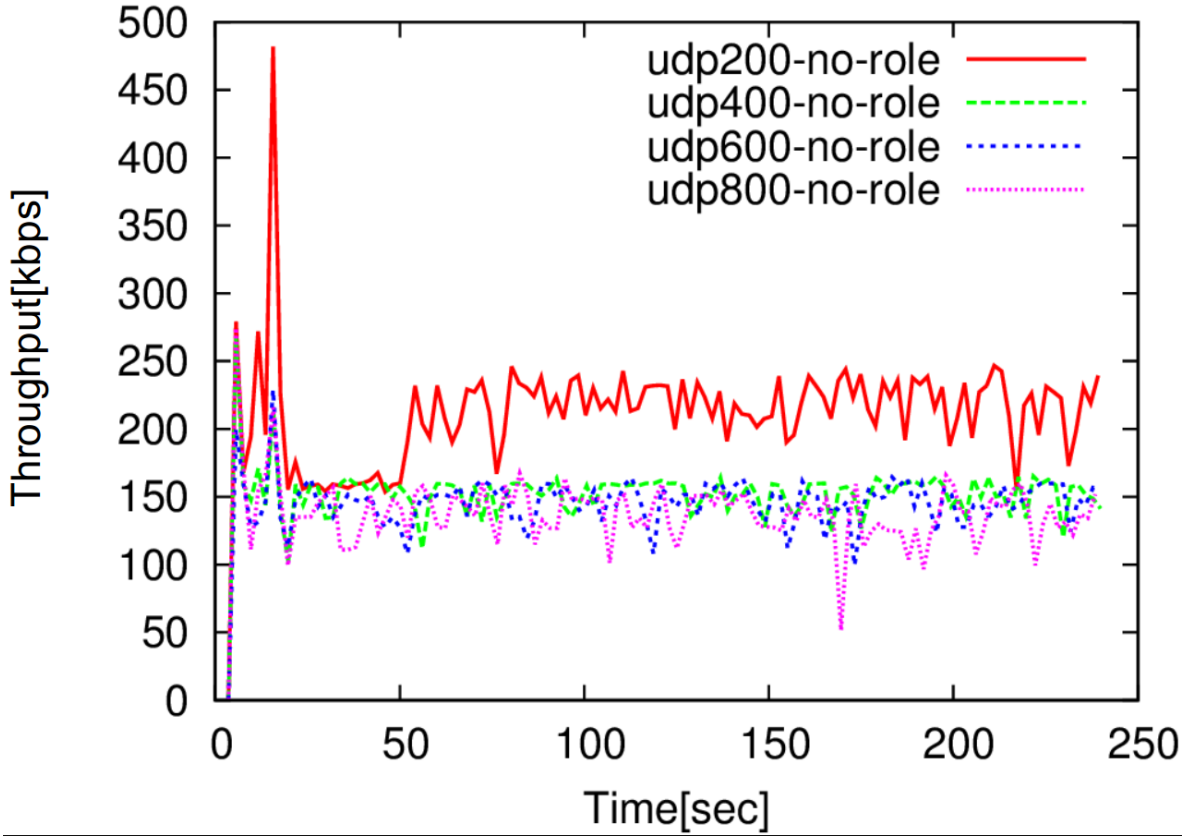

図 4.11　役割を与えない場合の各ノード数におけるスループットの推移

図 4.12 に役割を与えた場合と与えない場合の各ノード数におけるスループットの推移の平均の比較を示す．average が役割を与えた場合であり，average-no-role が役割を与えない場合である．図を見ると，役割が無い場合はノード数が大きくなるとスループットの値が低下しているのに対し，役割がある場合はノード数が大きくなってもスループットの値が低下していないことがわかる．また，役割を与えない場合の平均のスループットの値は 161.48kbps であり，役割を与えた場合の平均のスループットの値は 406.73kbps であった．

例えば，画面サイズ 678 × 432，フレームレート 19fps で H.264 エンコードでストリーミング配信を行った場合，目安となるビットレートは 390kbps となる．役割を与えた場合の平均のスループットの値は 406.73kbps なので，この値以上の性能が出ることがわかる．この結果，役割を与えたことによって P2P ネットワーク全体で質の高い映像を見ることが可能で，役割を与えたことの有用性を確認することが出来た．

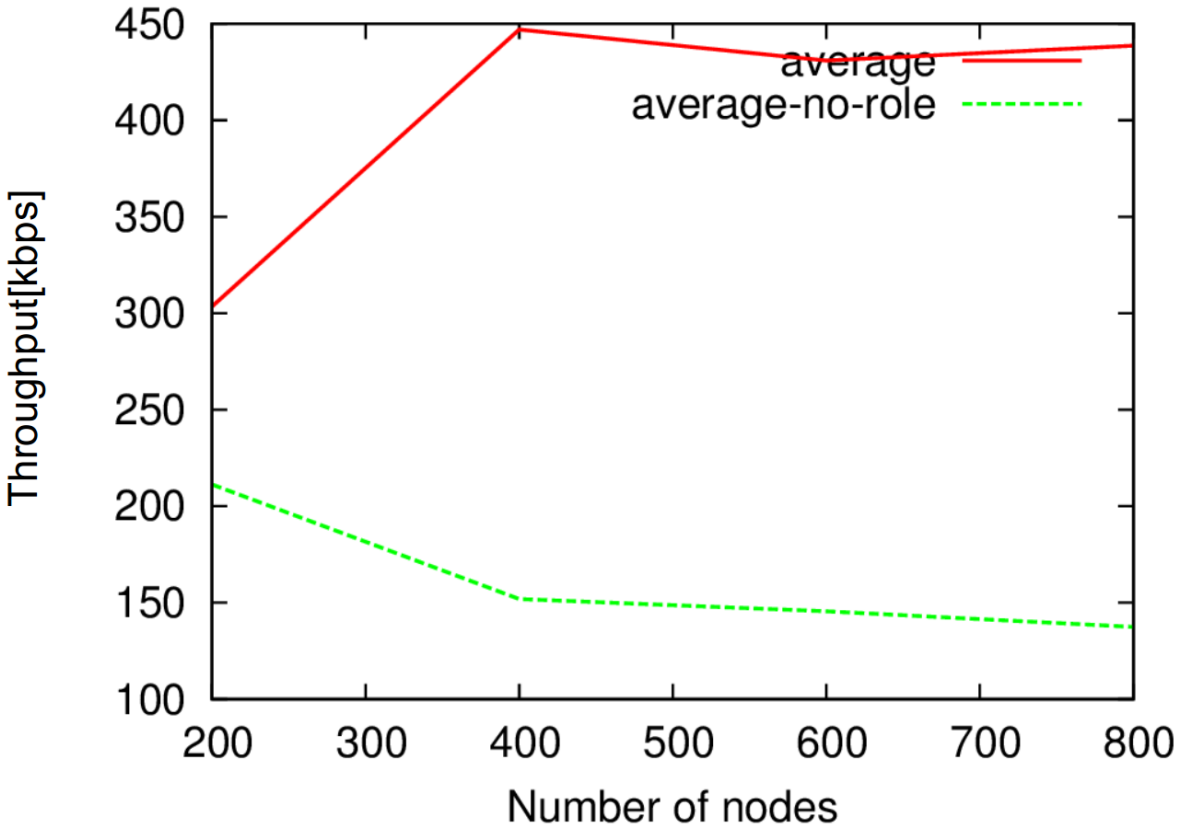

図 4.12　役割を与えない場合の各ノード数におけるスループットの推移の平均の比較

4.2.4　遅延に対する評価

遅延に対する評価を行う．遅延に対する評価として 2 つの要因を考えた．1 つ目は配信者ノードから各ノードへのホップ数であり, 2 つ目は各ノードの接続数である．まずはホップ数に対しての評価を行い, 次に接続数に対しての評価を行う．

表 4.13 は役割がある場合の各ノード数におけるネットワーク全体の平均ホップ数であり, 表 4.14 は役割が無い場合の各ノード数におけるネットワーク全体の平均ホップ数である．役割がある場合と無い場合の平均ホップ数をグラフ化したものを図 4.13 に示す．

表 4.13　役割がある場合の各ノード数におけるネットワーク全体の平均ホップ数

ノード数	合計ホップ数	平均ホップ数
200	645	3.23
400	1288	3.22
600	1864	3.11
800	2514	3.14

表 4.14　役割が無い場合の各ノード数におけるネットワーク全体の平均ホップ数

ノード数	合計ホップ数	平均ホップ数
200	488	2.44
400	980	2.45
600	1464	2.44
800	2040	2.45

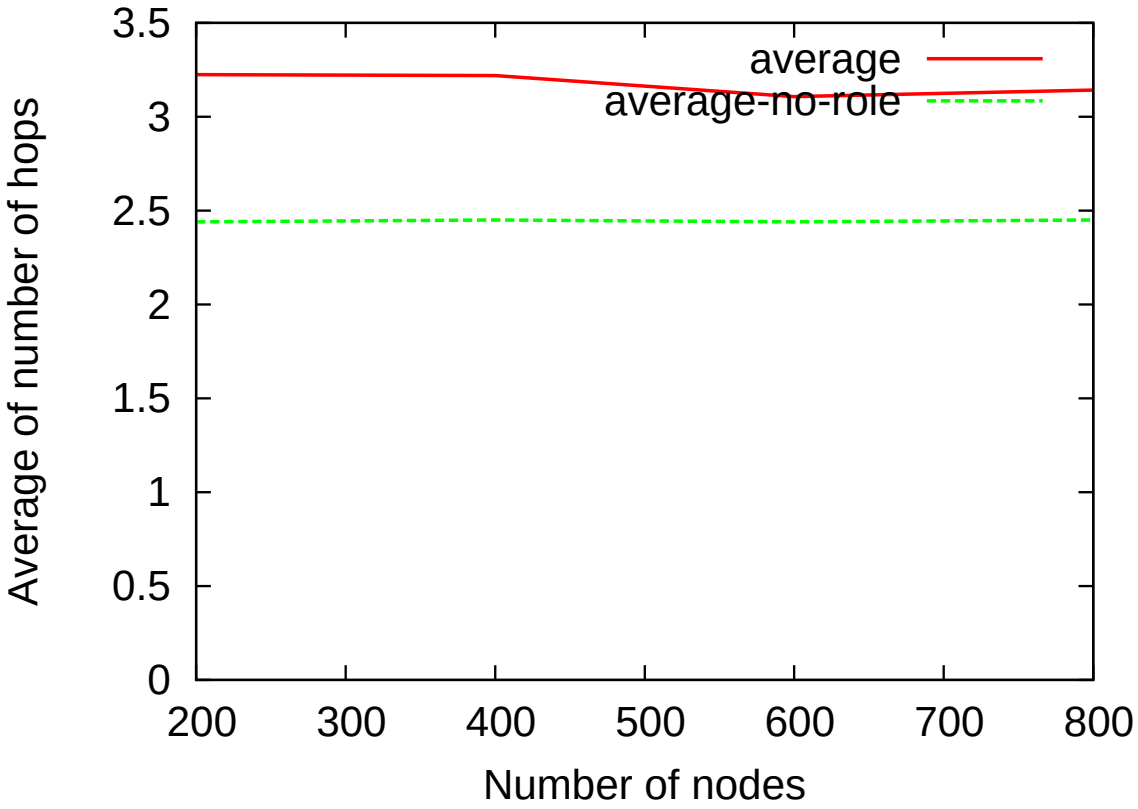

図 4.13　各ノード数におけるネットワーク全体の平均ホップ数グラフ

実験の結果, ノード数を変化させた時の平均のホップ数は役割がある場合と無い場合でそれぞれ 3.18 と 2.45 である. この結果, 役割が無い場合が役割がある場合に比べてホップ数が少なく, 優れていることがわかった.

表 4.15 は役割がある場合の各接続数におけるノード数を示しており, 表 4.16 は役割が無い場合の各接続数におけるノード数を示している.

表 4.15　役割がある場合の各接続数におけるノード数

		接続数									
		3	4	6	7	14	15	16	23	24	25
ノード数	200	0	25	21	0	12	12	2	0	0	0
	400	0	40	120	0	1	0	0	0	0	0
	600	14	56	108	0	9	1	0	10	0	0
	800	18	72	252	8	273	76	11	47	38	5

表 4.16　役割が無い場合の各接続数におけるノード数

		接続数		
		11	22	23
ノード数	200	0	0	0
	400	400	10	0
	600	12	442	146
	800	8	686	106

また, それぞれの表をグラフにしたものを図 4.14 と図 4.15 に示す.

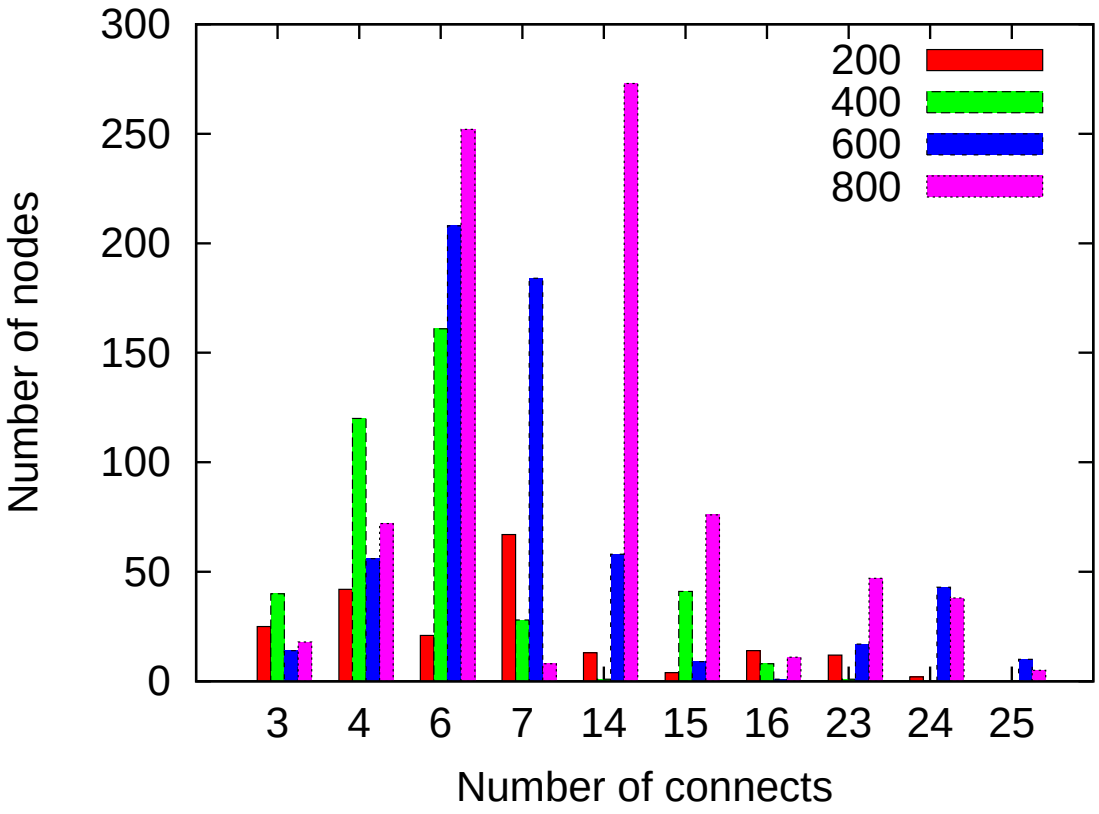

図 4.14　役割がある場合の各接続数におけるノード数グラフ

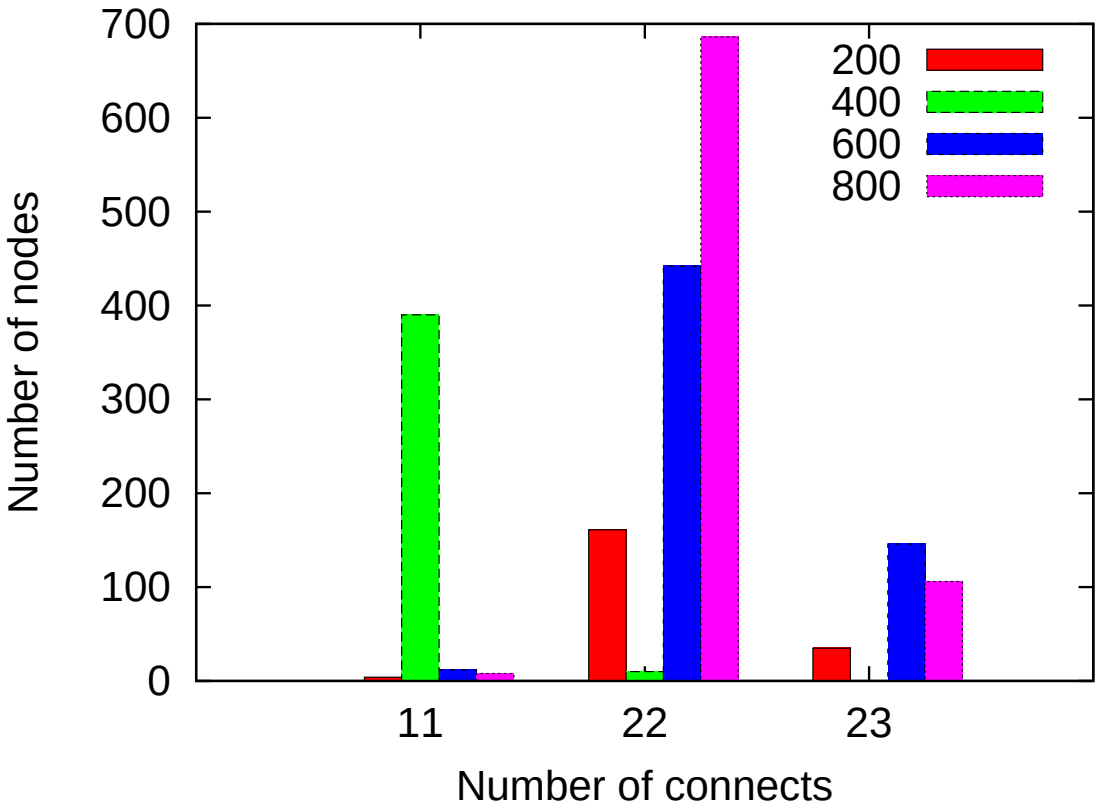

図 4.15　役割が無い場合の各接続数におけるノード数グラフ

また, 役割がある場合と無い場合の各接続数におけるノード数の平均の比較を図 4.16 に示す.

