社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2006－DPS－128 (10)
2006－GN－ 61 (10)
2006－EIP－ 33 (10)
2006／9／15

アプリケーション層マルチキャストミドルウェアの実装とPlanetLab上での評価

池田 和史 Thilmee M. Baduge 梅津 高朗 山口 弘純 東野 輝夫
大阪大学 大学院情報科学研究科
{k-ikeda,thilmee,umedu,h-yamagu,higashino}@ist.osaka-u.ac.jp

本稿では，ストリーミングを用いるアプリケーション層マルチキャスト（ALM）アプリケーションの開発，性能評価および運用を支援するための汎用的なミドルウェアを提案する．提案ミドルウェアは多くのALMプロトコルの開発に必要なトポロジ管理機能，通信支援機能，マルチメディア支援機能を提供することで，アプリケーション開発にかかる労力と時間を軽減する．また，パフォーマンスモニタリング機能やPlanetLab上での実行支援機能を用いて，開発したアプリケーションの実環境上での性能評価や試験を容易に行うことができる．提案ミドルウェアを用いて既存の分散型ALMプロトコルをPlanetLab上で実装・性能改善することで，その利便性を確認した．

Middleware Supporting Development of Application Layer Multicast Applications and its Evaluation on PlanetLab

Kazushi Ikeda Thilmee M. Baduge Takaaki Umedu
Hirozumi Yamaguchi Teruo Higashino
Graduate School of Information Science and Technology, Osaka University

In this paper, we present a middleware to support the development, evaluation and maintenance of application layer multicast (ALM) applications. The middleware facilitates the development of ALM applications by introducing topology administration, communication supporting and multimedia supporting facilities which are necessary for the development of most ALM applications. Furthermore, the middleware is equipped with performance monitoring and experiment-supporting functions on PlanetLab, which make it easy to examine and maintain the system. We have implemented an existing decentralized ALM protocol using the middleware and confirmed its usability. Also some experiments have been carried out on PlanetLab to show that the developed middleware could achieve reasonable performance to execute the protocol.

1 まえがき

近年インターネットの普及に伴い，電子ビデオ会議，ファイル交換，チャットやマルチメディア配信等のインターネットを介したサービスへの需要が高まってきている．このようなサービスを実現するための有用な通信方法としてアプリケーション層マルチキャスト（以下単にALM）が近年注目を集めている[1, 2, 3, 4, 5, 6, 9, 7]．しかし，実プラットフォームでの実装や実験環境整備の煩雑さなどから，先行研究の多くはシミュレーション実験による性能評価に留まっている場合が多い．文献[4, 8, 9]などでは，実環境での実証実験も報告されているが，個々の実装には多くの労力が必要であると共に，実装の方法に依存して実ネットワーク上でのプロトコルの性能に差が出ると考えられ，プロトコル間の公正な比較評価を行うのが困難であるなどの問題がある．

これらの問題を解決するためには，なるべく多くのALMアプリケーションの実装に利用できる汎用的なミドルウェアの開発が重要であると考えられる．多くのALMで用いられる汎用的な機能を備えるだけでなく，スループットや遅延測定機能，アプリケーションのデバッグ機能や遠隔操作機能，トポロジ表示機能等を提供することで，ALMアプリケーションの開発が容易になるだけではなく，実際に稼働した時にボトルネックユーザを容易に特定できるなど運営上でのメリットも大きい．

そこで我々はこれらの機能を備えたアプリケーション層マルチキャスト向けミドルウェアを提供する．さらに，ALMを用いるアプリケーションの実行試験を容易に行うための環境もあわせて提供する．ここでは世界中に配置されているサーバからなるPlanetLab上での実装実験が容易に行える機能を，我々の研究グループが開発しているPlanetLab用デバッガ[10]を拡張することで提供する．また，本ミドルウェアにおけるマルチメディア送受信はJMF(Java Media Framwork)[11]を利用し，マルチメディアストリーミング向けの通信プロトコルであるRTP/RTCPを用いて行う．

なお，ALMの実装事例である文献[4, 9, 13]の他に，ネットワーク状況に応じて動的にトランスコーディングを行うミドルウェア[12]や限られたネットワーク帯域を有効に利用するためにリアルタイムストリームを共有するミドルウェア[14]などがあるが，いずれもALMアプリケーションの開発および運用支援を目指したものではない．一方，文献[15]で提案されているミドルウェアはP2Pプロトコル開発に利用できる汎用APIを提案している点で本ミドルウェアと類似しているが，遠隔実行やデバッグ機能を含む開発支援機能は提供していない．

