2S-2

透過的なネットワーク環境を実現するグリッドミドルウェア

大迫勇哲 † 山崎航 †† 西山裕之 ‡ 溝口文雄 ‡
† 東京理科大学大学院 理工学研究科経営工学専攻 †† 東京理科大学 総合研究所
‡ 東京理科大学 理工学部経営工学科

1 はじめに

近年の高速ネットワークインフラの普及や計算機の高性能・低価格化などにより，複数の計算機や情報機器を協調させて使用するグリッドコンピューティング (以下，グリッド) が注目されている．しかしながら，複数の異なるプライベートネットワーク上の計算機を統合してグリッドを構築する場合，NAT(Network Address Translation) が問題を引き起こすことがある．NAT は仕組み上，外側から内側 (プライベートネットワーク側) への通信を制限するため，図 1 にあるように，あるプライベートネットワーク上の計算機に対し，上位のパブリックネットワークや他のプライベートネットワーク上の計算機から接続を行うことはできない (NAT 越え問題)．現在，グリッドのリソースとなる計算機は，企業や教育機関，一般家庭等のプライベートネットワーク上に存在しているケースが多く，このような計算機を統合する場合，NAT 越え問題により，各リソース間の透過的な通信が保証されない．既存のグリッドミドルウェアにおける NAT 越え問題への対策は不十分であり，グリッド上で実行されるアプリケーション側やユーザ側で対策を行わなければならなかった．

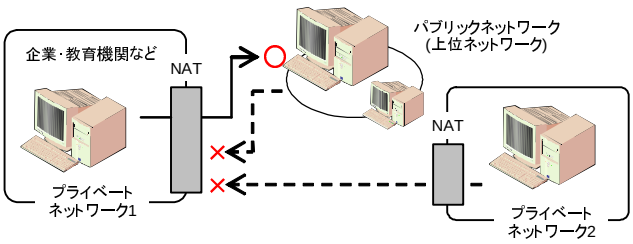

図 1: NAT 越え問題

本論では，グリッド上で実行されるアプリケーションが NAT を意識することなく通信を行える，透過的なネットワーク環境を実現するグリッドミドルウェア LampEye について述べる．LampEye は，各リソース間において，各アプリケーションごとに仮想的なネットワークを構築し，透過的な通信を可能にする．加えて，リソース検索やジョブ処理等の機能も提供する．なお，本論中の”NAT”は，特に断りのない限り，IP アドレスと Port 番号を変換する Network Address/Port Translation(NAPT) も含めて指すものとする．

A Grid Middleware offering transparent network environment. Takenori Ohsako†, Wataru Yamazaki††, Hiroyuki Nishiyama‡, Fumio Mizoguchi‡
{†Graduate School of Science and Technology, ††Research Institute for Science and Technology, ‡Faculty of Science and Technology }, Tokyo University of Science

2 NAT 越え問題の解決手法

NAT に対しポートフォワーディングの設定を行うことで NAT 越え問題の解決は可能であるが，この設定を行うためにはネットワーク管理者権限が必要であり，また，NAT 越えが必要な全ての通信に対して設定を行わなければならない．

LampEye では，以下の 3 つの手法を拡張し，組み合わせることで，NAT に対して特別な設定を行うことなく，ソフトウェアレベルで NAT 越え問題を解決する．

2.1 外部ノードによる中継

通信を行う 2 つのノードが，各 NAT の外側に位置する共に接続可能なノードに通信を中継させる手法である．この手法は NAT 越え手法の中でもっとも一般的であり，かつ，確実である．しかしながら，複数の通信を中継する場合などには，負荷が集中し遅延が発生する可能性がある．また，通信を行う間は常に中継ノードが必要となり，耐故障性の面でも不安がある．

2.2 UDP hole punching

NAT の内側にあるノードのポートに対して割りあてられる NAT の外側のポートが，送信先にかかわらず，一定期間同じであるということを利用して，パケットを NAT の外側から内側へ通過させる．これを利用することで，異なるプライベートネットワーク内のノード間で直接的な通信が行える．送信先ごとに割りあてるポートを変更するタイプの NAT(Symmetric NAT) には適用できないが，既存の約 85 %の NAT に対して適用できる [1].

2.3 UPnP

UPnP[2] に対応した情報機器に対し，メッセージを送信することで操作を行うことができる．この機能を利用し，NAT に対してポートフォワーディングの設定を自動的に行う．UDP hole punching と同様に異なるプライベートネットワーク内のノード間で直接的な通信が行える．ただし，複数の NAT が重なって構築されているネットワークなどには適用が難しい．

3 設計

主に NAT 越え通信機能の設計について述べる．外部ノードによる中継手法は，NAT 越えの確実性という面では優れているが，グリッドで重視される高速性や耐故障性という面では最良の手法とはいえない．

そこで，LampEye では，異なるプライベートネットワーク上のノード間における直接通信が可能な UDP hole punching と UPnP に基づく手法を順に適用し，これらの手法で NAT を越えられない場合のみ中継手法を利用する．

3.1 UDP hole punching の拡張

Symmetric NAT において，新しく割りあてられるポートは，最初に割りあてられたポートからある定数を加減したものになる場合が多い [3]．そこで，その定数を調べ，次に割りあてられるポートを予想することで，LampEye は Symmetric NAT にも対応する．