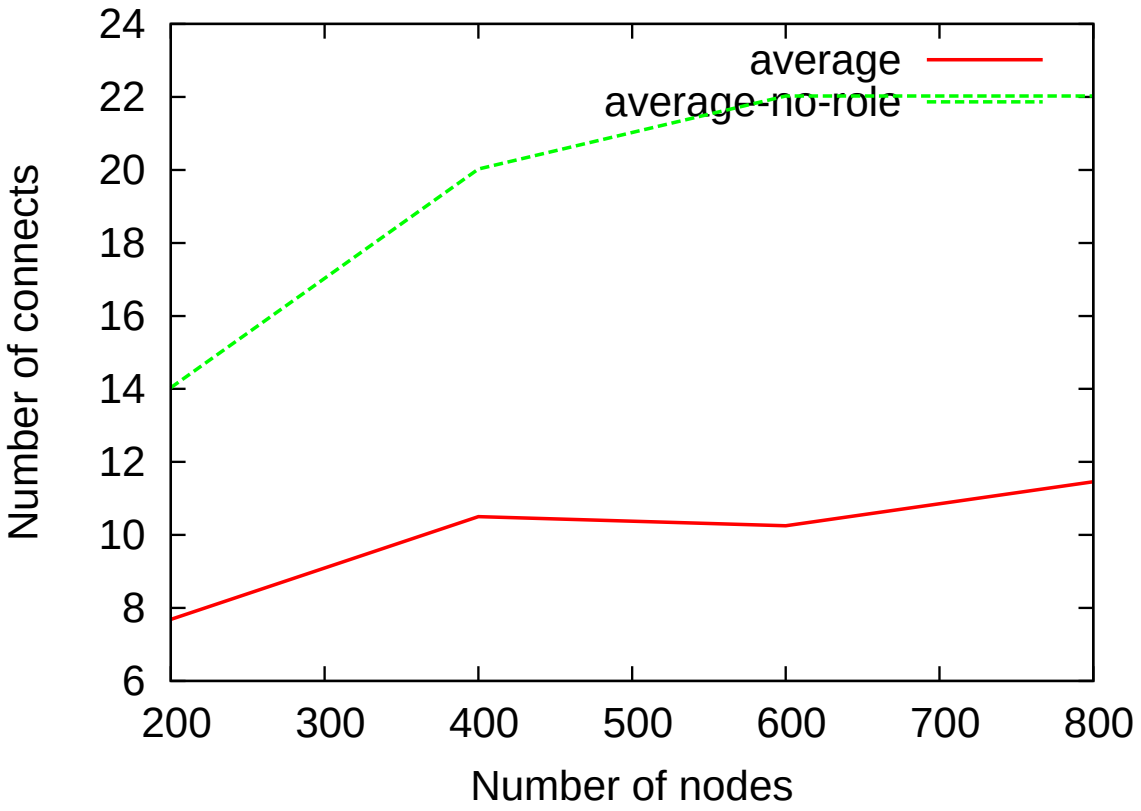

図 4.16　役割が無い場合の各接続数におけるノード数グラフ

実験の結果, 役割がある場合は接続数が 6 から 14 の時に集中しており, 役割が無い場合は 22 から 23 に集中していることがわかった. また, 各ノード数における平均の接続数では, 役割がある場合が役割が無い場合に比べて全体的に接続数が少なく, 優れていることがわかった.

4.2.5　ダイジェスト保有率に対する評価

ダイジェスト保有率に対する評価を行う. ダイジェスト保有率に対する評価では, ノードが参加と離脱を行う場合を想定してシミュレーションを行う. このシミュレーションを行うにあたって与えた条件を次に示す. まず, これまではノード数に対するクラスタ数を表 4.10 のように示してきたが, このシミュレーションでは 1 クラスタのみを対象として行う. 具体的にはノード数 28 に対してクラスタ数 1 で行う. また, NS-2 ではノードを新たに追加したり, 既存のノードを離脱させることが厳密には出来ない. そこで, このシミュレーションではノードに対してパケットを流すことでネットワークに参加していることを表し, パケットを流さないことでネットワークから離脱していることを表す. 表 4.17 にダイジェスト保有率に対する評価の構成を示す.

表 4.17　ダイジェスト保有率に対する評価の構成

ノード数	28
クラスタ数	1
役割とノード数	配信者ノード: 1
	ゲートノード: 3
	セミゲートノード: 3
	ダイジェストノード: 6
	ノーマルノード: 16

1 つのダイジェストは 1 つのダイジェストノードが保有しており, ダイジェストノードが離脱することでそのノードが保有していたダイジェストが失われることになる. ダイジェスト保有率は以下の式 4.2 に従う.

$$\text{ダイジェスト保有率} = \frac{\text{ダイジェストノード数} - \text{離脱したダイジェストノード数}}{\text{ダイジェストノード数}} \tag{4.2}$$

また, 各ノードの離脱率はコメント数に比例し, 表 4.9 に従う.

シミュレーションを 10 回行った結果を表 4.18 に示す.

表 4.18　ダイジェスト保有率に対する評価の結果

回数	1	2	3	4	5	6	7	8	9	10
ダイジェスト保有率（%）	66.7	83.3	83.3	66.7	66.7	83.3	88.3	88.3	66.7	83.3

実験の結果, 全ての回数で 66.7% 以上の保有率があり, 平均で 76% の保有率を達成した. 配信内容全体の 76% を把握出来るということであり, 配信内容の概要を知るのに十分であると考えられる.

4.2.6　トポロジ設計に対する評価の考察

トポロジ設計に対する実験では, まず「ダイジェストノード」, 「ゲートノード」, 「セミゲートノード」の 3 つの役割のそれぞれで適切な割合を決定した. 次に役割を与えたことの適切性についての評価を行った. そして配信者ノードからの遅延に対する評価を行った. さらにダイジェスト保有率に対する評価を行った. それぞれについての考察を行う.

ダイジェストノードの割合の決定に対する考察

ダイジェストノードの割合は 20% が良いという結果だった. 図 4.8 を見ると, 20% の時はノード数が 600 の時にとても高い値を記録しているが, それ以外は 30% の時とほぼ同

じ値を記録していることがわかる．20% の時は突発的に大きな値を取っていた可能性が高く, 実際は状況に応じて 20% から 30% の割合にするのが良いと考えた．

ゲートノードとセミゲートノードの割合の決定に対する考察

ゲートノードとセミゲートノードの割合はノード数によって割合を変えることが良いという結果であった．ノード数が小さい時は 10% と 10% に, ノード数が大きい時は 20% と 20% が良いということだった．ノード数が小さい時にそれぞれの割合が小さい方が良い理由として, 1 ノード当たりの接続数が少ないことが原因であると考えられる．ノード数が小さいと接続数が少ないので, 1 ノード当たりの負担が小さく, 小さい割合でも問題ないのではないかと考えられる．ノード数が大きい時にそれぞれの割合が大きい方が良い理由も同様である．1 ノード当たりの負担が大きくなり, 割合を増やす必要があるのではないかと考えられる．

役割の割合決定全体に対する考察

「ダイジェストノード」と,「ゲートノード」と「セミゲートノード」の割合のどちらも状況によって変える方が良いという結果だった．とくに後者では具体的に接続数が問題になっていた．本システムでは役割を割合で決定しているが, 接続数に関しては考慮していなかった．ノード数によってクラスタの数を変えているので, 接続数が爆発的に多くなることはないが, 役割の数を決定する方法として割合よりも接続数を考慮すべきであったと考えられる．

役割を与えた場合に対する考察

役割を与えた場合は, ノード数が 200 の時以外は高い値を取っているという結果だった．ノード数が 200 の時に低くなっている理由として, 1 ノード当たりの接続数が少なすぎるからであると考えられる．十分な接続数を確保できないため, 各ノードが取得可能なパケットが少なく, 全体としてスループットの値が低くなっているのではないかと考えられる．

役割を与えない場合に対する考察

役割を与えない場合は, ノード数が 200 の時にスループットが一番高い値を取っているという結果だった．ノード数が多くなるに連れてスループットの値が低くなっていく理由として, 接続数が爆発的に多くなるからではないかと考えられる．役割を与えない場合は全てのノードがノーマルノードであるため, 接続数が分散されずに多くなってしまう．そのためスループットの値が小さくなっていると考えられる．

役割の適切性全体に対する考察

役割を与えたほうが役割を与えない場合に比べ全体的なスループットの値が大きく, 役割を与えることは有用であったという結果だった. 値に差が出てしまった要因としてはやはり接続数の問題があった. 特に役割を与えない場合では接続数の多さがスループット低下の要因であることを顕著に表していた. 結果的に役割を与えることは有用であることはわかったが, さらに性能をあげるためには接続数を考慮すべきであることがわかった.

遅延に対する考察

ホップ数と接続数の 2 つについて実験を行った. 役割を与えた場合は与えない場合に比べて, ホップ数が多く劣っているが, 接続数は少なく優れているという結果だった. 上記で示した考察により, 役割を与えた場合は与えない場合に比べてスループットの値が大きく, 有用であった. その結果, 接続数の数がスループットにも影響していると考えられ, 同時に遅延にも影響していると考えられる.

ダイジェスト保有率に対する考察

ダイジェスト保有率は平均で 76% を記録し, 概要を知るのに十分という結果だった. ダイジェスト保有率が高かった理由として, ダイジェストノードをコメント数の多いノードから選出したことがあげられる. コメント数の多いノードは離脱可能性が低く, そのようなノードにダイジェストを持たせることで, ダイジェストを P2P ネットワーク内に保持出来ると考えられる. しかし, 本研究では 1 ダイジェストノードに 1 ダイジェストを持たせており, ダイジェストノードが抜けると, そのノードが保持していたダイジェストが失われてしまう. この問題を解決するために, 1 つのダイジェストは複数のダイジェストで保持することが良いと考えられる. このようにすることで, 接続数は多くなるものの, ダイジェスト保有率はより高く維持することが出来るはずである.

トポロジ設計の要求条件に対する考察

トポロジ設計の要求条件に対する考察を行う. 各要求条件を示し, それについての考察を行う.

- ライブストリーミングの映像を見られるだけの性能があること
  4.2.3 節の結果, 役割を与えた場合では, ノード数を変化させた時の平均のスループットの値が 406.73kbps であった. これはライブストリーミングの映像を十分に見ることが出来る性能である.

- 少ない遅延でパケットが届くこと
  4.2.4 節の結果, 接続数が遅延に影響していることがわかった. 役割を与えた場合では, ノード数が大きくなっても接続数が低く抑えられ, 少ない遅延でパケットが各ノードに届くと考えられる.
- ダイジェストを P2P ネットワーク内に確保できること
  4.2.5 節の結果, ダイジェスト保有率は平均で 76% を達成した これは配信内容の概要を知るのに十分なダイジェストを P2P ネットワーク内に確保出来ると考えられる.

第 5 章

まとめと今後の課題

5.1 結論

本研究では, P2P ライブストリーミングにおいて, 途中参加したユーザがダイジェスト視聴可能な P2P ライブストリーミングシステムを提案した. システムを実現するために 2 つの段階を考えた. 1 つ目は P2P ネットワーク内でダイジェストを生成することで, 2 つ目は作成したダイジェストを P2P ネットワーク内で保持し, 広めるためのトポロジ設計を行うことである.

1 つ目の段階ではダイジェスト生成方式を提案した. 3 つの方式を提案し, それぞれのダイジェストの適合率を比較した結果, 「最小二乗法に基づくダイジェスト生成方式」が最も適切で, 映像全体の 66% 以上のダイジェストを判定出来ることが分かった.

2 つ目の段階ではトポロジを提案した. 複数クラスタ型において, 各ノードに役割を持たせたトポロジを設計した. シミュレーションでは, 役割を持たせた場合と役割を持たせない場合のネットワーク全体のスループットの値を比較した. その結果, 役割を持たせた場合の平均のスループットの値は 406.73kbps と高く, またノード数が大きくなってもスループットの値は下がらなかった. 役割を持たせることの有用性を確認することが出来た. さらに遅延はホップ数よりも接続数に影響し, 平均の接続数は役割を与えない場合に比べて少なく, 優れていることが分かった. そして, ネットワーク内のダイジェスト保有率は 76% を記録し, ダイジェストを十分に確保することが出来た.

5.2 今後の課題

ダイジェスト生成方式においては, どのような種類にも対応可能な汎用的なダイジェスト生成方式を目指した. しかし, 本研究では 3 種類の動画を対象とした実験しか出来なかった. 今後はより多くの種類の動画に対して方式を適用して有用性の確認をしていく予定である.

また, トポロジ設計では既存の P2P ライブストリーミングシステムとは定性的な評価しか行えなかった. 同じ条件下でシミュレーションを行い, 他研究との比較を行うべきで

ある．さらに, 本研究ではシミュレーション上での評価にとどまっている．実際の P2P ネットワーク上でシステムを動かし, 有用性の検証をすべきである．

謝辞

本研究を遂行するにあたり、様々な方々にお世話になりました。

指導教員である末田欣子先生には, 仕事が忙しいにも関わらず毎週行われたゼミでの終始熱心なご指導を頂きました. また, ゼミ以外でもメール等を介して常に適切な助言を賜りました. ここに厚く御礼申し上げます.

講座内進捗や講座内輪講などにおいて研究を進める上での貴重な意見や助言を頂いた, 多田好克先生, 小宮常康先生, 鶴岡行雄先生, 本庄利守先生には深く感謝します.

研究室の先輩である中原祥吾様, 他の研究室の先輩である藤田竜一様, 村井栄王様, 森中翔太郎様, 片桐国建様, 石田峰文様, 神保直幸様, 若井英之様, 講座の同期である熊谷佑弥様, 山本峻丸様, 吉原大夢様, 磯谷俊明様, 工藤朋哉様, 講座の後輩であるグエン クアン ヒエップ様, 木田純平様, 関湧大様, ゾリーグ ウンダラム様, 李明元様, 新タ智啓様とは日々の研究室生活を共に過ごさせて頂き, 多くの助言を頂きました. ここに感謝の意を表します.

さらに, 私が電気通信大学大学院で様々なことを学ぶことが出来たことは, 両親, 友人, 先生, 先輩, 後輩など, これまでの人生で出会った方々のおかげです. 皆様への心から感謝の気持ちと御礼を申し上げたく、謝辞にかえさせていただきます。

2015 年 1 月吉日 鈴木駿介

参考文献

[1] ニコニコ生放送, http://live.nicovideo.jp/, 2015 年閲覧.