2 ミドルウェアの設計

2.1 ALMアプリケーションの特性とALMプロトコル

現実的なALMアプリケーションとしては数人～十数人からなるチャット，電子ビデオ会議，数百人～数万人からなるファイル交換，インターネットゲーム，マルチメディア配信，遠隔講義システムが挙げられる．これらのアプリケーションの性質を規模，遅延，帯域の観点から分類したものを表1に

アプリケーション	規模	帯域制約	遅延制約	配信元
チャット	小	小	大	複数
電子ビデオ会議	小	大	大	複数
ファイル交換	大	小	小	複数
インターネットゲーム	大	中	大	複数
マルチメディア配信	大	大	中	単一
遠隔講義システム	大	大	中	単一

表 1: ALM アプリケーションの性質

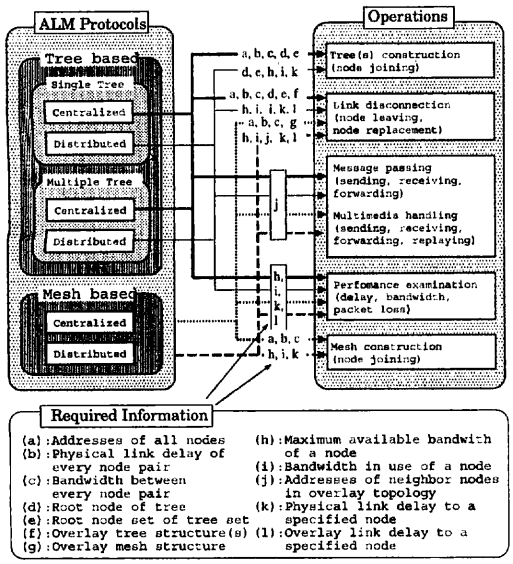

図 1: ALM プロトコルの分類

示す.

これらの ALM アプリケーションの特性に基づき，多くの異なる ALM プロトコルが考案されている．既存の ALM プロトコルにおけるオーバレイネットワークのトポロジと配信木の数，及び制御方式に基づく分類と，それらプロトコルで実行する基本的なオペレーションおよびそのために必要な情報を図 1 に示す．提案ミドルウェアはこれらの性質を十分考慮し，幅広い汎用性を実現するように設計する．

2.2 提案ミドルウェアの機能

図 2 に提案ミドルウェアの概要を示す．本ミドルウェアが提供する機能は大まかに以下の 5 つに分類できる．

（1） トポロジ管理機能
（2） 基本通信支援機能
（3） マルチメディア支援機能
（4） パフォーマンスモニタリング機能
（5） PlanetLab 上での実行支援機能

以下ではそれぞれについて順に説明する．

2.2.1 トポロジ管理機能

ALM プロトコルにおいては，アプリケーションに参加しているノード数，それらノードの帯域，ノード間の接続関係及び遅延などオーバレイネットワークに関する様々な情報を用いるものが多い．例えば，アプリケーションの遅延制約のため，途中参加ノードはなるべく送信ノードに近い位置に接続したいといった可能性もある．このような場合に配信木上の各ノードへの送信ノードからの遅延を取得する機能がミドルウェアによって提供されることでプロトコルの開発負荷を低減できる．以下ではオーバレイネットワークを構成するノードを単にノードと呼ぶ．また，トポロジとはオーバレイネットワークのトポロジを意味する．

[共通機能] 図 1 の全プロトコルタイプで必要とする下記の情報を提供する．

- ノードのノード情報：ノードの最大帯域，使用中の帯域（図 1 中の h，i）及びプロトコルで指定するその他のノード固有の情報
- ノードの接続情報：隣接ノード集合のノード ID とネットワークアドレス（図 1 中の j）
- ノードの遅延情報：指定する任意のノードへの物理ネットワーク上での遅延およびオーバレイトポロジ上での遅延（図 1 中の k，l）

これらの情報は，集中制御システムであれば管理ノードに，分散管理システムであればそれぞれのノードに問い合わせることで取得できる．

[集中管理型プロトコル共通機能] 図 1 の単一配信木構造，複数配信木構造とメッシュ構造の集中管理型のプロトコルで必要とする下記の共通情報を提供する．

- ノード集合情報：オーバレイトポロジ上の各ノード ID とネットワークアドレス（図 1 中の a）
- 全ノードペアに関する情報：ノード間の物理ネットワーク上での遅延，ノード間の帯域（図 1 中の b，c）
- 全ノードのログ情報：参加，離脱時刻の履歴および通信履歴のログ