また，UDP ベースの通信に信頼性を付加するために，データの先頭 5 バイト分を LampEye の通信管理用に使用し，パケット損失等の検出・修復を行う．

3.2 NAT 越え通信

図 2 にあるように，LampEye のノード間通信には，アプリケーションの要求により動的に確立される NAT 越え通信と，各ノードの制御や検索等を行うためのノード制御通信の 2 種類がある．NAT 越え通信の確立後は，ノード制御通信は必ずしも必要ではない．また，セキュリティのために，ノード制御通信の開始時に認証を行い，NAT 越え通信を暗号化することも可能である．

LampEye の各ノードは，最初に接続される側を親，接続する側を子とする相対的な関係を持ち，IP アドレスの代わりに，”子ノード名. 親ノードのパス”というパスか，ノード制御通信の開始時に割り振られる LampEyeID(指定することも可) によって管理・識別される．また，最上位の複数の親ノードは，ゲートウェイノードとして親・子の関係を持たずに相互に接続し，子ノードの情報を共有する．

LampEye の起動後に既知のゲートウェイノードに接続し，指定されたノードとノード制御通信を開始する．この時点で LampEye の仮想ネットワークに接続したことになり，アプリケーション等の要求に応じて，他ノードへの (他ノードからの)NAT 越え通信が行える．

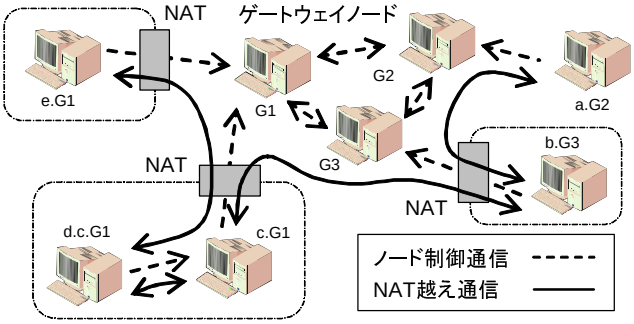

図 2: LampEye ネットワーク

4 実装

LampEye は，Java で実装されている．LampEye の利用に関しては，LampEye のパッケージを使用する方法と GUI を使用する方法の 2 つがある．また，簡易 Web サーバ機能を起動することで，Web ブラウザ等を用いてモニタリングや各ノードの制御等を行える．

NAT 越え通信は，図 3, 4 のように，java.net.Socket や ServerSocket と同様に，lampeye.net.LESocket や LEServerSocket を作成し，java.io.OutputStream と InputStream を取り出すことで可能になる．ただし，接続先の指定には IP アドレスの代わりに，LampEyeID かパスを用いる．この例では，外部ファイルにゲートウェイノードの情報等が記述されている．

具体的な使用方法としては，最も上位のネットワーク上に，ゲートウェイサービスを起動したサーバを最低 1 台設置し，アプリケーションのソケット通信部分のコードを図 3, 4 のように数行変更するだけでよい．

LampEye 上で実行できるジョブは，lampeye.job.Job インターフェイスを実装したもので，任意のノード上で実装された public Object exec() メソッドが実行され，戻り値の Object が結果として回収される．ノードの指定には，LampEyeID を利用する以外にも，ユーザが任意に記述したノード情報をもとに検索し，組み合わせることも可能である．また，実行時に必要となるクラスファイルは，ネットワーククラスローダを通じて自動的に各ノードに配置される．

```
//LampEyeの起動 および ノード制御通信の開始
ResourceManager rm = new ResorceManager();

//待ちうけサーバ起動   6000番で待ちうけ
LEServerSocket ss = new LEServerSocket(rm, 6000);
while(!soc.isClosed){
   //接続されたらソケットを作成
   LESocket soc = ss.accept();
   //ストリーム作成
   InputStream in = soc.getInputStream();
   OutputStream out = soc.getOutputStream();

                        //・・・処理・・・//
}
```

図 3: 待ちうけ側のサンプル

```
//LampEyeの起動 および ノード制御通信の開始
ResourceManager rm = new ResorceManager();

//接続   接続先パス(LampEyeIDでも可):sako.mizo3
LESocket soc = new LESocket(rm, “sako.mizo3”, 6000);
OutputStream out = soc.getOutputStream();
InputStream in = soc.getInputStream();
```

図 4: 接続側のサンプル

5 評価

まず，NAT 越えの確実性という面では，42 の環境で検証を行い，その全てにおいて NAT 越え直接通信に成功した．また，通信性能に関しては，100MB のファイル送信を行い，100BASE-TX 環境のネットワークでは約 87.6Mbps 程度の速度がえられた (暗号化なし．NAT を取り除いて FTP で送信した場合は 86Mbps)．他の環境においてもほぼ各ネットワークの最大の通信速度がえられることを確認した．

6 おわりに

本研究では，NAT 越え問題を解決するためのグリッドミドルウェア LampEye について述べた．LampEye を利用することで，高速で耐故障性に優れ，透過的な通信が行える仮想的なネットワークをグリッドアプリケーションごとに容易に構築することが可能になる．

参考文献

[1] Bryan Ford, Pyda Srisuresh, Dan Kegel. Peer-to-Peer Communication Across Network Address Translators, Proceeding of USENIX 2005, pp.179-192, 2005.

[2] Universal Plug and Play, http://www.upnp.org/

[3] Y. Takeda. IETF MIDCOM WG Internet draft: Symmetric NAT Traversal using STUN, June 2003.