[2] Twicasting, http://twitcasting.tv/, 2015 年閲覧.

[3] Ustream, http://www.ustream.tv/, 2015 年閲覧.

[4] AfreecaTV, http://www.afreeca.com/, 2015 年閲覧.

[5] Yang Guo, Chao Liang, and Yong Liu, “ Hierarchically Clustered P2P Video Streaming: Design, implementation, and evaluation, ”Computer Networks, pp.3432-3445, 2012.

[6] 元橋 智紀, 藤本 章宏, 廣田 悠介, 戸出 英樹, 村上 孝三, “ 多様な配信木により離脱耐性と遅延抑制を向上させる重畳クラスタ木型動画配信システム, ” 通信技術の革新を担う学生論文特集, p.132-142, 2014 年.

[7] Huey-Ing Liu and I-Feng Wu, “ MeTree: A Contribution and Locality-Aware P2P Live Streaming Architecture, ” AINA 24th IEEE International Conference on, pp.1136-1143, 2010.

[8] V. Padmanabhan, H. Wang, P. Chou, and K. Sripanidkulchai, “ Distributing streaming media content using cooperative networking, ” Network and Operating Systems Support for Digital Audio and Video, pp.177—186, May 2002.

[9] G. Bianchi, N. Melazzi, L. Bracciale, F. Piccolo, and S. Salsano, “ Streamline: An optimal distribution al-gorithm for peer-to-peer real-time streaming, ” IEEE Trans. Parallel Distrib. Syst., vol.21, no.6, pp.857—871, June 2010.

[10] 橋本 隆子, 加登 岡隆, 飯沢 篤志, ” スポーツ映像におけるシーン重要度算出アルゴリズムとその評価, ” データ工学ワークショップ, 2003.

[11] 熊野 雅仁, 有木 康雄, 塚田 清志, “ 野球中継のハイライトシーン実時間配信を目的とした特徴のマイニングによる PC シーンの自動検出, ” 映像情報メディア学会誌, vol.59, No.1, pp.77-84, 2005.

[12] Zhengye Liu, Yanming Shen, Ross K.W., Panwar S.S. and Yao Wang,“ Substream Trading: Towards an open P2P live streaming system, ” International Conference on Network Protocols, pp.94-130, 2008.

[13] C. Huang, J. Li, and K. W. Ross, ”Can Internet VoD be profitable?,” ACM SIGCOMM, 2007.

[14] M. Dischinger, A. Haeberlen, K. P. Gummadi, and S. Saroiu, ”Characterizing residential broadband networks,” ACM IMC, 2007.

[15] 大村淳己, 高田和也, 後藤滋樹, “ Location Based Clustering を用いた P2P ストリー

ミング, 電子情報通信学会技術研究報告, ” Vol. 110, No. 373, IN2010-124, pp.37-42, 2011.

[16] “ ニコニコ動画のコメントを分析してみる@ 2013 年冬アニメ ”, http://blog.livedoor.jp/mgpn/archives/51887779.html, 2014 年閲覧.

付録 A

シミュレーションで使用したコード

A.1 役割有無実験

A.1.1 役割有りメインコード

プログラム A.1 役割有りメインコード

```
1  #NS simulator object
2  set ns [new Simulator]
3
4  # デフォルトの値はここで定義
5  source my-goddard-default.tcl
6
7  # 共通の関数はここで定義
8  source my-goddard-procs.tcl
9
10 # 入力値(ユーザ数は必ず200の倍数)
11 set userNum [lindex $argv 0]
12
13 # ユーザ数に応じて変化
14 set clusterNum 0
15
16 # 実験用パラメータ
17 set digestUserRate 0.2
18 set gateBandWidthRate 0.3
19 set gateCommentRate 0.1
20 set semiGateBandWidthRate 0.3
21 set semiGateCommentRate 0.2
22 set notGetDigestRate 0.2
23 set connectNomalNodeRate 0.25
24
25 # ノード
26 set rootNode ""
27 set gateNode(0,0) ""
28 set semiGateNode(0,0) ""
29 set digestNode(0,0) ""
30 set nomalDigestNode(0,0) ""
31 set nomalNotDigestNode(0,0) ""
32
33 # ノードの数
34 set digestNodeNum 0
35 set gateNodeNum 0
36 set semiGateNodeNum 0
37 set nomalNodeNum 0
38 set notGetDigestNomalNum 0
39 set getDigestNomalNum 0
40
41 # ノードリスト
42 set nodeList(0) ""
43 set nodeListForBandwidth(0) ""
44
45 # 帯域幅ノードリスト(Mbps)
46 set bandwidthList(0) ""
47
48 # 一時退避帯域幅ノードリスト
49 set temporalBandwidthList(0) ""
50
51 # コメント数ノードリスト
52 set commentList(0) ""
53
54 # ダイジェスト以外のソートされた帯域幅ノードリスト
55 set sortedBandwidthList(0) ""
56
```

```
57 # goddardのための変数宣言
58 set goddard(0) ""
59 set gplayer(0) ""
60 set sfile(0) ""
61 set gCount 0
62
63 # my-goddardのための関数
64 proc gateNodeInit {gateNode sortedBandwidthList clusterNum gateNodeNum} {
65     upvar $gateNode gn $sortedBandwidthList sbl
66     set k 0
67     for {set i 0} {$i < $clusterNum} {incr i} {
68         for {set j 0} {$j < $gateNodeNum} {incr j} {
69             set gn($i,$j) $sbl($k)
70             # ゲートノードの色
71             $gn($i,$j) color #006400
72             incr k
73         }
74     }
75     return
76 }
77 proc semiGateNodeInit {semiGateNode sortedBandwidthList gateNode clusterNum
       semiGateNodeNum} {
78     upvar $semiGateNode sgn $sortedBandwidthList sbl $gateNode gn
79     set k [array size gn]
80     for {set i 0} {$i < $clusterNum} {incr i} {
81         for {set j 0} {$j < $semiGateNodeNum} {incr j} {
82             set sgn($i,$j) $sbl($k)
83             # セミゲートノードの色
84             $sgn($i,$j) color #00ff00
85             incr k
86         }
87     }
88     return
89 }
90 proc nomalNodeInit {nomalNotDigestNode nomalDigestNode gateNode semiGateNode
       sortedBandwidthList clusterNum notGetDigestNomalNum getDigestNomalNum} {
91     upvar $nomalNotDigestNode nndn $nomalDigestNode ndn $gateNode gn $semiGateNode sgn
         $sortedBandwidthList sbl
92     set k [expr [array size gn] + [array size sgn]]
93     for {set i 0} {$i < $clusterNum} {incr i} {
94         for {set j 0} {$j < $notGetDigestNomalNum} {incr j} {
95             set nndn($i,$j) $sbl($k)
96             # ダイジェスト未取得ノーマルノードの色
97             $nndn($i,$j) color pink
98             incr k
99         }
100    }
101    for {set i 0} {$i < $clusterNum} {incr i} {
102        for {set j 0} {$j < $getDigestNomalNum} {incr j} {
103            set ndn($i,$j) $sbl($k)
104            # ダイジェスト取得ノーマルノードの色
105            $ndn($i,$j) color orange
106            incr k
107        }
108    }
109    # 残りのノードはu全てダイジェスト取得済みノーマルノードへ
110    set limit [expr [array size sbl]-$k]
111    for {set i 0} {$i < $limit} {incr i} {
112        set ndn($i,$getDigestNomalNum) $sbl($k)
113        # ダイジェスト取得ノーマルノードの色
114        $ndn($i,$getDigestNomalNum) color orange
115        incr k
116    }
117    return
118 }
119 # この中で便宜上一時的に帯域幅リストからノードを削除している
120 proc digestNodeInit {digestNode bandwidthList temporalBandwidthList nodeListForBandwidth
       nodeList userNum clusterNum digestNodeNum} {
121    upvar $digestNode dn $bandwidthList bl $temporalBandwidthList tbl
         $nodeListForBandwidth nlfb $nodeList nl
122    copy bl tbl
123    set commentI [expr $userNum-1]
124    for {set i 0} {$i < $clusterNum} {incr i} {
125        for {set j 0} {$j < $digestNodeNum} {incr j} {
126            set dn($i,$j) $nl($commentI)
127            # 帯域幅リストからダイジェストノードを削除
128            array unset bl $nl($commentI)
129            # 帯域幅ノードリストからダイジェストノードを削除
130            for {set k 0} {$k < [array size nlfb]} {incr k} {
131                if {[array get nlfb $k] == []} {
132                    continue
133                }
134                if {$nlfb($k) == $nl($commentI)} {
135                    array unset nlfb $k
136                    break
137                }
138            }
```

```
139             # ダイジェストノードの色
140             $dn($i,$j) color yellow
141             decr commentI
142         }
143     }
144     return
145 }
146
147 # ノード間の接続
148 # 常に低いノード側の帯域幅で接続
149 # 帯域幅の設定する必要あり
150 proc connectGateNodeInCluster {gateNode semiGateNode bandwidthList rootNode ns gateNodeNum
        clusterNum selfClusterNum} {
151     upvar $gateNode gn $semiGateNode sgn $bandwidthList bl
152     # 配信者ノード
153     for {set i 0} {$i < $gateNodeNum} {incr i} {
154         $ns duplex-link $gn($selfClusterNum,$i) $rootNode $bl($gn($selfClusterNum,$i))Mb
              500ms DropTail
155     }
156     # ゲートノード同士 : 1 → 2   2 → 3   3 → 1
157     for {set i 0} {$i < $gateNodeNum} {incr i} {
158         if { [expr $i+1] >= $gateNodeNum } {
159             set bandwidth [returnLowBandwidth bl $gn($selfClusterNum,$i) $gn(
                    $selfClusterNum,[expr $i+1-$gateNodeNum])]
160             $ns duplex-link $gn($selfClusterNum,$i) $gn($selfClusterNum,[expr $i+1
                    -$gateNodeNum]) [expr $bandwidth]Mb 100ms DropTail
161         } else {
162             set bandwidth [returnLowBandwidth bl $gn($selfClusterNum,$i) $gn(
                    $selfClusterNum,[expr $i+1])]
163             $ns duplex-link $gn($selfClusterNum,$i) $gn($selfClusterNum,[expr $i+1]) [expr
                     $bandwidth]Mb 100ms DropTail
164         }
165     }
166     # ゲートノードとセミゲートノード
167     for {set i 0} {$i < $gateNodeNum} {incr i} {
168         set bandwidth [returnLowBandwidth bl $gn($selfClusterNum,$i) $sgn(
                $selfClusterNum,$i)]
169         $ns duplex-link $gn($selfClusterNum,$i) $sgn($selfClusterNum,$i) [expr $bandwidth]
              Mb 100ms DropTail
170     }
171 }
172
173 proc connectGateNodeOutside {gateNode bandwidthList nsArg clusterNum gateNodeNum
      selfIndexNum} {
174     upvar $gateNode gn $bandwidthList bl $nsArg ns
175     # クラスタ外のゲートノード同士 : 1 → 2  2 → 3 ... 7→ 1
176     for {set i 0} {$i < $clusterNum} {incr i} {
177         if { [expr $i+1] >= $clusterNum } {
178             set bandwidth [returnLowBandwidth bl $gn($i,$selfIndexNum) $gn([expr $i+1
                    -$clusterNum],$selfIndexNum)]
179             $ns duplex-link $gn($i,$selfIndexNum) $gn([expr $i+1-$clusterNum]
                    ,$selfIndexNum) [expr $bandwidth]Mb 100ms DropTail
180         } else {
181             set bandwidth [returnLowBandwidth bl $gn($i,$selfIndexNum) $gn([expr $i+1]
                    ,$selfIndexNum)]
182             $ns duplex-link $gn($i,$selfIndexNum) $gn([expr $i+1],$selfIndexNum) [expr
                     $bandwidth]Mb 100ms DropTail
183         }
184     }
185 }
186
187 proc connectSemiGateNode {semiGateNode digestNode nomalDigestNode nomalNotDigestNode
      bandwidthList ns clusterNum semiGateNodeNum notGetDigestNomalNum getDigestNomalNum
      selfIndexNum } {
188     upvar $semiGateNode sgn $digestNode dn $nomalDigestNode ndn $nomalNotDigestNode nndn
            $bandwidthList bl
189     # ダイジェストノード
190     for {set i 0} {$i < $semiGateNodeNum} {incr i} {
191         if {[array get dn $selfIndexNum,[expr $i*2]] == []} {
192             continue
193         }
194         set bandwidth [returnLowBandwidth bl $sgn($selfIndexNum,$i) $dn($selfIndexNum,[
                expr $i*2])]
195         $ns duplex-link $sgn($selfIndexNum,$i) $dn($selfIndexNum,[expr $i*2]) [expr
                $bandwidth]Mb 100ms DropTail
196         if {[array get dn $selfIndexNum,[expr $i*2+1]] == []} {
197             continue
198         }
199         set bandwidth [returnLowBandwidth bl $sgn($selfIndexNum,$i) $dn($selfIndexNum,[
                expr $i*2+1])]
200         $ns duplex-link $sgn($selfIndexNum,$i) $dn($selfIndexNum,[expr $i*2+1]) [expr
                $bandwidth]Mb 100ms DropTail
201     }
202     # ノーマルノード
203     for {set i 0} {$i < $semiGateNodeNum} {incr i} {
204         set digestBorderNum [expr int(($notGetDigestNomalNum+$getDigestNomalNum)*rand())]
205         if {$digestBorderNum >= $notGetDigestNomalNum} {
```

```
206             set bandwidth [returnLowBandwidth bl $sgn($selfIndexNum,$i) $ndn(
                    $selfIndexNum,[expr $digestBorderNum-$notGetDigestNomalNum])]
207             $ns duplex-link $sgn($selfIndexNum,$i) $ndn($selfIndexNum,[expr
                    $digestBorderNum-$notGetDigestNomalNum]) [expr $bandwidth]Mb 100ms
                    DropTail
208         } else {
209             set bandwidth [returnLowBandwidth bl $sgn($selfIndexNum,$i) $nndn(
                    $selfIndexNum,$digestBorderNum)]
210             $ns duplex-link $sgn($selfIndexNum,$i) $nndn($selfIndexNum,$digestBorderNum) [
                    expr $bandwidth]Mb 100ms DropTail
211         }
212     }
213 }
214
215 proc connectDigestNode {digestNode nomalNotDigestNode bandwidthList ns
      notGetDigestNomalNum getDigestNomalNum digestNodeNum selfIndexNum } {
216     upvar $digestNode dn $nomalNotDigestNode nndn $bandwidthList bl
217     # ダイジェスト未取得ノーマルノード
218     for {set i 0} {$i < $digestNodeNum} {incr i} {
219         for {set j 0} {$j < $notGetDigestNomalNum} {incr j} {
220             set bandwidth [returnLowBandwidth bl $dn($selfIndexNum,$i) $nndn(
                    $selfIndexNum,$j)]
221             $ns duplex-link $dn($selfIndexNum,$i) $nndn($selfIndexNum,$j) [expr $bandwidth
                    ]Mb 100ms DropTail
222         }
223     }
224 }
225
226 proc connectNomalNode {nomalDigestNode nomalNotDigestNode bandwidthList ns clusterNum
      connectNomalNodeRate notGetDigestNomalNum getDigestNomalNum nomalNodeNum selfIndexNum
       } {
227     upvar $nomalDigestNode ndn $nomalNotDigestNode nndn $bandwidthList bl
228     # とりあえずリストに全部入れる
229     for {set i 0} {$i < [expr $nomalNodeNum+1]} {incr i} {
230         if {$i >= $notGetDigestNomalNum} {
231             if {[array get ndn $selfIndexNum,[expr $i-$notGetDigestNomalNum]] == []} {
232                 continue
233             }
234             set nomalNodeList($i) $ndn($selfIndexNum,[expr $i-$notGetDigestNomalNum])
235         } else {
236             set nomalNodeList($i) $nndn($selfIndexNum,$i)
237         }
238     }
239     # 適当な回数リストの中身をシャッフル
240     set temp ""
241     for {set i 0} {$i < 100 } {incr i} {
242         set randomNum1 [expr int(($nomalNodeNum)*rand())]
243         set randomNum2 [expr int(($nomalNodeNum)*rand())]
244         set temp $nomalNodeList($randomNum1)
245         set $nomalNodeList($randomNum1) $nomalNodeList($randomNum2)
246         set $nomalNodeList($randomNum2) $temp
247     }
248     set connectNomalNum [expr int(ceil($nomalNodeNum*$connectNomalNodeRate))]
249     # ノーマルノード同士：0 → 1　0 → 2　0 → 3　0 → 4 、1 → 2　1 → 3
          ...1 4 → 1 5　1 4 → 0　1 4 → 1　1 4 → 2
250     for {set i 0} {$i < [expr $nomalNodeNum+1]} {incr i} {
251         for {set j 0} {$j < $connectNomalNum} {incr j} {
252             if { [expr $i+$j+1] >= $nomalNodeNum } {
253                 if {[array get nomalNodeList $i] == []} {
254                     continue
255                 }
256                 set bandwidth [returnLowBandwidth bl $nomalNodeList($i) $nomalNodeList([
                        expr $i+$j+1-$nomalNodeNum])]
257                 $ns duplex-link $nomalNodeList($i) $nomalNodeList([expr $i+$j+1
                        -$nomalNodeNum]) [expr $bandwidth]Mb 100ms DropTail
258             } else {
259                 set bandwidth [returnLowBandwidth bl $nomalNodeList($i) $nomalNodeList([
                        expr $i+$j+1])]
260                 $ns duplex-link $nomalNodeList($i) $nomalNodeList([expr $i+$j+1]) [expr
                        $bandwidth]Mb 100ms DropTail
261             }
262         }
263     }
264 }
265
266 #Define a 'finish' procedure
267 proc finish {} {
268     global ns f gCount sfile userNum
269     $ns flush-trace
270     set awkCode {
271         {
272             if ($8 == 3000) {
273                 if ($2 >= t_end_tcp) {
274                     tput_tcp = bytes_tcp * 8 / ($2 - t_start_tcp)/1000;
275                     print $2, tput_tcp >> "tput-tcp.tr";
276                     t_start_tcp = $2;
277                     t_end_tcp = $2 + 2;
```

```
278                     bytes_tcp = 0;
279                 }
280                 if ($1 == "r") {
281                     bytes_tcp += $6;
282                 }
283             }
284             else if ($8 == 3001) {
285                 if ($2 >= t_end_udp) {
286                     tput_udp = bytes_udp * 8 / ($2 - t_start_udp)/1000;
287                     print $2, tput_udp >> "tput-udp.tr";
288                     t_start_udp = $2;
289                     t_end_udp = $2 + 2;
290                     bytes_udp = 0;
291                 }
292                 if ($1 == "r") {
293                     bytes_udp += $6;
294                 }
295             }
296         }
297     }
298     for {set i 0} {$i < $gCount} {incr i} {
299         if { [info exists sfile($i)] } {
300             close $sfile($i)
301         }
302     }
303     close $f
304     exec rm -f tput-tcp.tr tput-udp.tr
305     exec touch tput-tcp.tr tput-udp.tr
306     exec awk $awkCode out.tr
307     exec cp out.nam [append outNamName "out" $userNum ".nam"]
308     exec cp out.tr [append outTrName "out" $userNum ".tr"]
309     exec cp tput-tcp.tr [append tputTcpName "tput-tcp" $userNum ".tr"]
310     exec cp tput-udp.tr [append tputUdpName "tput-udp" $userNum ".tr"]
311     exec xgraph -bb -tk -m -x Seconds -y "Throughput (kbps)" tput-tcp.tr tput-udp.tr &
312     exec nam out.nam &
313     exit 0
314 }
315
316 ## 処理開始
317 setPacketColor $ns
318 setClusterNum clusterNum $userNum
319 setNodeNum digestNodeNum gateNodeNum semiGateNodeNum nomalNodeNum notGetDigestNomalNum
        getDigestNomalNum $userNum $clusterNum $digestUserRate $gateCommentRate
        $semiGateCommentRate $gateCommentRate $semiGateNodeNum $notGetDigestRate
320
321 puts "1クラスタ当たりのノードの数\n"
322 puts "ダイジェストノード: \t\t\t$digestNodeNum"
323 puts "ゲートノード: \t\t\t\t$gateNodeNum"
324 puts "セミゲートノード: \t\t\t$semiGateNodeNum"
325 puts "ノーマルノード: \t\t\t$nomalNodeNum"
326 puts "ダイジェスト未取得ノーマルノード: \t$notGetDigestNomalNum"
327 puts "ダイジェスト取得済みノーマルノード: \t$getDigestNomalNum"
328
329 ratioSetting bandwidthRatio commentRatio $clusterNum $userNum
330
331 # 各ノードリストのinit処理
332 nodeListInit nodeList nodeListForBandwidth $ns $userNum
333 bandwidthListInit bandwidthList bandwidthRatio nodeListForBandwidth $ns $userNum
334 commentListInit commentList commentRatio nodeList $ns $userNum
335 nodeListForBandwidthShuffle nodeListForBandwidth $userNum
336
337 # 各役割のinit処理
338 rootNodeInit rootNode $ns
339 digestNodeInit digestNode bandwidthList temporalBandwidthList nodeListForBandwidth
        nodeList $userNum $clusterNum $digestNodeNum
340 sortBandwidthList sortedBandwidthList bandwidthRatio bandwidthList
341 gateNodeInit gateNode sortedBandwidthList $clusterNum $gateNodeNum
342 semiGateNodeInit semiGateNode sortedBandwidthList gateNode $clusterNum $semiGateNodeNum
343 nomalNodeInit nomalNotDigestNode nomalDigestNode gateNode semiGateNode sortedBandwidthList
        $clusterNum $notGetDigestNomalNum $getDigestNomalNum
344
345 puts "\nノードの数\n"
346 puts "ダイジェストノード: \t\t\t[array size digestNode]"
347 puts "ゲートノード: \t\t\t\t[array size gateNode]"
348 puts "セミゲートノード: \t\t\t[array size semiGateNode]"
349 puts "ダイジェスト未取得ノーマルノード: \t[array size nomalNotDigestNode]"
350 puts "ダイジェスト取得済みノーマルノード: \t[array size nomalDigestNode]"
351
352 # namファイルの設定
353 set f [open out.tr w]
354 $ns trace-all $f
355 set nf [open out.nam w]
356 $ns namtrace-all $nf
357
358 # 一時的にノードを削除していたので帯域幅リストを元に戻す
359 copy temporalBandwidthList bandwidthList
360
```

```
361 # ゲートノードの数実行
362 for {set i 0} {$i < $gateNodeNum} {incr i} {
363     connectGateNodeOutside gateNode bandwidthList ns $clusterNum $gateNodeNum $i
364 }
365
366 # クラスタの数実行
367 for {set i 0} {$i < $clusterNum} {incr i} {
368     connectGateNodeInCluster gateNode semiGateNode bandwidthList $rootNode $ns
          $gateNodeNum $clusterNum $i
369     connectSemiGateNode semiGateNode digestNode nomalDigestNode nomalNotDigestNode
          bandwidthList $ns $clusterNum $semiGateNodeNum $notGetDigestNomalNum
          $getDigestNomalNum $i
370     connectDigestNode digestNode nomalNotDigestNode bandwidthList $ns
          $notGetDigestNomalNum $getDigestNomalNum $digestNodeNum $i
371     connectNomalNode nomalDigestNode nomalNotDigestNode bandwidthList $ns $clusterNum
          $connectNomalNodeRate $notGetDigestNomalNum $getDigestNomalNum $nomalNodeNum $i
372 }
373
374 createNomalNodeStream nomalDigestNode nomalNotDigestNode digestNode goddard gplayer sfile
      gCount $rootNode $ns $clusterNum $getDigestNomalNum $notGetDigestNomalNum
      $digestNodeNum
375
376 # Scehdule Simulation
377 for {set i 0} {$i < $gCount} {incr i} {
378     $ns at 0 "$goddard($i) start"
379     $ns at 240.0 "$goddard($i) stop"
380 }
381
382 $ns at 240.0 "finish"
383
384 $ns run
```