これらの情報は管理ノードに問い合わせることで取得できる．

[その他のプロトコル固有機能] 図 1 の単一配信木構造，複数配信木構造とメッシュ構造の集中管理型のプロトコルにおけるそれぞれに固有な情報を提供する．

- 木構造情報：図 1 の単一木構造と複数木構造を用いるプロトコルで利用できる木のルートノードおよび木構造に関する情報（図 1 中の d，e，f）．但し，複数木の場合にはそれぞれの木に対する情報
- 木の情報：木の最大遅延時間と最大ホップ数、ルートノードからの最大遅延と最大ホップ数
- メッシュ構造情報：図 1 のメッシュ構造を用いるプロトコルで利用できるメッシュ構造に関する情報（図 1 中の g）

これらの情報は管理ノードに問い合わせることで取得できる．

2.2.2 基本通信支援機能

ALM プロトコルの実装の際には，指定するノードへのリンク接続および切断，ノード間の制御メッセージ送受信やデータ転送をを行う必要がある．これに対し，ミドルウェアは下記の機能を提供する．

- オーバレイネットワークリンク構築および切断機能：指定するノードへのオーバレイネットワークリンクを構築，もしくは切断する
- オーバレイリンクを介したメッセージ送受信機能：メッセージを指定するノードへ送信，又は受信する
- メッセージ転送機能：指定する隣接ノード群へ受信したメッセージを転送する
- スケジュール送受信機能：メッセージ送受信を指定されたタイミングで行う

2.2.3 マルチメディア支援機能

表 1 の電子ビデオ会議，マルチメディア配信と遠隔講義システム等の ALM アプリケーションにおいては動画や音声の利用は不可欠であるが，一般にそれら動画や音声の送受信や再生機能の実装はシステムやアプリケーションに大きく依存する．しかし，動画や音声をオーバレイネットワーク上で扱う機能をミドルウェアが提供することで，それらマルチメディアを扱う ALM プロトコルの設計開発において，実際のメディアを扱うプロトタイプ実験が容易に行えるといった利点や，同じフォーマットや符号化のもとででプロトコルの制御機能を比較できるといった利点もある．そこでミドルウェ

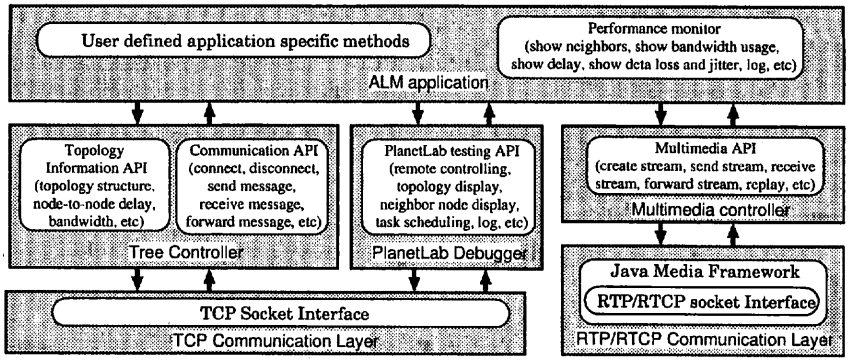

図 2: ミドルウェアの設計概要

アでは，動画，音声等の送受信および再生を容易に行うための下記の機能を提供する．

- メディアストリーム送受信機能：指定するノードへメディアストリームを送信，又は受信する
- メディアストリーム転送機能機能：指定するノード群へ受信したメディアストリームを送信する
- メディアストリーム再生機能：メディアストリームを再生する

なお，ここでの通信はマルチメディア通信向けの通信プロトコルとして知られている RTP/RTCP を利用する．

2.2.4 パフォーマンスモニタリング機能

開発したプロトコルやアプリケーションの性能評価，及び運用上における性能観測は非常に重要である．そのような性能測定を支援する機能として下記の機能を提供する．

- 隣接ノード表示機能：指定したノードの隣接ノードを表示
- 帯域表示機能：指定したノードが直接接続するネットワークの物理帯域および，そのノードが現在利用している帯域のリアルタイム表示
- 遅延表示機能：指定されたノードへのオーバレイネットワークリンクの遅延のリアルタイム表示
- データロス，ジッタ表示機能：受信するマルチメディア情報のパケットロスとジッタのリアルタイム表示
- ログ機能：制御メッセージやデータの送受信履歴，隣接ノードとの接続，切断履歴のログ作成