A.1.2 役割無しメインコード

プログラム A.2 役割無しメインコード

```
1  #NS simulator object
2  set ns [new Simulator]
3
4  # デフォルトの値はここで定義
5  source my-goddard-default.tcl
6
7  # 共通の関数はここで定義
8  source my-goddard-procs.tcl
9
10 # 入力値(ユーザ数は必ず200の倍数)
11 set userNum [lindex $argv 0]
12
13 # ユーザ数に応じて変化
14 set clusterNum 0
15
16 # 実験用パラメータ
17 set digestUserRate 0
18 set gateBandWidthRate 0
19 set gateCommentRate 0
20 set semiGateBandWidthRate 0
21 set semiGateCommentRate 0
22 set notGetDigestRate 0
23 set connectNomalNodeRate 0.25
24
25 # ノード
26 set rootNode ""
27 set gateNode(0,0) ""
28 set semiGateNode(0,0) ""
29 set digestNode(0,0) ""
30 set nomalDigestNode(0,0) ""
31 set nomalNotDigestNode(0,0) ""
32
33 # ノードの数
34 set digestNodeNum 0
35 set gateNodeNum 0
36 set semiGateNodeNum 0
37 set nomalNodeNum 0
38 set notGetDigestNomalNum 0
39 set getDigestNomalNum 0
40
41 # ノードリスト
42 set nodeList(0) ""
43 set nodeListForBandwidth(0) ""
44
45 # 帯域幅ノードリスト(Mbps)
```

```
46 set bandwidthList(0) ""
47
48 # 一時退避帯域幅ノードリスト
49 set temporalBandwidthList(0) ""
50
51 # コメント数ノードリスト
52 set commentList(0) ""
53
54 # ダイジェスト以外のソートされた帯域幅ノードリスト
55 set sortedBandwidthList(0) ""
56
57 # goddardのための変数宣言
58 set goddard(0) ""
59 set gplayer(0) ""
60 set sfile(0) ""
61 set gCount 0
62
63 # my-goddard-no-rollのための関数
64 proc nomalNodeInit {nomalDigestNode nomalNotDigestNode sortedBandwidthList clusterNum
       notGetDigestNomalNum getDigestNomalNum} {
65     upvar $nomalDigestNode ndn $nomalNotDigestNode nndn $sortedBandwidthList sbl
66     set k 0
67     for {set i 0} {$i < $clusterNum} {incr i} {
68         for {set j 0} {$j < $getDigestNomalNum} {incr j} {
69             set ndn($i,$j) $sbl($k)
70             # ダイジェスト取得ノーマルノードの色
71             $ndn($i,$j) color orange
72             incr k
73         }
74     }
75     # 残りのノードは全てダイジェスト取得済みノーマルノードへ
76     set k [expr $k]
77     set limit [expr [array size sbl]-$k]
78     for {set i 0} {$i < $limit} {incr i} {
79         set ndn($i,$getDigestNomalNum) $sbl($k)
80         # ダイジェスト取得ノーマルノードの色
81         $ndn($i,$getDigestNomalNum) color orange
82         incr k
83     }
84     return
85 }
86
87 proc connectNomalNode {nomalDigestNode nomalNotDigestNode bandwidthList rootNode ns
       clusterNum connectNomalNodeRate notGetDigestNomalNum getDigestNomalNum nomalNodeNum
       selfIndexNum } {
88     upvar $nomalDigestNode ndn $nomalNotDigestNode nndn $bandwidthList bl
89     # とりあえずリストに全部入れる
90     for {set i 0} {$i < [expr $nomalNodeNum+1]} {incr i} {
91         if {[array get ndn $selfIndexNum,[expr $i-$notGetDigestNomalNum]] == []} {
92             continue
93         }
94         set nomalNodeList($i) $ndn($selfIndexNum,[expr $i-$notGetDigestNomalNum])
95     }
96     # 適当な回数リストの中身をシャッフル
97     set temp ""
98     for {set i 0} {$i < 100 } {incr i} {
99         set randomNum1 [expr int(($nomalNodeNum)*rand())]
100        set randomNum2 [expr int(($nomalNodeNum)*rand())]
101        set temp $nomalNodeList($randomNum1)
102        set $nomalNodeList($randomNum1) $nomalNodeList($randomNum2)
103        set $nomalNodeList($randomNum2) $temp
104    }
105    set connectNomalNum [expr int(ceil($nomalNodeNum*$connectNomalNodeRate))]
106    # ノーマルノード同士：0→1　0→2　0→3　0→4、1→2　1→3
         ...14→15　14→0　14→1　14→2
107    for {set i 0} {$i < [expr $nomalNodeNum+1]} {incr i} {
108        for {set j 0} {$j < $connectNomalNum} {incr j} {
109            if { [expr $i+$j+1] >= $nomalNodeNum } {
110                if {[array get nomalNodeList $i] == []} {
111                    continue
112                }
113                set bandwidth [returnLowBandwidth bl $nomalNodeList($i) $nomalNodeList([
                       expr $i+$j+1-$nomalNodeNum])]
114                $ns duplex-link $nomalNodeList($i) $nomalNodeList([expr $i+$j+1
                       -$nomalNodeNum]) [expr $bandwidth]Mb 100ms DropTail
115            } else {
116                set bandwidth [returnLowBandwidth bl $nomalNodeList($i) $nomalNodeList([
                       expr $i+$j+1])]
117                $ns duplex-link $nomalNodeList($i) $nomalNodeList([expr $i+$j+1]) [expr
                       $bandwidth]Mb 100ms DropTail
118            }
119        }
120    }
121    # 配信者ノード
122    $ns duplex-link $nomalNodeList(0) $rootNode $bl($nomalNodeList(0))Mb 500ms DropTail
123 }
124
```

```
125 #Define a 'finish' procedure
126 proc finish {} {
127     global ns f gCount sfile userNum
128     $ns flush-trace
129     set awkCode {
130         {
131             if ($8 == 3000) {
132                 if ($2 >= t_end_tcp) {
133                     tput_tcp = bytes_tcp * 8 / ($2 - t_start_tcp)/1000;
134                     print $2, tput_tcp >> "tput-tcp.tr";
135                     t_start_tcp = $2;
136                     t_end_tcp = $2 + 2;
137                     bytes_tcp = 0;
138                 }
139                 if ($1 == "r") {
140                     bytes_tcp += $6;
141                 }
142             }
143             else if ($8 == 3001) {
144                 if ($2 >= t_end_udp) {
145                     tput_udp = bytes_udp * 8 / ($2 - t_start_udp)/1000;
146                     print $2, tput_udp >> "tput-udp.tr";
147                     t_start_udp = $2;
148                     t_end_udp = $2 + 2;
149                     bytes_udp = 0;
150                 }
151                 if ($1 == "r") {
152                     bytes_udp += $6;
153                 }
154             }
155         }
156     }
157     for {set i 0} {$i < $gCount} {incr i} {
158         if { [info exists sfile($i)] } {
159             close $sfile($i)
160         }
161     }
162     close $f
163     exec rm -f tput-tcp.tr tput-udp.tr
164     exec touch tput-tcp.tr tput-udp.tr
165     exec awk $awkCode out.tr
166     exec xgraph -bb -tk -m -x Seconds -y "Throughput (kbps)" tput-tcp.tr tput-udp.tr &
167     exec nam out.nam
168     exec cp out.nam [append outNamName "out" $userNum "-no-roll.nam"]
169     exec cp out.tr [append outTrName "out" $userNum "-no-roll.tr"]
170     exec cp tput-tcp.tr [append tputTcpName "tput-tcp" $userNum "-no-roll.tr"]
171     exec cp tput-udp.tr [append tputUdpName "tput-udp" $userNum "-no-roll.tr"]
172     exit 0
173 }
174
175 ## 処理開始
176 setPacketColor $ns
177 setClusterNum clusterNum $userNum
178 setNodeNum digestNodeNum gateNodeNum semiGateNodeNum nomalNodeNum notGetDigestNomalNum
        getDigestNomalNum $userNum $clusterNum $digestUserRate $gateCommentRate
        $semiGateCommentRate $gateCommentRate $semiGateNodeNum $notGetDigestRate
179
180 puts "1クラスタ当たりのノードの数\n"
181 puts "ダイジェストノード: \t\t\t$digestNodeNum"
182 puts "ゲートノード: \t\t\t\t$gateNodeNum"
183 puts "セミゲートノード: \t\t\t$semiGateNodeNum"
184 puts "ノーマルノード: \t\t\t$nomalNodeNum"
185 puts "ダイジェスト未取得ノーマルノード: \t$notGetDigestNomalNum"
186 puts "ダイジェスト取得済みノーマルノード: \t$getDigestNomalNum"
187
188 ratioSetting bandwidthRatio commentRatio $clusterNum $userNum
189
190 # 各ノードリストのinit処理
191 nodeListInit nodeList nodeListForBandwidth $ns $userNum
192 bandwidthListInit bandwidthList bandwidthRatio nodeListForBandwidth $ns $userNum
193 commentListInit commentList commentRatio nodeList $ns $userNum
194 nodeListForBandwidthShuffle nodeListForBandwidth $userNum
195
196 # 各役割のinit処理
197 rootNodeInit rootNode $ns
198 sortBandwidthList sortedBandwidthList bandwidthRatio bandwidthList
199 nomalNodeInit nomalDigestNode nomalNotDigestNode sortedBandwidthList $clusterNum
        $notGetDigestNomalNum $getDigestNomalNum
200
201 puts "\nノードの数\n"
202 puts "ダイジェストノード: \t\t\t[array size digestNode]"
203 puts "ゲートノード: \t\t\t\t[array size gateNode]"
204 puts "セミゲートノード: \t\t\t[array size semiGateNode]"
205 puts "ダイジェスト未取得ノーマルノード: \t[array size nomalNotDigestNode]"
206 puts "ダイジェスト取得済みノーマルノード: \t[array size nomalDigestNode]"
207
208 # namファイルの設定
```

```
209 set f [open out.tr w]
210 $ns trace-all $f
211 set nf [open out.nam w]
212 $ns namtrace-all $nf
213
214 # 一時的にノードを削除していたので帯域幅リストを元に戻す
215 copy temporalBandwidthList bandwidthList
216
217 # クラスタの数実行
218 for {set i 0} {$i < $clusterNum} {incr i} {
219     connectNomalNode nomalDigestNode nomalNotDigestNode bandwidthList $rootNode $ns
          $clusterNum $connectNomalNodeRate $notGetDigestNomalNum $getDigestNomalNum
          $nomalNodeNum $i
220 }
221
222 createNomalNodeStream nomalDigestNode nomalNotDigestNode digestNode goddard gplayer sfile
      gCount $rootNode $ns $clusterNum $getDigestNomalNum $notGetDigestNomalNum
      $digestNodeNum
223
224 # Scehdule Simulation
225 for {set i 0} {$i < $gCount} {incr i} {
226     $ns at 0 "$goddard($i) start"
227     $ns at 240.0 "$goddard($i) stop"
228 }
229
230 $ns at 240.0 "finish"
231
232 $ns run
```