これらの機能を実現するために必要な情報は 2.2.1 節〜2.2.3 節で述べた機能を用いて取得する．なお，表示は 2.2.5 節で述べるデバッガのグラフィカルな表示機能と連携させることで実現する．

2.2.5 PlanetLab 上での実行支援機能

2.2.1 節〜2.2.4 節までに述べた機能は ALM アプリケーションの容易な実装，性能試験および運用を支援するものである．本節では PlanetLab 上での性能試験を想定し，試験およびデバッグにかかる負荷軽減を目指した機能について述べる．PlanetLab は世界中にサーバが配置されていることから実環境を考慮した性能試験を行うプラットフォームとして非常に有用である．しかし，PlanetLab 上でのアプリケーション実行は個々のサーバを操作しなければならず，特に大規模アプリケーションを想定した試験においては手間が大きい．ミドルウェアはそのような実証実験を支援する機能を提供する．

- ノードタスクスケジューリング機能：各ノードの起動時刻，終了時刻，セッション中に行う動作等の記述に基づき，その時刻に合わせて自動的に該当するタスクを実行する
- ノードの遠隔操作：多数の PlanetLab 端末に手動でログインすることなく，各ノードを一括管理する
- オーバレイトポロジ情報取得：ノード間の帯域，遅延，データロス，ジッタを取得する
- 接続関係表示機能：トポロジ全体を表示し，ノード間の接続関係が容易に把握できるようにする
- ノード入れ換え機能：トポロジ表示画面上でノードの入れ換えを指定できるようにする

なお，ここでの機能は既存の PlanetLab 用のデバッガ [10] の機能を流用または拡張することで実現する．

3 ミドルウェアの実装

ミドルウェアは Java 言語により実装する．2 章で述べた機能を提供するため，ミドルウェアは ALM プロトコル管理用 API，ALM アプリケーションデータ (マルチメディアデータ) 操作用 API，および PlanetLab 上でのプロトコル・アプリケーション実行，操作用 API を提供する形で実装する．以下ではそれぞれの API 実装について説明する．

3.1 ALM プロトコル管理用 API

TCP ソケット通信を用いた，参加，離脱等 ALM プロトコル管理用の API の実装について述べる．

- **ALM ノード**
  ノードの生成は IP アドレスとポート番号からなるノード ID を与えて行う．ノードに対しては制御メッセージの送信指示，指定したノードからの切断，接続状況の取得などを行うことができる．スレッドを利用したタスクスケジューリング機能により，指定した時刻にこれらの処理を実行することもできる．ユーザはこのクラスを継承して，ノードの参加，離脱時の処理やその他の必要な処理を記述することができる．
- **制御メッセージ**
  ミドルウェアが提供する制御メッセージはメッセージを作成したノードのノード ID のみを持つ．ユーザは制御メッセージクラスを継承したクラスに必要な情報を追加して使用する．制御メッセージの送信は送信するメッセージと相手ノードのノード ID を指定して行う．初めてメッセージを送信する場合に TCP 接続を確立し，それ以降はその接続を用いてメッセージの送受信を行う．
  制御メッセージを受信した場合，受信したメッセージと送信ノードのノード ID をプロトコルに通知する．メッセージの送信に失敗した場合や，接続しているノードが離脱した場合にも同様にノード ID を通知する．
- **API の利点**
  ミドルウェアがソケットとノード ID を対応付けて管理することで，ユーザはわずらわしいコネクション設定を意識することなく，メッセージの送受信を行うことが

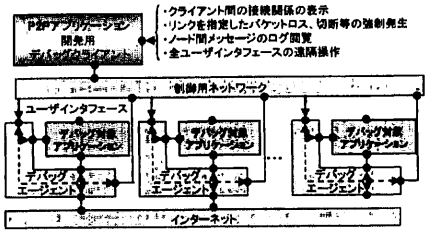

図 3: P2P アプリケーション開発用デバッガの概要

できる．

ノードやノード ID，制御メッセージは継承により，拡張可能なように設計されている．これにより，ミドルウェアは様々なプロトコルに応用できる汎用性を提供する．

3.2 JMF を用いたマルチメディア操作用 API

マルチメディア操作用 API の実装について述べる．

- **JMF(Java Media Framework API)**
  Java で映像や音声を扱うために提供された標準的な API であり，動画ファイルの再生やキャプチャデバイスからの映像の録画などができる．加えて，JMF では RTP/RTCP を利用したストリーム配信もサポートしており，ミドルウェアはこの機能を利用して，ストリームの送受信および転送を実現している．
- **マルチメディア配信ノード**
  マルチメディア配信ノードは 3.1 節の ALM ノードを継承する．ALM 管理用の機能に加え，ノードが送受信しているストリームの取得ができる．
- **送受信ストリーム**
  マルチメディア配信に用いられる送信ストリームは動画ファイル名やキャプチャデバイス名を指定して生成するものと，他ノードから受信したストリームを転送するものとがある．送信ストリームに対して，送信対象ノードの追加や削除，送信ストリームのビットレートやフレームレートの取得などを行うことができる．動画ファイルの終わりや送信対象ノードの離脱はイベントとして通知される．

  受信ストリームに対して，ストリームの転送，パケットロスや遅延などの統計情報の取得，バッファサイズの設定などができる．また，PlanetLab のような，ディスプレイがない環境も考慮して，ストリームを視聴するかしないかを選択できるようにしている．
- **API の利点**
  ミドルウェアが送信ストリーム，受信ストリームを実装することで，ユーザは JMF の複雑な API を理解しなくても，マルチメディア配信を行うことができる．マルチメディア配信は送信するオブジェクトと対象ノードを指定するだけで行うことができ，ストリームの受信は受信したストリームと送信ノードのノード ID がイベントとしてプロトコルに通知されるという，制御メッセージの送受信と同様の仕様になっている．これにより，ユーザは制御メッセージを送受信するように，容易にマルチメディアの送受信を行うことができる．

3.3 PlanetLab 上での実行・操作用 API

3.3.1 基盤とするデバッガ

提案フレームワークを用いて実装したプロトコルを PlanetLab 上で効率的に動作検証するため，ミドルウェアを我々の提案している P2P アプリケーション開発用デバッガ [10] と連携させる．デバッガの概要を図 3 に示す．このデバッガでは，デバッグ・実装実験を行う端末群であらかじめデバッグを支援するデバッグエージェントを実行しておく．デバッグクライアントから多数のエージェントを一括して遠隔操作し，デバッグ対象のアプリケーションの配布や開始・終了，GUI の一元的な操作などの機能を開発者に対して提供することで P2P アプリケーションの開発を支援する．デバッグクライアント群からの情報を集約することで，P2P 環境において把握が難しい各ノードごとの接続関係 (スパニングツリーなど) を表示する機能，TCP/UDP ソケットを監視しノード間でのメッセージ (パケット) の送受信を時系列で表示する機能などにより，P2P アプリケーションのデバッグを支援する．

このデバッガでは，一般的なソケット通信を用いた P2P アプリケーションを開発対象としている．ここでは，デバッガを拡張し，JMF 経由で行われる RTP 通信を監視して使用帯域やパケットロス率を取得する機能や，提案ミドルウェアの状態を監視する機能などを持たせる．それらの情報は，ALM で用いられるスパニングツリーなどの接続状態と併せてグラフィカルに表示される．また，バックアップリンク候補などといった，ALM プロトコルの内部状態をデバッガに知らせる API を用意する．ALM プロトコル実装時にそれらの API を適切に呼び出すことで，ある時点で実際に接続・利用されているリンク以外の情報もデバッガから参照することができ，プロトコルの動作の確認に利用できる．

3.4 ALM アプリケーション実装例

ミドルウェアアプリケーション実装例は以下のようである．

- **マルチメディアの送信**
  標準入力から動画ファイル名，対象ノードの IP アドレス，ポート番号を入力してマルチメディア配信を行うプログラムは次のように実装できる．

```
// input <- file address1 port1 [address2 port2 ...]
StringTokenizer st = new StringTokenizer(input);
String filename = st.nextToken()
// マルチメディアソース (almobj) を作成
ALMMultimediaObject almobj
  = new ALMMultimediaObject(filename);
// 対象ノードのノード ID を Vector に読み込む
Vector<ALMMultimediaNodeID> targetNodeIDs
  = new Vector<ALMMultimediaNodeID>();
while(st.hasMoreTokens())
  targetNodeIDs.addElement(new ALMMultimediaNodeID
    (st.nextToken(),Integer.parseInt(st.nextToken())));
// targetNodeIDs に対して，ストリームを配信する
myNode.sendMultimediaObject(targetNodeIDs,almobj);
```

- **マルチメディアの受信，転送**
  他ノードから配信されるマルチメディアはストリームとして受信され，容易に視聴，転送を行うことができる．

```
public void onReceiveStream(ALMReceiveStream stream) {
  // ストリーム受信イベントが呼び出される
  System.err.println("Protocol : Received new stream from "
    + stream.getSenderNodeID().IPAddress + " "
    + stream.getSenderNodeID().port);
  stream.forward(targetNodeIDs);  // 転送
  stream.setVisible(true);        // 視聴
}
```