A.1.3 外部プロシージャ共通コード

プログラム A.3 外部プロシージャ共通コード

```
1  # 処理のための一般的なメソッド
2  proc decr { int { n 1 } } {
3      if { [ catch {
4          uplevel incr $int -$n
5      } err ] } {
6          return -code error "decr: $err"
7      }
8      return [ uplevel set $int ]
9  }
10
11 # 配列をコピー
12 proc copy {ary1 ary2} {
13     upvar $ary1 from $ary2 to
14     foreach {index value} [array get from *] {
15         set to($index) $value
16     }
17 }
18
19 # 低い方の帯域幅を返す
20 proc returnLowBandwidth {bandwidthList node1 node2} {
21     upvar $bandwidthList bl
22     if { $bl($node1) >= $bl($node2) } {
23         return $bl($node2)
24     } else {
25         return $bl($node1)
26     }
27 }
28
29 # ここからシミュレータの処理
30 # パケットの色設定
31 proc setPacketColor { ns } {
32     $ns color 0 blue
33     $ns color 1 red
34     $ns color 2 white
35 }
36
37 # ノードの設定
38 proc setClusterNum {clusterNumArg userNum} {
39     upvar $clusterNumArg clusterNum
40     if {$userNum == 200} {
41         set clusterNum 7
42     } elseif {$userNum == 400} {
43         set clusterNum 10
44     } elseif {$userNum == 600} {
45         set clusterNum 14
46     } elseif {$userNum == 800} {
47         set clusterNum 18
48     }
```

```
49 }
50
51 proc setNodeNum {digestNodeNum gateNodeNum semiGateNodeNum nomalNodeNum
       notGetDigestNomalNum getDigestNomalNum userNum clusterNum digestUserRate
       gateCommentRate semiGateCommentRate gateCommentRate semiGateNodeNum notGetDigestRate}
        {
52     upvar $digestNodeNum dnn $gateNodeNum gnn $semiGateNodeNum sgnn $nomalNodeNum nnn
           $notGetDigestNomalNum ngdn $getDigestNomalNum gdnn
53     set dnn [expr int(ceil([expr $userNum / $clusterNum * $digestUserRate]))]
54     set gnn [expr int(ceil([expr $userNum / $clusterNum * $gateCommentRate]))]
55     set sgnn [expr int(ceil([expr $userNum / $clusterNum * ($semiGateCommentRate -
           $gateCommentRate)]))]
56     set nnn [expr $userNum / $clusterNum - $dnn - $gnn - $sgnn]
57     set ngdn [expr int(ceil([expr $nnn * $notGetDigestRate]))]
58     set gdnn [expr $nnn - $ngdn]
59 }
60
61 proc ratioSetting {bandwidthRatio commentRatio clusterNum userNum} {
62     upvar $bandwidthRatio br $commentRatio cr
63     set basicRatio [expr $userNum/200]
64     foreach {index val} [array get br] {
65         set tempBandwidthRatio($index) [expr $val*$basicRatio]
66     }
67     copy tempBandwidthRatio br
68     foreach {index val} [array get cr] {
69         set tempCommentRatio($index) [expr $val*$basicRatio]
70     }
71     copy tempCommentRatio cr
72 }
73
74 proc nodeListInit {nodeList nodeListForBandwidth ns userNum} {
75     upvar $nodeList nl $nodeListForBandwidth nlfb
76     for {set i 0} {$i < $userNum} {incr i} {
77         set nl($i) [$ns node]
78         set nlfb($i) $nl($i)
79     }
80 }
81
82 proc bandwidthListInit {bandwidthList bandwidthRatio nodeListForBandwidth ns userNum} {
83     upvar $bandwidthList bl $bandwidthRatio br $nodeListForBandwidth nlfb
84     set j 0
85     foreach {index val} [array get br] {
86         for {set i 0} {$i < $val} {incr i} {
87             set bl($nlfb($j)) $index
88             incr j
89         }
90     }
91 }
92
93 proc commentListInit {commentList commentRatio nodeList ns userNum } {
94     upvar $commentList cl $commentRatio cr $nodeList nl
95     set j 0
96     foreach {index val} [array get cr] {
97         for {set i 0} {$i < $val} {incr i} {
98             set cl($nl($j)) $index
99             incr j
100        }
101    }
102 }
103
104 proc nodeListForBandwidthShuffle {nodeListForBandwidth userNum} {
105    upvar $nodeListForBandwidth nlfb
106    for {set i 0} {$i < [expr $userNum*5]} {incr i} {
107        set temp1 [expr int($userNum*rand())]
108        set temp2 [expr int($userNum*rand())]
109        set tempNode $nlfb($temp1)
110        set nlfb($temp1) $nlfb($temp2)
111        set nlfb($temp2) $tempNode
112    }
113 }
114
115 proc rootNodeInit {rootNode ns} {
116    upvar $rootNode rn
117    set rn [$ns node]
118    # 配信者ノードの色
119    $rn color red
120 }
121
122 proc sortBandwidthList {sortedBandwidthList bandwidthRatio bandwidthList} {
123    upvar $sortedBandwidthList sbl $bandwidthRatio br $bandwidthList bl
124    # 帯域幅の種類のリスト
125    set i 0
126    foreach val [lsort -real [array names br]] {
127        set kindOfBandwidthList($i) $val
128        incr i
129    }
130    set k 0
```

```
131     for {set i [expr [array size kindOfBandwidthList]-1]} {$i >= 0} {decr i} {
132         foreach {index val} [array get bl] {
133             if {$val == $kindOfBandwidthList($i)} {
134                 set sbl($k) $index
135                 incr k
136             }
137         }
138     }
139 }
140
141 # Setup Goddard Streaming
142 # goddardストリーミング生成関数
143 proc createGoddard { goddard gplayer sfile gCount ns l_node r_node } {
144     upvar $goddard gd $gplayer gp $sfile sf $gCount gc
145     set gs($gc) [new GoddardStreaming $ns $l_node $r_node UDP 1000 $gc]
146     set gd($gc) [$gs($gc) getobject goddard]
147     set gp($gc) [$gs($gc) getobject gplayer]
148     $gp($gc) set upscale_interval_ 30.0
149     set sf($gc) [open stream-udp.tr w]
150     $gp($gc) attach $sf($gc)
151     incr gc
152     return
153 }
154
155 # create goddard
156 proc createNomalNodeStream {nomalDigestNode nomalNotDigestNode digestNode goddard gplayer
        sfile gCount rootNode ns clusterNum getDigestNomalNum notGetDigestNomalNum
        digestNodeNum} {
157     upvar $nomalDigestNode ndn $nomalNotDigestNode nndn $digestNode dn $goddard gd
          $gplayer gp $sfile sf $gCount gc
158     for {set i 0} {$i < $clusterNum} {incr i} {
159         for {set j 0} {$j < $getDigestNomalNum} {incr j} {
160             createGoddard gd gp sf gc $ns $rootNode $ndn($i,$j)
161         }
162         for {set j 0} {$j < $notGetDigestNomalNum} {incr j} {
163             createGoddard gd gp sf gc $ns $rootNode $nndn($i,$j)
164         }
165         for {set j 0} {$j < $digestNodeNum} {incr j} {
166             createGoddard gd gp sf gc $ns $rootNode $dn($i,$j)
167         }
168     }
169 }
```

A.1.4 デフォルトパラメータコード

プログラム A.4 デフォルトパラメータコード

```
1  # 帯域幅割合
2  array set bandwidthRatio {
3      3.000 30
4      1.500 3
5      1.024 56
6      0.768 13
7      0.640 3
8      0.512 4
9      0.448 25
10     0.384 17
11     0.320 29
12     0.256 20
13 }
14
15 # 全体的に高めの場合
16 # array set bandwidthRatio {
17 # 0.768 70
18 # 1.500 80
19 # 3.000 50
20 # }
21
22 # コメント数割合
23 array set commentRatio {
24     25 2
25     22 2
26     20 3
27     17 5
28     15 22
29     12 28
30     10 33
31     7 36
32     5 36
33     2 33
34 }
```