- **API の継承**
  API を継承して独自のクラスを作成し，使用することができる．以下はチャットプログラムの例である．

```
public class CustomNodeID extends ALMMultimediaNodeID {
  // 必要な情報を追加できる
  public String name;
  public int degree;
  public long joinTime;
  public CustomNodeID(String IPAddress,int port) {
    super(IPAddress,port);
  }
}
class ChatMessage extends ALMControlMessage{
  // 必要な情報を追加できる
  public String comment;
  public ChatMessage(ALMNodeID myNodeID) {
    super(myNodeID);
  }
}
...
public void onReceiveControlMessage
  (ALMNodeID senderNodeID, ALMControlMessage controlMessage){
```

```
if(controlMessage instanceof ChatMessage){
    // 継承したクラスにキャスト
    ChatMessage message = (ChatMessage)controlMessage;
    CustomNodeID speaker = (CustomNodeID)message.myNodeID;
    System.out.println
      (speaker.name + " > "+ message.comment);
  }
...
}
```

4 性能評価

ミドルウェアの実用性を評価するため，ストリーム配信機能や ALM アプリケーションの実装支援に関する実験を PlanetLab 上で行った．

4.1 実験環境

実験環境の詳細を以下に示す．

- 使用端末数：Linux 49，Windows 1
- Windows 端末：Pentium M 1.5GHz, 512MB memory, Windows XP
- Linux 端末：Pentium 3 1.2GHz ～ Pentium 4 3.4GHz, 512MB ～ 3.6GB memory, version 2.6.12-1.1398_FC4.5.planetlab (support@planet-lab.org)
- Java：Windows, Linux 共に JDK1.5
- JMF：Windows, Linux 共に JMF2.1.1e

また，PlanetLab 上の Linux 端末にはディスプレイがないため，ストリーム品質の確認の目的で Windows 端末を用いる．PlanetLab には世界各地から端末が提供されているが，国外の端末に関しては各地域から提供されている端末数に応じて均一に選んだ．使用した端末が所属する地域を表 2 に示す．

表 2: 使用した PlanetLab 端末の所属地域と台数

地域	使用台数
日本国内	7
アジア (日本を除く)	6
ヨーロッパ	18
アメリカ	18
全体	49

4.2 ALM アプリケーション実行性能

この実験ではミドルウェアが ALM アプリケーション実行に十分な性能を提供できるかを調べる．ミドルウェアの実装方法に依存し，次のような原因でジッタやパケットロスによりストリーム品質が低下する可能性がある．

- ノード間でストリームの転送を繰り返し行うため，各ノードが生成するジッタやパケットロスが蓄積する．
- 次数の増加に伴い，ノードの負荷上昇やパケット送出タイミングのずれが発生する．

これらは一般にネットワーク環境に大きく依存するが，十分な帯域があるにも関わらず，そのような現象が起きていないかを調べることで，ミドルウェアの性能を評価する．

ここでは，ストリームを複数回転送した後，最後に Windows 端末で視聴する．転送回数を増加させ，視聴可能な品質を保ったまま転送できる上限の回数を求めることで，ミドルウェアのストリーム転送機能を評価する．転送可能回数は転送に携わるノードが同一 LAN 内のみ，国内のみ，海外におよぶ場合に分けて行った．ストリームを視聴する Windows 端末と同一 LAN 内のノードとして，我々の研究室が PlanetLab に提供している端末を利用し，2 ノード間で繰り返し転送を行った．国内と海外のノードを利用する実験では，N 回転送を行う場合，ランダムに N ノードを選び，そのうちの 1 ノードからストリームの配信を開始し，N ノード間でランダムに N-1 回転送を行った．図 4 に海外ノードを利用したストリーム転送機能評価実験のトポロジ例を示す．実験の結果を図 5 に示す．

この結果から ping 応答時間が大きい遠隔地のノードへの転送ほど，転送可能回数が少ないことが分かる．これはノード間の距離が大きいほど，経路の途中でパケットロスや遅延が発生するためと考えられる．

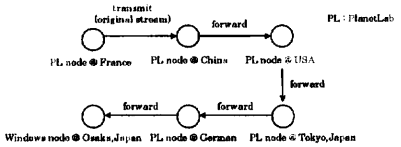

図 4: 海外ノードを利用した転送機能評価実験のトポロジ例 (転送回数 4 回)

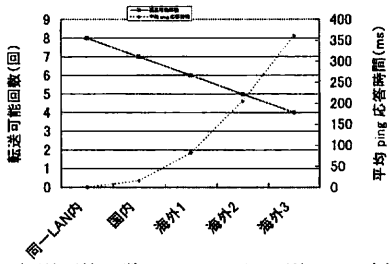

図 5: 転送可能回数とノード間の平均 ping 応答時間

次に，ノードの最大次数を計測する．送信する動画には約 700kbps の MPEG 動画ストリームを用い，受信はランダムに選んだ世界各国のノードと Windows 端末で行い，送信ノードが国内と海外にある場合に分けて計測した．図 6 に最大次数評価実験のトポロジ例を示す．実験結果を表 3 に示す．

この結果から最大次数は 10 以上であることが分かった．また，7Mbps の帯域が得られており，ミドルウェアは十分な性能を提供できているといえる．また，最大次数は送信ノードと受信ノードの所在地には依存しないと考えられる．

4.3 ミドルウェアを利用した ALM アプリケーションの実装

ミドルウェアを利用して ALM アプリケーションを実装し，ミドルウェアの実用性と API 利用による実装支援の効果を調べた．

ALM アプリケーションとして，既存のプロトコル MODE[7] を用いたビデオチャットアプリケーションを実装した．ミドルウェアの提供する制御メッセージの送信や接続ノード取得機能，ストリームの送受信，転送機能を利用することで，容易に実装することができた．また，MODE を用いた既存のチャットアプリケーションプログラムにミドルウェアが提供するストリームの送受信機能を追加する形でも実装を行った．この場合は始めから実装した場合に比べ，さらに容易に実装することができた．変更に要した時間は約 2 時間，約 150 行をプログラムに追加するだけであった．このように，ミドルウェアを利用することで，新たにプロトコルを実装する場合も，既存のプロトコル実装にストリーム配信機能を持たせる場合にも，ミドルウェアは有用であることが分かる．

実装したビデオチャットアプリケーションの動作試験と性能評価を PlanetLab 上の端末を利用して行った．MODE のパラメータとしては次数が考えられる．次数が大きいと帯域を圧迫し，ストリームの品質が低下する．次数が小さいと，転送回数の増加によるストリームの品質低下や転送にかかる

表 3: ノードの所属地域と最大次数

地域	最大次数
日本	12
アジア	10
ヨーロッパ	11
アメリカ	12

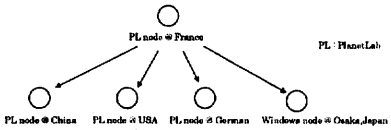

図 6: 最大次数評価実験のトポロジ例 (次数 4)

遅延が大きくなる．ここでは，実環境での性能試験を行うことで，最適なパラメータ設定を求める実験を行う．

次数を 4,8,12 に設定し，それぞれ参加ノード数 50 で実験を行った．木の最も長い経路を流れたストリームを視聴することができれば，木全体でストリームを視聴できると考えられる．実験方法を以下に示す．PlanetLab 上の 49 ノードを参加させ，最後に Windows 端末を参加させて，木を構築する．Windows 端末と PlanetLab 上のあるノード間の経路が最も長くなっていることを確認し，その PlanetLab 上のノードからストリーム配信を行う．ストリームが木全体に転送され，Windows 端末で視聴できるか確認する．MODE を利用したストリーム配信の例を図 7 に示す．

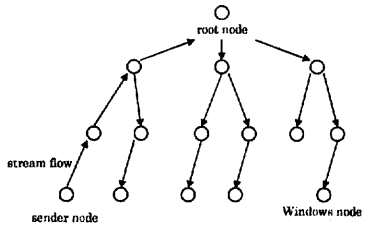

図 7: MODE によるストリーム配信の例 (次数 3，参加ノード数 15)

実験の結果，全ての場合に Windows 端末でストリームを視聴することができたが，最も品質が良かったのは次数 8 の場合であった．次数 12 のときはルートノードのネットワーク負荷が大きかったこと，次数 4 のときは転送回数が 7 回と多くなったことが品質の低下につながったと考えられる．

このようにプロトコルやアプリケーションを実装して実環境でテストを行うことで，最適なパラメータ設定を行ったり，不具合を発見したりすることができる．提案するミドルウェアはプロトコルやアプリケーションの実装支援，PlanetLab 上での実験支援，ストリームの品質情報取得機能などを提供することで，プロトコルやアプリケーションの開発に貢献することができる．