A.2 参加離脱実験

A.2.1 メインコード

プログラム A.5 メインコード

```
#NS simulator object
set ns [new Simulator -multicast on]

# デフォルトの値はここで定義
source my-goddard-default.tcl

# 共通の関数はここで定義
source my-goddard-procs.tcl

# 入力値(ユーザ数は必ず200の倍数)
set userNum 28

# ユーザ数に応じて変化
set clusterNum 0

# 実験用パラメータ
set digestUserRate 0.2
set gateBandWidthRate 0.3
set gateCommentRate 0.1
set semiGateBandWidthRate 0.3
set semiGateCommentRate 0.2
set notGetDigestRate 0.2
set connectNomalNodeRate 0.25

# ノード
set rootNode ""
set gateNode(0,0) ""
set semiGateNode(0,0) ""
set digestNode(0,0) ""
set nomalDigestNode(0,0) ""
set nomalNotDigestNode(0,0) ""
set joinNode(0,0) ""
set replaceGateNode(0,0) ""
set replaceSemiGateNode(0,0) ""
set replaceDigestNode(0,0) ""

# ノードの数
set digestNodeNum 0
set gateNodeNum 0
set semiGateNodeNum 0
set nomalNodeNum  0
set notGetDigestNomalNum 0
set getDigestNomalNum 0
set joinNodeNum 0
set replaceGateNodeNum 0
set replaceSemiGateNodeNum 0
set replaceDigestNodeNum 0

# ノードリスト
set nodeList(0) ""
set nodeListForBandwidth(0) ""
set joinNodeList(0) ""
set replaceGateNodeList(0) ""
set replaceSemiGateNodeList(0) ""
set replaceDigestNodeList(0) ""

# 各ノードの種類のリスト
# {node1: "digest", node2: "gate"...}
set gateNodeTypeList(0) ""
set semiGateNodeTypeList(0) ""
set digestNodeTypeList(0) ""

# 各ノードと代わりのノードリスト
# {node1: replaceNode1, node2: replaceNode2...}
set gateToReplaceList(0) ""
set semiGateToReplaceList(0) ""
set digestToReplaceList(0) ""

# ゲートノードとセミゲートノードのノードリスト
# {node1: replaceNode1, node2: replaceNode2...}
set gateToSemiGateList(0) ""

# 帯域幅ノードリスト(Mbps)
```

```
74 set bandwidthList(0) ""
75
76 # 一時退避帯域幅ノードリスト
77 set temporalBandwidthList(0) ""
78
79 # コメント数ノードリスト
80 set commentList(0) ""
81
82 # ダイジェスト以外のソートされた帯域幅ノードリスト
83 set sortedBandwidthList(0) ""
84
85 # goddardのための変数宣言
86 set goddard(0) ""
87 set gplayer(0) ""
88 # set sfile(0) ""
89 set gCount 0
90
91 # グループ
92 set mproto ""
93 set mrthandlee ""
94 set group ""
95 set udp ""
96 set cbr ""
97 set sfile ""
98 set startTime 0.0
99 set joinLeaveInterval 10.0
100
101 # my-goddardのための関数
102
103 # この中で便宜上一時的に帯域幅リストからノードを削除している
104 proc digestNodeInit {digestNode digestNodeTypeList bandwidthList temporalBandwidthList
      nodeListForBandwidth nodeList userNum clusterNum digestNodeNum} {
105     upvar $digestNode dn $digestNodeTypeList dntl $bandwidthList bl $temporalBandwidthList
          tbl $nodeListForBandwidth nlfb $nodeList nl
106
107     copy bl tbl
108
109     set commentI [expr $userNum-1]
110     for {set i 0} {$i < $clusterNum} {incr i} {
111         for {set j 0} {$j < $digestNodeNum} {incr j} {
112             set dn($i,$j) $nl($commentI)
113             set dntl($nl($commentI)) "digestNode"
114
115             # 帯域幅リストからダイジェストノードを削除
116             array unset bl $nl($commentI)
117
118             # 帯域幅ノードリストからダイジェストノードを削除
119             for {set k 0} {$k < [array size nlfb]} {incr k} {
120                 if {[array get nlfb $k] == []} {
121                     continue
122                 }
123                 if {$nlfb($k) == $nl($commentI)} {
124                     array unset nlfb $k
125                     break
126                 }
127             }
128
129             # ダイジェストノードの色
130             $dn($i,$j) color yellow
131
132             decr commentI
133         }
134     }
135     return
136 }
137
138 proc gateNodeInit {gateNode gateNodeTypeList sortedBandwidthList clusterNum gateNodeNum} {
139     upvar $gateNode gn $gateNodeTypeList gntl $sortedBandwidthList sbl
140     set k 0
141     for {set i 0} {$i < $clusterNum} {incr i} {
142         for {set j 0} {$j < $gateNodeNum} {incr j} {
143             set gn($i,$j) $sbl($k)
144             set gntl($sbl($k)) "gateNode"
145
146             # ゲートノードの色
147             $gn($i,$j) color #006400
148
149             incr k
150         }
151     }
152     return
153 }
154
155 proc semiGateNodeInit {semiGateNode semiGateNodeTypeList gateToSemiGateList
      sortedBandwidthList gateNode clusterNum semiGateNodeNum} {
156     upvar $semiGateNode sgn $semiGateNodeTypeList sgntl $gateToSemiGateList gtsgl
          $sortedBandwidthList sbl $gateNode gn
```

```
157     set k [array size gn]
158     for {set i 0} {$i < $clusterNum} {incr i} {
159         for {set j 0} {$j < $semiGateNodeNum} {incr j} {
160             set sgn($i,$j) $sbl($k)
161             set sgntl($sbl($k)) "semiGateNode"
162             set gtsgl($gn($i,$j)) $sgn($i,$j)
163 
164             # セミゲートノードの色
165             $sgn($i,$j) color #00ff00
166 
167             incr k
168         }
169     }
170 
171     return
172 }
173 
174 proc nomalNodeInit {nomalNotDigestNode nomalDigestNode gateNode semiGateNode
     sortedBandwidthList clusterNum notGetDigestNomalNum getDigestNomalNum} {
175     upvar $nomalNotDigestNode nndn $nomalDigestNode ndn $gateNode gn $semiGateNode sgn
         $sortedBandwidthList sbl
176 
177     set k [expr [array size gn] + [array size sgn]]
178     for {set i 0} {$i < $clusterNum} {incr i} {
179         for {set j 0} {$j < $notGetDigestNomalNum} {incr j} {
180             set nndn($i,$j) $sbl($k)
181 
182             # ダイジェスト未取得ノーマルノードの色
183             $nndn($i,$j) color pink
184 
185             incr k
186         }
187     }
188 
189     for {set i 0} {$i < $clusterNum} {incr i} {
190         for {set j 0} {$j < $getDigestNomalNum} {incr j} {
191             set ndn($i,$j) $sbl($k)
192 
193             # ダイジェスト取得ノーマルノードの色
194             $ndn($i,$j) color orange
195 
196             incr k
197         }
198     }
199 
200     # 残りのノードは全てダイジェスト取得済みノーマルノードへ
201     set limit [expr [array size sbl]-$k]
202 
203     for {set i 0} {$i < $limit} {incr i} {
204         set ndn($i,$getDigestNomalNum) $sbl($k)
205 
206         # ダイジェスト取得ノーマルノードの色
207         $ndn($i,$getDigestNomalNum) color orange
208 
209         incr k
210     }
211     return
212 }
213 
214 proc joinNodeInit {joinNode joinNodeList bandwidthRatio clusterNum finishTime} {
215     upvar $joinNode jn $joinNodeList jnl
216 
217     set k 0
218     for {set i 0} {$i < $clusterNum} {incr i} {
219         for {set j 0} {$j < [expr ($finishTime / 10)]} {incr j} {
220             set jn($i,$j) $jnl($k)
221 
222             # 新規参加ノードの色
223             $jn($i,$j) color #800080
224 
225             incr k
226         }
227     }
228 }
229 
230 proc replaceGateNodeInit {gateNode gateToReplaceList replaceGateNode replaceGateNodeList
     clusterNum gateNodeNum} {
231     upvar $gateNode gn $gateToReplaceList gtrl $replaceGateNode rgn $replaceGateNodeList
         rgnl
232     set k 0
233     for {set i 0} {$i < $clusterNum} {incr i} {
234         for {set j 0} {$j < $gateNodeNum} {incr j} {
235             set rgn($i,$j) $rgnl($k)
236             set gtrl($gn($i,$j)) $rgn($i,$j)
237 
238             # 代わりのゲートノードの色
239             $rgn($i,$j) color #0000ff
```

```
240
241             incr k
242         }
243     }
244 }
245
246 proc replaceSemiGateNodeInit {semiGateNode semiGateToReplaceList replaceSemiGateNode
      replaceSemiGateNodeList clusterNum semiGateNodeNum} {
247     upvar $semiGateNode sgn $semiGateToReplaceList sgtrl $replaceSemiGateNode rsgn
          $replaceSemiGateNodeList rsgnl
248     set k 0
249     for {set i 0} {$i < $clusterNum} {incr i} {
250         for {set j 0} {$j < $semiGateNodeNum} {incr j} {
251             set rsgn($i,$j) $rsgnl($k)
252             set sgtrl($sgn($i,$j)) $rsgn($i,$j)
253
254             # 代わりのセミゲートノードの色
255             $rsgn($i,$j) color #4169e1
256
257             incr k
258         }
259     }
260 }
261
262 proc replaceDigestNodeInit {digestNode digestToReplaceList replaceDigestNode
      replaceDigestNodeList clusterNum digestNodeNum} {
263     upvar $digestNode dn $digestToReplaceList dtrl $replaceDigestNode rdn
          $replaceDigestNodeList rdnl
264     set k 0
265     for {set i 0} {$i < $clusterNum} {incr i} {
266         for {set j 0} {$j < $digestNodeNum} {incr j} {
267             set rdn($i,$j) $rdnl($k)
268             set dtrl($dn($i,$j)) $rdn($i,$j)
269
270             # 代わりのダイジェストノードの色
271             $rdn($i,$j) color #00bfff
272
273             incr k
274         }
275     }
276 }
277
278
279 # ノード間の接続
280 # 常に低いノード側の帯域幅で接続
281
282 # 帯域幅の設定する必要あり
283 proc connectGateNodeInCluster {gateNode semiGateNode replaceGateNode bandwidthList
      rootNode ns gateNodeNum clusterNum selfClusterNum} {
284     upvar $gateNode gn $semiGateNode sgn $replaceGateNode rgn $bandwidthList bl
285     # 配信者ノード
286     for {set i 0} {$i < $gateNodeNum} {incr i} {
287         $ns duplex-link $gn($selfClusterNum,$i) $rootNode $bl($gn($selfClusterNum,$i))Mb
              500ms DropTail
288     }
289
290     # ゲートノード同士：1 → 2   2 → 3   3 → 1
291     for {set i 0} {$i < $gateNodeNum} {incr i} {
292         if { [expr $i+1] >= $gateNodeNum } {
293             set bandwidth [returnLowBandwidth bl $gn($selfClusterNum,$i) $gn(
                  $selfClusterNum,[expr $i+1-$gateNodeNum])]
294             $ns duplex-link $gn($selfClusterNum,$i) $gn($selfClusterNum,[expr $i+1
                  -$gateNodeNum]) [expr $bandwidth]Mb 100ms DropTail
295         } else {
296             set bandwidth [returnLowBandwidth bl $gn($selfClusterNum,$i) $gn(
                  $selfClusterNum,[expr $i+1])]
297             $ns duplex-link $gn($selfClusterNum,$i) $gn($selfClusterNum,[expr $i+1]) [expr
                   $bandwidth]Mb 100ms DropTail
298         }
299     }
300
301     # ゲートノードとセミゲートノード
302     for {set i 0} {$i < $gateNodeNum} {incr i} {
303         set bandwidth [returnLowBandwidth bl $gn($selfClusterNum,$i) $sgn(
              $selfClusterNum,$i)]
304         $ns duplex-link $gn($selfClusterNum,$i) $sgn($selfClusterNum,$i) [expr $bandwidth]
              Mb 100ms DropTail
305
306         # 代わりのゲートノードの帯域幅設定
307         set bl($rgn($selfClusterNum,$i)) $bl($sgn($selfClusterNum,$i))
308     }
309 }
310
311 proc connectGateNodeOutside {gateNode bandwidthList nsArg clusterNum gateNodeNum
      selfIndexNum} {
312     upvar $gateNode gn $bandwidthList bl $nsArg ns
313     # クラスタ外のゲートノード同士：1 → 2   2 → 3 ...  7→ 1
```

```
314     for {set i 0} {$i < $clusterNum} {incr i} {
315         if { [expr $i+1] >= $clusterNum } {
316             set bandwidth [returnLowBandwidth bl $gn($i,$selfIndexNum) $gn([expr $i+1
                    -$clusterNum],$selfIndexNum)]
317             $ns duplex-link $gn($i,$selfIndexNum) $gn([expr $i+1-$clusterNum]
                    ,$selfIndexNum) [expr $bandwidth]Mb 100ms DropTail
318         } else {
319             set bandwidth [returnLowBandwidth bl $gn($i,$selfIndexNum) $gn([expr $i+1]
                    ,$selfIndexNum)]
320             $ns duplex-link $gn($i,$selfIndexNum) $gn([expr $i+1],$selfIndexNum) [expr
                    $bandwidth]Mb 100ms DropTail
321         }
322     }
323 }
324
325 proc connectSemiGateNode {semiGateNode digestNode nomalDigestNode nomalNotDigestNode
      bandwidthList ns clusterNum semiGateNodeNum notGetDigestNomalNum getDigestNomalNum
      selfIndexNum } {
326     upvar $semiGateNode sgn $digestNode dn $nomalDigestNode ndn $nomalNotDigestNode nndn
          $bandwidthList bl
327     # ダイジェストノード
328     for {set i 0} {$i < $semiGateNodeNum}  {incr i} {
329         if {[array get dn $selfIndexNum,[expr $i*2]] == []} {
330                 continue
331         }
332         set bandwidth [returnLowBandwidth bl $sgn($selfIndexNum,$i) $dn($selfIndexNum,[
              expr $i*2])]
333         $ns duplex-link $sgn($selfIndexNum,$i) $dn($selfIndexNum,[expr $i*2]) [expr
              $bandwidth]Mb 100ms DropTail
334         if {[array get dn $selfIndexNum,[expr $i*2+1]] == []} {
335                 continue
336         }
337         set bandwidth [returnLowBandwidth bl $sgn($selfIndexNum,$i) $dn($selfIndexNum,[
              expr $i*2+1])]
338         $ns duplex-link $sgn($selfIndexNum,$i) $dn($selfIndexNum,[expr $i*2+1]) [expr
              $bandwidth]Mb 100ms DropTail
339     }
340
341     # ノーマルノード
342     for {set i 0} {$i < $semiGateNodeNum}  {incr i} {
343         set digestBorderNum [expr int(($notGetDigestNomalNum+$getDigestNomalNum)*rand())]
344         if {$digestBorderNum >= $notGetDigestNomalNum} {
345             set bandwidth [returnLowBandwidth bl $sgn($selfIndexNum,$i) $ndn(
                    $selfIndexNum,[expr $digestBorderNum-$notGetDigestNomalNum])]
346             $ns duplex-link $sgn($selfIndexNum,$i) $ndn($selfIndexNum,[expr
                    $digestBorderNum-$notGetDigestNomalNum]) [expr $bandwidth]Mb 100ms
                    DropTail
347         } else {
348             set bandwidth [returnLowBandwidth bl $sgn($selfIndexNum,$i) $nndn(
                    $selfIndexNum,$digestBorderNum)]
349             $ns duplex-link $sgn($selfIndexNum,$i) $nndn($selfIndexNum,$digestBorderNum) [
                    expr $bandwidth]Mb 100ms DropTail
350         }
351     }
352 }
353
354 proc connectDigestNode {digestNode nomalNotDigestNode bandwidthList ns
      notGetDigestNomalNum getDigestNomalNum digestNodeNum selfIndexNum } {
355     upvar $digestNode dn $nomalNotDigestNode nndn $bandwidthList bl
356     # ダイジェスト未取得ノーマルノード
357     for {set i 0} {$i < $digestNodeNum}  {incr i} {
358         for {set j 0} {$j < $notGetDigestNomalNum} {incr j} {
359             set bandwidth [returnLowBandwidth bl $dn($selfIndexNum,$i) $nndn(
                    $selfIndexNum,$j)]
360             $ns duplex-link $dn($selfIndexNum,$i) $nndn($selfIndexNum,$j) [expr $bandwidth
                    ]Mb 100ms DropTail
361         }
362     }
363 }
364
365 proc connectNomalNode {nomalDigestNode nomalNotDigestNode bandwidthList ns clusterNum
      connectNomalNodeRate notGetDigestNomalNum getDigestNomalNum nomalNodeNum selfIndexNum
        } {
366     upvar $nomalDigestNode ndn $nomalNotDigestNode nndn $bandwidthList bl
367
368     # とりあえずリストに全部入れる
369     for {set i 0} {$i < [expr $nomalNodeNum+1]} {incr i} {
370         if {$i >= $notGetDigestNomalNum} {
371             if {[array get ndn $selfIndexNum,[expr $i-$notGetDigestNomalNum]] == []} {
                    continue }
372             set nomalNodeList($i) $ndn($selfIndexNum,[expr $i-$notGetDigestNomalNum])
373         } else {
374             set nomalNodeList($i) $nndn($selfIndexNum,$i)
375         }
376     }
377
378     # 適当な回数リストの中身をシャッフル
379     set temp ""
```

```
380     for {set i 0} {$i < 100 } {incr i} {
381         set randomNum1 [expr int(($nomalNodeNum)*rand())]
382         set randomNum2 [expr int(($nomalNodeNum)*rand())]
383         set temp $nomalNodeList($randomNum1)
384         set $nomalNodeList($randomNum1) $nomalNodeList($randomNum2)
385         set $nomalNodeList($randomNum2) $temp
386     }
387
388     set connectNomalNum [expr int(ceil($nomalNodeNum*$connectNomalNodeRate))]
389
390     # ノーマルノード同士：0 → 1   0 → 2   0 → 3   0 → 4 、1 → 2   1 → 3
            ...14 → 15   14 → 0   14 → 1   14 → 2
391     for {set i 0} {$i < [expr $nomalNodeNum+1]} {incr i} {
392         for {set j 0} {$j < $connectNomalNum} {incr j} {
393             if { [expr $i+$j+1] >= $nomalNodeNum } {
394                 if {[array get nomalNodeList $i] == []} { continue }
395                 set bandwidth [returnLowBandwidth bl $nomalNodeList($i) $nomalNodeList([
                        expr $i+$j+1-$nomalNodeNum])]
396                 $ns duplex-link $nomalNodeList($i) $nomalNodeList([expr $i+$j+1
                        -$nomalNodeNum]) [expr $bandwidth]Mb 100ms DropTail
397             } else {
398                 set bandwidth [returnLowBandwidth bl $nomalNodeList($i) $nomalNodeList([
                        expr $i+$j+1])]
399                 $ns duplex-link $nomalNodeList($i) $nomalNodeList([expr $i+$j+1]) [expr
                        $bandwidth]Mb 100ms DropTail
400             }
401         }
402     }
403 }
404
405 proc connectJoinNode {joinNode nomalDigestNode nomalNotDigestNode bandwidthList ns
        clusterNum joinNodeNum nomalNodeNum selfIndexNum} {
406     upvar $joinNode jn $nomalDigestNode ndn $nomalNotDigestNode nndn $bandwidthList bl
407     # joinNodeとnomalDigestNode
408     set i 0
409     while {[array get ndn $selfIndexNum,$i] != []} {
410         if {[array get jn $selfIndexNum,$i] == []} {
411             break
412         }
413         $ns duplex-link $jn($selfIndexNum,$i) $ndn($selfIndexNum,$i) $bl($ndn(
                $selfIndexNum,$i))Mb 100ms DropTail
414         incr i
415     }
416     # joinNodeとnomalNotDigestNode
417     set j 0
418     while {[array get nndn $selfIndexNum,$j] != [] && [array get jn $selfIndexNum,$i] !=
            []} {
419         $ns duplex-link $jn($selfIndexNum,$i) $nndn($selfIndexNum,$j) $bl($ndn(
                $selfIndexNum,$j))Mb 100ms DropTail
420         incr i
421     }
422
423     set connectJoinNum 4
424     for {set i 0 } {$i < [expr $joinNodeNum - 1]} {incr i} {
425         set jList($i) $jn($selfIndexNum,$i)
426     }
427
428     # joinNode同士
429     for {set i 0} {$i < [expr $joinNodeNum-2]} {incr i} {
430         for {set j 0} {$j < $connectJoinNum} {incr j} {
431             if { [expr $i+$j+1] >= $joinNodeNum } {
432                 if {[array get jList $i] == []} { continue }
433                 $ns duplex-link $jList($i) $jList([expr int($i+$j+1-$joinNodeNum)]) 1.0Mb
                        100ms DropTail
434             } else {
435                 if {$jList($i) == $jList([expr $i+$j])} {
436                     $ns duplex-link $jList($i) $jList([expr $i+$j+1]) 1.0Mb 100ms DropTail
437                 } else {
438                     $ns duplex-link $jList($i) $jList([expr $i+$j]) 1.0Mb 100ms DropTail
439                 }
440             }
441         }
442     }
443 }
444
445 # 代わりのゲートノード
446 proc connectReplaceGateNodeInCluster {gateNode semiGateNode replaceGateNode bandwidthList
        rootNode ns gateNodeNum clusterNum selfClusterNum} {
447     upvar $gateNode gn $semiGateNode sgn $replaceGateNode rgn $bandwidthList bl
448     # 配信者ノード
449     for {set i 0} {$i < $gateNodeNum} {incr i} {
450         $ns duplex-link $rgn($selfClusterNum,$i) $rootNode $bl($rgn($selfClusterNum,$i))Mb
                500ms DropTail
451     }
452
453     # ゲートノード同士：1 → 2   2 → 3   3 → 1
454     for {set i 0} {$i < $gateNodeNum} {incr i} {
```

```
455         if { [expr $i+1] >= $gateNodeNum } {
456             set bandwidth [returnLowBandwidth bl $rgn($selfClusterNum,$i) $gn(
                    $selfClusterNum,[expr $i+1-$gateNodeNum])]
457             $ns duplex-link $rgn($selfClusterNum,$i) $gn($selfClusterNum,[expr $i+1
                    -$gateNodeNum]) [expr $bandwidth]Mb 100ms DropTail
458         } else {
459             set bandwidth [returnLowBandwidth bl $rgn($selfClusterNum,$i) $gn(
                    $selfClusterNum,[expr $i+1])]
460             $ns duplex-link $rgn($selfClusterNum,$i) $gn($selfClusterNum,[expr $i+1]) [
                    expr $bandwidth]Mb 100ms DropTail
461         }
462     }
463
464     # ゲートノードとセミゲートノード
465     for {set i 0} {$i < $gateNodeNum} {incr i} {
466         set bandwidth [returnLowBandwidth bl $rgn($selfClusterNum,$i) $sgn(
                $selfClusterNum,$i)]
467         $ns duplex-link $rgn($selfClusterNum,$i) $sgn($selfClusterNum,$i) [expr $bandwidth
                ]Mb 100ms DropTail
468     }
469 }
470
471 # 代わりのゲートノード
472 proc connectReplaceGateNodeOutside {gateNode replaceGateNode bandwidthList nsArg
        clusterNum gateNodeNum selfIndexNum} {
473     upvar $gateNode gn $replaceGateNode rgn $bandwidthList bl $nsArg ns
474     # クラスタ外のゲートノード同士：1 → 2  2 → 3 ... 7→ 1
475     for {set i 0} {$i < $clusterNum} {incr i} {
476         if { [expr $i+1] >= $clusterNum } {
477             set bandwidth [returnLowBandwidth bl $rgn($i,$selfIndexNum) $gn([expr $i+1
                    -$clusterNum],$selfIndexNum)]
478             $ns duplex-link $rgn($i,$selfIndexNum) $gn([expr $i+1-$clusterNum]
                    ,$selfIndexNum) [expr $bandwidth]Mb 100ms DropTail
479         } else {
480             set bandwidth [returnLowBandwidth bl $rgn($i,$selfIndexNum) $gn([expr $i+1]
                    ,$selfIndexNum)]
481             $ns duplex-link $rgn($i,$selfIndexNum) $gn([expr $i+1],$selfIndexNum) [expr
                    $bandwidth]Mb 100ms DropTail
482         }
483     }
484 }
485
486 # 代わりのセミゲートノード
487 proc replaceSemiGateNodeBandwitdhSetting {nomalDigestNode replaceSemiGateNode
        sortedBandwidthList bandwidthList clusterNum} {
488     upvar $nomalDigestNode ndn $replaceSemiGateNode rsgn $sortedBandwidthList sbl
            $bandwidthList bl
489
490     for {set i 0} {$i < $clusterNum} {incr i} {
491         set rsgnI 0
492         for {set j 0} {$j < [expr [array size sbl] - 1]} {incr j} {
493             set k 0
494             while { [array get ndn $i,$k] != []} {
495                 if {[array get rsgn $i,$rsgnI] == []} { break }
496                 if {$ndn($i,$k) == $sbl($j)} {
497                     set bl($rsgn($i,$rsgnI)) $ndn($i,$k)
498                     incr rsgnI
499                 }
500                 incr k
501             }
502         }
503     }
504 }
505
506 proc connectReplaceSemiGateNode {semiGateNode digestNode nomalDigestNode
        nomalNotDigestNode replaceSemiGateNode bandwidthList ns clusterNum semiGateNodeNum
        notGetDigestNomalNum getDigestNomalNum selfIndexNum } {
507     upvar $semiGateNode sgn $digestNode dn $nomalDigestNode ndn $nomalNotDigestNode nndn
            $bandwidthList bl $replaceSemiGateNode rsgn
508     # ダイジェストノード
509     for {set i 0} {$i < $semiGateNodeNum}  {incr i} {
510         if {[array get dn $selfIndexNum,[expr $i*2]] == []} {
511                 continue
512         }
513         set bandwidth [returnLowBandwidth bl $rsgn($selfIndexNum,$i) $dn($selfIndexNum,[
                expr $i*2])]
514         $ns duplex-link $rsgn($selfIndexNum,$i) $dn($selfIndexNum,[expr $i*2]) [expr
                $bandwidth]Mb 100ms DropTail
515         if {[array get dn $selfIndexNum,[expr $i*2+1]] == []} {
516                 continue
517         }
518         set bandwidth [returnLowBandwidth bl $rsgn($selfIndexNum,$i) $dn($selfIndexNum,[
                expr $i*2+1])]
519         $ns duplex-link $rsgn($selfIndexNum,$i) $dn($selfIndexNum,[expr $i*2+1]) [expr
                $bandwidth]Mb 100ms DropTail
520     }
521
522     # ノーマルノード
```

```
523     for {set i 0} {$i < $semiGateNodeNum}  {incr i} {
524         set digestBorderNum [expr int(($notGetDigestNomalNum+$getDigestNomalNum)*rand())]
525         if {$digestBorderNum >= $notGetDigestNomalNum} {
526             set bandwidth [returnLowBandwidth bl $rsgn($selfIndexNum,$i) $ndn(
                    $selfIndexNum,[expr $digestBorderNum-$notGetDigestNomalNum])]
527             $ns duplex-link $rsgn($selfIndexNum,$i) $ndn($selfIndexNum,[expr
                    $digestBorderNum-$notGetDigestNomalNum]) [expr $bandwidth]Mb 100ms
                    DropTail
528         } else {
529             set bandwidth [returnLowBandwidth bl $rsgn($selfIndexNum,$i) $nndn(
                    $selfIndexNum,$digestBorderNum)]
530             $ns duplex-link $rsgn($selfIndexNum,$i) $nndn($selfIndexNum,$digestBorderNum)
                    [expr $bandwidth]Mb 100ms DropTail
531         }
532     }
533 }
534
535 # 代わりのダイジェストノード
536 proc replaceDigestNodeBandwitdhSetting {nomalDigestNode replaceDigestNode commentList
        bandwidthList clusterNum} {
537     upvar $nomalDigestNode ndn $replaceDigestNode rdn $commentList cl $bandwidthList bl
538     set i 0
539     set num [expr [array size cl] - 1]
540     foreach {index val} [array get cl] {
541         set tcl($num) $index
542         decr num
543     }
544
545     for {set i 0} {$i < $clusterNum} {incr i} {
546         set rdnI 0
547         for {set j 0} {$j < [expr [array size tcl] - 1]} {incr j} {
548             set k 0
549             while { [array get ndn $i,$k] != []} {
550                 if {[array get rdn $i,$rdnI] == []} { break }
551                 if {$ndn($i,$k) == $tcl($j)} {
552                     set bl($rdn($i,$rdnI)) $ndn($i,$k)
553                     incr rdnI
554                 }
555                 incr k
556             }
557         }
558     }
559 }
560
561 proc connectReplaceDigestNode {digestNode nomalNotDigestNode replaceDigestNode
        bandwidthList ns notGetDigestNomalNum getDigestNomalNum digestNodeNum selfIndexNum }
        {
562     upvar $digestNode dn $nomalNotDigestNode nndn $bandwidthList bl $replaceDigestNode rdn
563     # ダイジェスト未取得ノーマルノード
564     for {set i 0} {$i < $digestNodeNum}  {incr i} {
565         for {set j 0} {$j < $notGetDigestNomalNum} {incr j} {
566             set bandwidth [returnLowBandwidth bl $rdn($selfIndexNum,$i) $nndn(
                    $selfIndexNum,$j)]
567             $ns duplex-link $rdn($selfIndexNum,$i) $nndn($selfIndexNum,$j) [expr
                    $bandwidth]Mb 100ms DropTail
568         }
569     }
570 }
571
572 proc UDPStreamInit {mproto mrthandle group udp cbr rcvr sfile ns rootNode } {
573     upvar $mproto mp $mrthandle mh $group g $udp u $cbr c $rcvr r $sfile sf
574     set mp DM
575     set mh [$ns mrtproto $mp {}]
576     set g [Node allocaddr]
577     set u [new Agent/UDP]
578     $u set dst_addr_ $g
579     $u set dst_port_ 0
580     $u set class_ 1
581
582     set sf [open stream-udp.tr w]
583     $ns attach-agent $rootNode $u
584     set c [new Application/Traffic/CBR]
585     $c attach-agent $u
586
587     set r [new Agent/LossMonitor]
588 }
589
590 proc attachInit {nodeList joinNodeList replaceGateNodeList replaceSemiGateNodeList
        replaceDigestNodeList startTime ns rcvr group} {
591     upvar $nodeList nl $joinNodeList jnl $replaceGateNodeList rgnl
            $replaceSemiGateNodeList rsgnl $replaceDigestNodeList rdnl $startTime st
592     set st 0.0
593     for {set i 0} {$i < [expr [array size nl] - 1]} {incr i} {
594         $ns attach-agent $nl($i) $rcvr
595         $ns at $st "$nl($i) join-group $rcvr $group"
596         set st [expr $st + 0.01]
597     }
```

```
598
599     for {set i 0} {$i < [expr [array size rgnl] - 1]} {incr i} {
600         $ns attach-agent $rgnl($i) $rcvr
601         $ns at $st "$rgnl($i) join-group $rcvr $group"
602         set st [expr $st + 0.01]
603         $ns at $st "$rgnl($i) leave-group $rcvr $group"
604         set st [expr $st + 0.01]
605     }
606
607     for {set i 0} {$i < [expr [array size rsgnl] - 1]} {incr i} {
608         $ns attach-agent $rsgnl($i) $rcvr
609         $ns at $st "$rsgnl($i) join-group $rcvr $group"
610         set st [expr $st + 0.01]
611         $ns at $st "$rsgnl($i) leave-group $rcvr $group"
612         set st [expr $st + 0.01]
613     }
614
615     for {set i 0} {$i < [expr [array size rdnl] - 1]} {incr i} {
616         $ns attach-agent $rdnl($i) $rcvr
617         $ns at $st "$rdnl($i) join-group $rcvr $group"
618         set st [expr $st + 0.01]
619         $ns at $st "$rdnl($i) leave-group $rcvr $group"
620         set st [expr $st + 0.01]
621     }
622 }
623
624 # 参加時は n規定の時刻+0.01の時
625 proc newJoin {joinNodeList ns rcvr group startTime finishTime joinLeaveInterval} {
626     upvar $joinNodeList jnl
627     set joinTime [expr $startTime + 0.01]
628     copy jnl nl
629     set num [expr [array size jnl] - 1]
630     for {set i 0} {$i < [expr $num*5]} {incr i} {
631         set temp1 [expr int($num*rand())]
632         set temp2 [expr int($num*rand())]
633         set tempNode $nl($temp1)
634         set nl($temp1) $nl($temp2)
635         set nl($temp2) $tempNode
636     }
637     set k 0
638     while {$joinTime < $finishTime} {
639         $ns at $joinTime "$nl($k) join-group $rcvr $group"
640         set joinTime [expr $joinTime + $joinLeaveInterval]
641         incr k
642     }
643 }
644
645 # 離脱時は n規定の時刻+0.02の時
646 # 続けて代わりのノードが参加する時は+0.03の時
647 proc leaveNode {nodeList joinNodeList replaceGateNodeList replaceSemiGateNodeList
      replaceDigestNodeList gateNodeTypeList semiGateNodeTypeList digestNodeTypeList
      gateToReplaceList semiGateToReplaceList digestToReplaceList gateToSemiGateList
      commentList startTime finishTime joinLeaveInterval ns rcvr group} {
648     upvar $nodeList nl $joinNodeList jnl $replaceGateNodeList rgnl
          $replaceSemiGateNodeList rsgnl $replaceDigestNodeList rdnl $gateNodeTypeList gntl
          $semiGateNodeTypeList sgntl $digestNodeTypeList dntl $gateToReplaceList gtrl
          $semiGateToReplaceList sgtrl $digestToReplaceList dtrl $gateToSemiGateList gtsgl
          $commentList cl
649     set num [expr [array size nl] - 1]
650     copy nl tnl
651
652     copy cl tcl
653
654     set selectNum 0
655     set num [expr [array size tnl]]
656     set loopNum [expr $finishTime / $joinLeaveInterval]
657     while {$selectNum < $loopNum} {
658         set rand1 [expr int(($num - 1)*rand())]
659         if {[array get tnl $rand1] == []} {
660             continue
661         }
662
663         if {[array get tcl $tnl($rand1)] == []} {
664             continue
665         }
666
667         set border [expr 1000*rand()]
668         if {$tcl($tnl($rand1)) == 25} {
669             if {$border > 990} {
670                 set list($selectNum) $tnl($rand1)
671                 array unset tnl $rand1
672                 incr selectNum
673             }
674         } elseif {$tcl($tnl($rand1)) == 22} {
675             if {$border > 960} {
676                 set list($selectNum) $tnl($rand1)
677                 array unset tnl $rand1
```

```
678                 incr selectNum
679             }
680         } elseif {$tcl($tnl($rand1)) == 20} {
681             if {$border > 930} {
682                 set list($selectNum) $tnl($rand1)
683                 array unset tnl $rand1
684                 incr selectNum
685             }
686         } elseif {$tcl($tnl($rand1)) == 17} {
687             if {$border > 900} {
688                 set list($selectNum) $tnl($rand1)
689                 array unset tnl $rand1
690                 incr selectNum
691             }
692         } elseif {$tcl($tnl($rand1)) == 15} {
693             if {$border > 800} {
694                 set list($selectNum) $tnl($rand1)
695                 array unset tnl $rand1
696                 incr selectNum
697             }
698         } elseif {$tcl($tnl($rand1)) == 12} {
699             if {$border > 500} {
700                 set list($selectNum) $tnl($rand1)
701                 array unset tnl $rand1
702                 incr selectNum
703             }
704         } elseif {$tcl($tnl($rand1)) == 10} {
705             if {$border > 400} {
706                 set list($selectNum) $tnl($rand1)
707                 array unset tnl $rand1
708                 incr selectNum
709             }
710         } elseif {$tcl($tnl($rand1)) == 7} {
711             if {$border > 300} {
712                 set list($selectNum) $tnl($rand1)
713                 array unset tnl $rand1
714                 incr selectNum
715             }
716         } elseif {$tcl($tnl($rand1)) == 5} {
717             if {$border > 100} {
718                 set list($selectNum) $tnl($rand1)
719                 array unset tnl $rand1
720                 incr selectNum
721             }
722         } elseif {$tcl($tnl($rand1)) == 2} {
723             if {$border > 300} {
724                 set list($selectNum) $tnl($rand1)
725                 array unset tnl $rand1
726                 incr selectNum
727             }
728         }
729     }
730
731     set k 0
732     set leaveTime [expr $startTime + 0.02]
733     while {$leaveTime < $finishTime} {
734         # digestNodeだったら
735         if {[array get dntl $list($k)] != []} {
736             if {$dntl($list($k)) == "digestNode"} {
737                 $ns at $leaveTime "$list($k) leave-group $rcvr $group"
738                 $ns at [expr $leaveTime + 0.01] "$dtrl($list($k)) join-group $rcvr $group"
739             }
740         }
741
742         if {[array get sgntl $list($k)] != []} {
743             if {$sgntl($list($k)) == "semiGateNode"} {
744                 $ns at $leaveTime "$list($k) leave-group $rcvr $group"
745                 $ns at [expr $leaveTime + 0.01] "$sgtrl($list($k)) join-group $rcvr $group
                        "
746             }
747         }
748
749         if {[array get gntl $list($k)] != []} {
750             if {$gntl($list($k)) == "gateNode"} {
751                 $ns at $leaveTime "$list($k) leave-group $rcvr $group"
752                 $ns at [expr $leaveTime + 0.01] "$gtrl($list($k)) join-group $rcvr $group"
753
754                 $ns at [expr $leaveTime + 0.02] "$gtsgl($list($k)) leave-group $rcvr
                        $group"
755                 $ns at [expr $leaveTime + 0.03] "$sgtrl($gtsgl($list($k))) join-group
                        $rcvr $group"
756             }
757         }
758
759         set leaveTime [expr $leaveTime + $joinLeaveInterval]
760         incr k
761     }
```

```
762 }
763
764 #Define a 'finish' procedure
765 proc finish {} {
766     global ns f sfile userNum
767     $ns flush-trace
768
769     set awkCode {
770         {
771             if ($5 == "cbr") {
772                 if ($2 >= t_end_udp) {
773                     tput_udp = bytes_udp * 8 / ($2 - t_start_udp)/1000;
774                     print $2, tput_udp >> "tput-udp.tr";
775                     t_start_udp = $2;
776                     t_end_udp   = $2 + 2;
777                     bytes_udp = 0;
778                 }
779                 if ($1 == "r") {
780                     bytes_udp += $6;
781                 }
782             }
783         }
784     }
785
786     if { [info exists sfile] } {
787         close $sfile
788     }
789
790     close $f
791
792     exec rm -f tput-tcp.tr tput-udp.tr
793     exec touch tput-tcp.tr tput-udp.tr
794     exec awk $awkCode out.tr
795     exec cp out.tr [append outTrName "out" $userNum ".tr"]
796     exec cp tput-udp.tr [append tputUdpName "tput-udp" $userNum ".tr"]
797     exec xgraph -bb -tk -m -x Seconds -y "Throughput (kbps)" tput-udp.tr &
798     exec nam out.nam &
799     exit 0
800 }
801
802 ## 処理開始
803
804 setPacketColor $ns
805 set clusterNum 1
806 setNodeNum digestNodeNum gateNodeNum semiGateNodeNum nomalNodeNum notGetDigestNomalNum
        getDigestNomalNum joinNodeNum $userNum $clusterNum $digestUserRate $gateCommentRate
        $semiGateCommentRate $gateCommentRate $semiGateNodeNum $notGetDigestRate $finishTime
807
808 puts "1クラスタ当たりのノードの数\n"
809 puts "ダイジェストノード: \t\t\t$digestNodeNum"
810 puts "ゲートノード: \t\t\t\t$gateNodeNum"
811 puts "セミゲートノード: \t\t\t$semiGateNodeNum"
812 puts "ノーマルノード: \t\t\t$nomalNodeNum"
813 puts "ダイジェスト未取得ノーマルノード: \t$notGetDigestNomalNum"
814 puts "ダイジェスト取得済みノーマルノード: \t$getDigestNomalNum"
815
816 ratioSetting bandwidthRatio commentRatio $clusterNum $userNum
817
818 # 各ノードリストのinit処理
819 nodeListInit nodeList nodeListForBandwidth joinNodeList replaceGateNodeList
        replaceSemiGateNodeList replaceDigestNodeList $ns $userNum $finishTime $clusterNum
        $digestNodeNum $gateNodeNum $semiGateNodeNum $joinLeaveInterval
820 bandwidthListInit bandwidthList bandwidthRatio nodeListForBandwidth $ns $userNum
821 commentListInit commentList commentRatio nodeList $ns $userNum
822 nodeListForBandwidthShuffle nodeListForBandwidth $userNum
823
824
825 # 各役割のinit処理
826 rootNodeInit rootNode $ns
827 digestNodeInit digestNode digestNodeTypeList bandwidthList temporalBandwidthList
        nodeListForBandwidth nodeList $userNum $clusterNum $digestNodeNum
828 sortBandwidthList sortedBandwidthList bandwidthRatio bandwidthList
829 gateNodeInit gateNode gateNodeTypeList sortedBandwidthList $clusterNum $gateNodeNum
830 semiGateNodeInit semiGateNode semiGateNodeTypeList gateToSemiGateList sortedBandwidthList
        gateNode $clusterNum $semiGateNodeNum
831 nomalNodeInit nomalNotDigestNode nomalDigestNode gateNode semiGateNode sortedBandwidthList
        $clusterNum $notGetDigestNomalNum $getDigestNomalNum
832 joinNodeInit joinNode joinNodeList bandwidthRatio $clusterNum $finishTime
833 replaceGateNodeInit gateNode gateToReplaceList replaceGateNode replaceGateNodeList
        $clusterNum $gateNodeNum
834 replaceSemiGateNodeInit semiGateNode semiGateToReplaceList replaceSemiGateNode
        replaceSemiGateNodeList $clusterNum $semiGateNodeNum
835 replaceDigestNodeInit digestNode digestToReplaceList replaceDigestNode
        replaceDigestNodeList $clusterNum $digestNodeNum
836
837 puts "\nノードの数\n"
838 puts "ダイジェストノード: \t\t\t[array size digestNode]"
```

```
839 puts "ゲートノード: \t\t\t\t[array size gateNode]"
840 puts "セミゲートノード: \t\t\t[array size semiGateNode]"
841 puts "ダイジェスト未取得ノーマルノード: \t[array size nomalNotDigestNode]"
842 puts "ダイジェスト取得済みノーマルノード: \t[array size nomalDigestNode]"
843
844 # トレースファイルの設定
845 set f [open out.tr w]
846 $ns trace-all $f
847 set nf [open out.nam w]
848 $ns namtrace-all $nf
849
850 # 一時的にノードを削除していたので帯域幅リストを元に戻す
851 copy temporalBandwidthList bandwidthList
852
853 # クラスタ内部接続(クラスタの数実行)
854 for {set i 0} {$i < $clusterNum} {incr i} {
855     connectGateNodeInCluster gateNode semiGateNode replaceGateNode bandwidthList $rootNode
            $ns $gateNodeNum $clusterNum $i
856     connectSemiGateNode semiGateNode digestNode nomalDigestNode nomalNotDigestNode
            bandwidthList $ns $clusterNum $semiGateNodeNum $notGetDigestNomalNum
            $getDigestNomalNum $i
857     connectDigestNode digestNode nomalNotDigestNode bandwidthList $ns
            $notGetDigestNomalNum $getDigestNomalNum $digestNodeNum $i
858     connectNomalNode nomalDigestNode nomalNotDigestNode bandwidthList $ns $clusterNum
            $connectNomalNodeRate $notGetDigestNomalNum $getDigestNomalNum $nomalNodeNum $i
859     connectJoinNode joinNode nomalDigestNode nomalNotDigestNode bandwidthList $ns
            $clusterNum $joinNodeNum $nomalNodeNum $i
860
861     # 代わりゲートのノード接続
862     connectReplaceGateNodeInCluster gateNode semiGateNode replaceGateNode bandwidthList
            $rootNode $ns $gateNodeNum $clusterNum $i
863 }
864
865 # 代わりのセミゲートノードの帯域幅設定
866 replaceSemiGateNodeBandwitdhSetting nomalDigestNode replaceSemiGateNode
        sortedBandwidthList bandwidthList $clusterNum
867
868 # 代わりのダイジェストノードの帯域幅設定
869 replaceDigestNodeBandwitdhSetting nomalDigestNode replaceDigestNode commentList
        bandwidthList $clusterNum
870
871 # クラスタ内部接続(クラスタの数実行, 代わりのセミゲートノードと代わりのダイジェストノード)
872 for {set i 0} {$i < $clusterNum} {incr i} {
873     connectReplaceSemiGateNode semiGateNode digestNode nomalDigestNode nomalNotDigestNode
            replaceSemiGateNode bandwidthList $ns $clusterNum $semiGateNodeNum
            $notGetDigestNomalNum $getDigestNomalNum $i
874     connectReplaceDigestNode digestNode nomalNotDigestNode replaceDigestNode bandwidthList
            $ns $notGetDigestNomalNum $getDigestNomalNum $digestNodeNum $i
875 }
876
877 # udp通信
878 UDPStreamInit mproto mrthandle group udp cbr rcvr sfile $ns $rootNode
879 attachInit nodeList joinNodeList replaceGateNodeList replaceSemiGateNodeList
        replaceDigestNodeList startTime $ns $rcvr $group
880
881 # 終了時間設定
882 set finishTime [expr $startTime + $finishTime]
883
884 # 新規参加ノード設定
885 newJoin joinNodeList $ns $rcvr $group $startTime $finishTime $joinLeaveInterval
886
887 # 離脱ノード設定
888 leaveNode nodeList joinNodeList replaceGateNodeList replaceSemiGateNodeList
        replaceDigestNodeList gateNodeTypeList semiGateNodeTypeList digestNodeTypeList
        gateToReplaceList semiGateToReplaceList digestToReplaceList gateToSemiGateList
        commentList $startTime $finishTime $joinLeaveInterval $ns $rcvr $group
889
890 $ns at $startTime "$cbr start"
891
892 $ns at $finishTime "finish"
893
894 $ns run
```