5 まとめと今後の課題

本稿ではストリーミングを用いる ALM アプリケーションの開発，性能評価および運用を支援するための汎用的なミドルウェアを提案した．提案ミドルウェアは多くの ALM プロトコルに共通のトポロジ管理機能や通信支援機能を API として提供することで，アプリケーションの開発にかかる労力を削減する．さらに，パフォーマンスモニタリング機能や PlanetLab 上での実行支援機能を利用することで，開発したアプリケーションの性能評価を実環境で効率的に行うことができる．

今後の課題として，未実装機能の実装，冗長なコードの除去によるミドルウェア性能の向上，より豊富な API の提供などを考えている．より豊富な API を提供するための方法として，既存のプロトコルをさらに詳しく分類し，必要な機能を抽出する．実装が完成したミドルウェアを用いて，既存の様々な ALM プロトコルを実装し，比較実験を行うことで，プロトコルを評価する．実験により判明したプロトコルの長所，短所を元に，新たなプロトコルの開発につなげていくことも可能であると考えている．

参考文献

[1] B. Zhang, S. Jamin and L. Zhang, "Host Multicast: A Framework for Delivering Multicast to End Users," *Proc. of IEEE INFOCOM 2002*, 2002.

[2] S. Shi, J. Turner and M. Waldvogel, "Dimensioning Server Access Bandwidth and Multicast Routing in Overlay Networks," *Proc. of Network and Operating System Support for Digital Audio and Video (NOSSDAV'01)*, pp. 83–91, 2001.

[3] V. Roca and A. El-Sayed, "A Host-Based Multicast (HBM) Solution for Group Communications," *Proc. of 2001 IEEE Int. Conf. on Networking (ICN'01)*, 2001.

[4] D. Pendarakis, S. Shi, D. Verma and M. Waldvogel, "ALMI: An Application Level Multicast Infrastructure," *Proc. of 3rd USENIX Symp. on Internet Technologies and Systems (USITS)*, pp. 49–60, 2001.

[5] Y. H. Chu, S. G. Rao and H. Zhang, "A Case for End System Multicast," *Proc. of ACM SIGMETRICS*, pp. 1–12, 2000.

[6] S. Banerjee, C. Kommareddy, K. Kar, B. Bhattacharjee and S. Khuller, "Construction of an Efficient Overlay Multicast Infrastructure for Real-time Applications," *Proc. of IEEE INFOCOM 2003*, 2003.

[7] T. M. Baduge, 廣森 聡仁, 山口 弘純, 東野 輝夫, "帯域制約のもとで遅延最小のオーバレイマルチキャスト木を構築する分散アルゴリズム," **電子情報通信学会論文誌** *(D-I)*, vol. J88-D-I, no. 11, pp. 1648–1658, 2005.

[8] Y.H. Chu, A. Ganjam, T. S. E. Ng, S.G. Rao, K. Sripanidkulchai, J. Zhan, and H. Zhang, "Early Experience with an Internet Broadcast System Based on Overlay Multicast" *USENIX Annual Technical Conference, Boston, MA* 2004.

[9] T. Yamashita, H. Yamaguchi, K. Yasumoto, T. Higashino and K. Taniguchi, "Emma Middleware: An Application-level Multicast Infrastructure for Multi-party Video Communication," *Proc. of IASTED PDCS2003*, pp. 416–421, 2003

[10] 梅津 高朗 , 池田 直徒 , 東野 輝夫, "P2P アプリケーションの開発と性能評価のための統合開発環境の提案," **情報処理学会論文誌**, Vol.47,No. 7, pp. 2194–2201, 2006

[11] JMF(Java Media Framework API) http://java.sun.com/products/java-media/jmf/

[12] K. Hashimoto and Y. Shibata, "Dynamic transcoding functions by extended media stream," *Proc. of IEEE AINA*, pp. 334–339, 2004

[13] X. Zhang, J. Liu, B. Li and Y. Yum, "CoolStreaming/DONet: a data-driven overlay network for peer-to-peer live media streaming," *Proc. of IEEE INFOCOM*, pp. 2102–2111, 2005.

[14] J. Wul, H. Lien, C. Huang, Y. Shiau and W. Tsai, "An efficienct way for sharing real-time data and the application of remote network video system," *Proc. of IEEE CNNA*, pp. 308–310, 2005.

[15] F. Delmastro, M. Conti and E. Gregori, "P2P Common API for Structured Overlay Networks: A Cross-Layer Extension," *Proc. of IEEE WoWMoM*, pp. 593–597, 2006.