A.2.2 外部プロシージャコード

プログラム A.6 外部プロシージャコード

```
1 # 処理のための一般的なメソッド
2
3 proc decr { int { n 1 } } {
4     if { [ catch {
```

```
5           uplevel incr $int -$n
6       } err ] } {
7           return -code error "decr: $err"
8       }
9       return [ uplevel set $int ]
10  }
11
12  # 配列をコピー
13  proc copy {ary1 ary2} {
14      upvar $ary1 from $ary2 to
15      foreach {index value} [array get from *] {
16          set to($index) $value
17      }
18  }
19
20  # 低い方の帯域幅を返す
21  proc returnLowBandwidth {bandwidthList node1 node2} {
22      upvar $bandwidthList bl
23      if { $bl($node1) >= $bl($node2) } {
24          return $bl($node2)
25      } else {
26          return $bl($node1)
27      }
28  }
29
30  # ここからシミュレータの処理
31
32  # パケットの色設定
33  proc setPacketColor { ns } {
34      $ns color 0 blue
35      $ns color 1 red
36      $ns color 2 white
37  }
38
39  proc setNodeNum {digestNodeNum gateNodeNum semiGateNodeNum nomalNodeNum
        notGetDigestNomalNum getDigestNomalNum joinNodeNum userNum clusterNum digestUserRate
        gateCommentRate semiGateCommentRate gateCommentRate semiGateNodeNum notGetDigestRate
        finishTime} {
40      upvar $digestNodeNum dnn $gateNodeNum gnn $semiGateNodeNum sgnn $nomalNodeNum nnn
            $notGetDigestNomalNum ngdn $getDigestNomalNum gdnn $joinNodeNum jnn
41      set dnn [expr int(ceil([expr $userNum / $clusterNum * $digestUserRate]))]
42      set gnn [expr int(ceil([expr $userNum / $clusterNum * $gateCommentRate]))]
43      set sgnn [expr int(ceil([expr $userNum / $clusterNum * ($semiGateCommentRate -
            $gateCommentRate)]))]
44      set nnn [expr $userNum / $clusterNum - $dnn - $gnn - $sgnn]
45      set ngdn [expr int(ceil([expr $nnn * $notGetDigestRate]))]
46      set gdnn [expr $nnn - $ngdn]
47      set jnn [expr $finishTime / 10]
48  }
49
50  proc ratioSetting {bandwidthRatio commentRatio clusterNum userNum} {
51      upvar $bandwidthRatio br $commentRatio cr
52
53      set basicRatio [expr $userNum/$userNum]
54      foreach {index val} [array get br] {
55          set tempBandwidthRatio($index) [expr $val*$basicRatio]
56      }
57      copy tempBandwidthRatio br
58
59      foreach {index val} [array get cr] {
60          set tempCommentRatio($index) [expr $val*$basicRatio]
61      }
62      copy tempCommentRatio cr
63  }
64
65  proc nodeListInit {nodeList nodeListForBandwidth joinNodeList replaceGateNodeList
        replaceSemiGateNodeList replaceDigestNodeList ns userNum finishTime clusterNum
        digestNodeNum gateNodeNum semiGateNodeNum joinLeaveInterval} {
66      upvar $nodeList nl $nodeListForBandwidth nlfb $joinNodeList jnl $replaceGateNodeList
            rgnl $replaceSemiGateNodeList rsgnl $replaceDigestNodeList rdnl
67
68      # 既存ノード
69      for {set i 0} {$i < $userNum} {incr i} {
70          set nl($i) [$ns node]
71          set nlfb($i) $nl($i)
72      }
73
74
75      # 新規参加ノード
76      for {set i 0} {$i < [expr $finishTime * $clusterNum / $joinLeaveInterval]} {incr i} {
77          set jnl($i) [$ns node]
78      }
79
80      # 代わりのノード
81      # ゲートノード
82      for {set i 0} {$i < [expr $gateNodeNum*$clusterNum]} {incr i} {
83          set rgnl($i) [$ns node]
```

```
84      }
85
86      # セミゲートノード
87      for {set i 0} {$i < [expr $semiGateNodeNum*$clusterNum]} {incr i} {
88          set rsgnl($i) [$ns node]
89      }
90
91      # ダイジェストノード
92      for {set i 0} {$i < [expr $digestNodeNum*$clusterNum]} {incr i} {
93          set rdnl($i) [$ns node]
94      }
95  }
96
97  proc bandwidthListInit {bandwidthList bandwidthRatio nodeListForBandwidth ns userNum} {
98      upvar $bandwidthList bl $bandwidthRatio br $nodeListForBandwidth nlfb
99      set j 0
100     foreach {index val} [array get br] {
101         for {set i 0} {$i < $val} {incr i} {
102             set bl($nlfb($j)) $index
103             incr j
104         }
105     }
106 }
107
108 proc commentListInit {commentList commentRatio nodeList ns userNum } {
109     upvar $commentList cl $commentRatio cr $nodeList nl
110     set j 0
111     foreach {index val} [array get cr] {
112         for {set i 0} {$i < $val} {incr i} {
113             set cl($nl($j)) $index
114             incr j
115         }
116     }
117 }
118
119 proc nodeListForBandwidthShuffle {nodeListForBandwidth userNum} {
120     upvar $nodeListForBandwidth nlfb
121     for {set i 0} {$i < [expr $userNum*5]} {incr i} {
122         set temp1 [expr int($userNum*rand())]
123         set temp2 [expr int($userNum*rand())]
124         set tempNode $nlfb($temp1)
125         set nlfb($temp1) $nlfb($temp2)
126         set nlfb($temp2) $tempNode
127     }
128 }
129
130 proc rootNodeInit {rootNode ns} {
131     upvar $rootNode rn
132     set rn [$ns node]
133     # 配信者ノードの色
134     $rn color red
135 }
136
137
138 proc sortBandwidthList {sortedBandwidthList bandwidthRatio bandwidthList} {
139     upvar $sortedBandwidthList sbl $bandwidthRatio br $bandwidthList bl
140
141     # 帯域幅の種類のリスト
142     set i 0
143     foreach val [lsort -real [array names br]] {
144         set kindOfBandwidthList($i) $val
145         incr i
146     }
147
148     set k 0
149     for {set i [expr [array size kindOfBandwidthList]-1]} {$i >= 0} {decr i} {
150         foreach {index val} [array get bl] {
151             if {$val == $kindOfBandwidthList($i)} {
152                 set sbl($k) $index
153                 incr k
154             }
155         }
156     }
157 }
158
159 proc createUDP {udp cbr sfile udpCount ns l_node r_node group} {
160     upvar $udp u $cbr c $sfile sf $udpCount uc
161     set u($uc) [new Agent/UDP]
162     set sf($uc) [open stream-udp.tr w]
163     $u($uc) attach $sf($uc)
164     $ns attach-agent $l_node $u($uc)
165     $u($uc) set dst_addr_ $group
166     $u($uc) set dst_port_ 0
167     set c($uc) [new Application/Traffic/CBR]
168     $c($uc) attach-agent $u($uc)
169
170     incr uc
```

```
171 }
172
173 proc createNomalNodeUDPStream {nomalDigestNode nomalNotDigestNode digestNode udp cbr sfile
        udpCount joinNodeList rootNode ns clusterNum getDigestNomalNum notGetDigestNomalNum
      digestNodeNum group} {
174     upvar $nomalDigestNode ndn $nomalNotDigestNode nndn $digestNode dn $udp u $cbr c
            $sfile sf $udpCount uc $joinNodeList jnl
175     for {set i 0} {$i < $clusterNum} {incr i} {
176         for {set j 0} {$j < $getDigestNomalNum} {incr j} {
177             createUDP u c sf uc $ns $rootNode $ndn($i,$j) $group
178         }
179         for {set j 0} {$j < $notGetDigestNomalNum} {incr j} {
180             createUDP u c sf uc $ns $rootNode $nndn($i,$j) $group
181         }
182         for {set j 0} {$j < $digestNodeNum} {incr j} {
183             createUDP u c sf uc $ns $rootNode $dn($i,$j) $group
184         }
185     }
186 }
```

A.2.3 デフォルトパラメータコード

プログラム A.7 デフォルトパラメータコード

```
1  # 実験時間
2  set finishTime 60.0
3
4  # 帯域幅割合
5  array set bandwidthRatio {
6      3.000 4
7      1.500 1
8      1.024 6
9      0.768 2
10     0.640 1
11     0.512 1
12     0.448 4
13     0.384 2
14     0.320 4
15     0.256 3
16 }
17
18 # コメント数割合
19 array set commentRatio {
20     25 1
21     22 1
22     20 1
23     17 1
24     15 3
25     12 3
26     10 4
27     7 5
28     5 5
29     2 4
30 }
```