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

A GRID Middleware offering transparent network environment

大迫 勇哲 † 山崎 航 †† 西山 裕之 ††† 溝口 文雄 †††
Takenori OHSAKO Wataru YAMAZAKI Hiroyuki NISHIYAMA Fumio MIZOGUCHI

† 東京理科大学大学院 理工学研究科
Graduate School of Science and Technology, Tokyo University of Science
j7404618@ed.noda.tus.ac.jp
†† 東京理科大学 総合研究所
Research Institute for Science and Technology, Tokyo University of Science
††† 東京理科大学 理工学部
Department of Science and Technology, Tokyo University of Science

複数の異なるプライベートネットワーク上の計算機によって構成されたグリッドにおいて，各計算機間の通信の透過性を容易には実現できない．これは，NAT(Network Address Translator) の外側からプライベートネットワーク側への通信が制限されるためで，この問題は NAT 越え問題として知られている．既存のグリッドミドルウェアは，NAT 越え問題に対する取り組みが不十分であったため，グリッド利用者やアプリケーション側で対策を行う必要があった．本研究では，この問題を解決するために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 の設計と実装を行った．LampEye は，環境や条件に応じて，UDP hole punching 等の既存の NAT 越え手法の中から最適な手法を自動的に選択して NAT 越えを実現する．各手法に対して統一したインターフェイスを提供することで，LampEye 上のアプリケーションは，どの NAT 越え手法をどの通信プロトコルで利用しているのか，ということを意識することなく通信が行える．

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．グリッドは，分散したリソースを統合し，分散処理やアプリケーションの連携，利用者間のコラボレーションなどを実現する．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 Translator) が問題を引き起こすことがある．基本的に，NAT は外側から内側 (プライベートネットワーク側) への通信を制限する．このためプライベートネットワーク上の計算機から他の異なるプライベートネットワーク上の計算機への直接的な通信は行えない (NAT 越え問題)．すなわち，マスタ・ワーカ型のグリッドのように，ある特定の計算機への通信を確立するだけでよい場合はそれほど問題にはならないが，コラボレーションや大量のデータ転送を必要とする P2P 型のグリッドのように，各計算機が互いに通信を行わなければならない場合は，非常に大きな問題となる．現在，グリッドのリソースは各プライベートネットワーク上に存在している場合が多く，また，プライベートネットワーク内に，さらに複数のプライベートネットワークを階層的に構築している場合も少なくない．しかし，この NAT 越え問題に対する既存のグリッドミドルウェア [1, 2] の取り組みは不十分であり，グリッド構築における大きな制約となっている．

NAT 越え問題の解決策としては，NAT にポートフォワーディングの設定を行うことが一般的であるが，ネットワーク管理者権限が必要となり，グリッド利用者が必ずしも行えるとは限らない．そこで，アプリケーションレベルで NAT 越え問題を解決する手法として，(1) NAT の外側に位置する計算機に通信を中継させる (中継方式)，(2) UPnP(Universal Plug and Play)[3]，(3)Hole punching[4]，の 3 つが知られている．しかしながら，これらの手法には 2 章にあるように長所と短所があり，個々の手法では決定的な解決策とはならない．また，各手法をグリッドアプリケーションに実装することは容易ではない．

本研究では，上記の 3 つの手法を組み合わせて NAT 越えを行うことにより，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 の設計と実装を行った．LampEye は，それぞれの環境や条件に応じて最適な NAT 越え手法を自動的に選択し，異なるプライベートネットワーク間での接続を確立する．そして，確立した接続に対して統一したインターフェイスを提供することで，LampEye 上のアプリケーションは，どの NAT 越え手法をどの通信プロトコルで利用しているのか，ということを意識することなく，通信が行える．加えて，LampEye にリソースの管理・検索機能や遠隔計算機上でのジョブ実行機能など実装することで，アプリケーションの作成や実行を容易に行える．

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) を指すとし，”ノード”はグリッドを構成する計算機を示すものとする．

2 既存の NAT 越え手法

本章では，LampEye の NAT 越え機能に用いられている既存の NAT 越え手法を示す．

2.1 中継方式

通信を行う 2 つのノードが，共に接続可能な各 NAT の外側に位置する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を行う必要があり，グリッドにおいて極めて重要な耐故障性という点でも不安がある．通信プロトコルは，一般的に TCP を用いる場合が多い．

2.2 UPnP[3]

UPnP 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．最近の一般向け NAT ルータには，高い割合で実装されているが，UPnP 機能を無効にしている場合がある．また，操作に必要な情報を取得するのに多少時間がかかることがある (長いときは数十秒程度)．通信プロトコルは，設定さえ行えれば TCP でも UDP でも可能である．

2.3 Hole punching[4]

NAT の内側にあるノードのポートに対してマッピングされる NAT の外側のポートが，アドレス変換ルールを NAT が保持している間は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．しかしながら，適用できないタイプの NAT も存在する．通信プロトコルは，UDP でも TCP でも可能であるが，TCP よりも，UDP の方 (UDP hole punching) が適用できる NAT が多く，約 85 %に適用できるという調査結果 [4] もある．また，UDP hole punching の原理を利用した STUN[5](RFC3489) が公開されたため，今後，適用できる NAT が増えると期待される．

UDP hole punching による NAT 越え手法の概要は次の通り．まず，通信を行う通信ノードが，一時的に仲介を行う仲介ノードにそれぞれ接続する．次に，通信ノード間の直接通信に使用する UDP ソケットを作成し，そのソケットを使用して仲介ノードに対し UDP パケットを送信する．仲介ノードは，各通信ノードの UDP ソケットにマッピングされた NAT 上のポートと，NAT のタイプを調べる．最後に，仲介ノードは各通信ノードに対し，相手の UDP ソケットにマッピングされたポートと NAT の IP アドレスを通知し，各通信ノードは通知されたアドレスに対し UDP パケットを送信する．以後，仲介ノード等を介さない直接通信が行える．

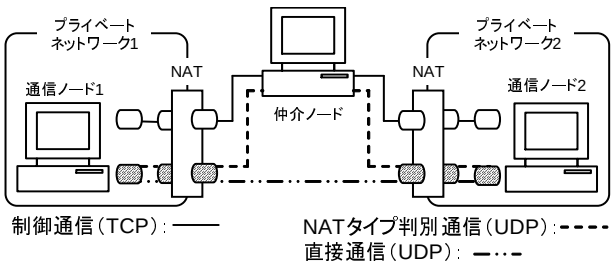

図 1: UDP hole punching による NAT 越えの概要

3 設計

本章では，LampEye の NAT 越え通信機能に加え，ノード (リソース) の管理・検索機能，ノード上でのジョブ実行機能の各設計について述べる．

3.1 設計方針

まず，LampEye の NAT 越え機能に求められることは，確実に NAT を越えること，そして，通信性能をできるだけ損なわないようにすること，である．

次に，ノードの管理・検索機能では，NAT 越えの際に必要な各ノードの協調が容易に行えること，そして，アプリケーションが柔軟にリソースを検索・使用できるように，リソース情報の記述とその検索の自由度を確保すること，である．

最後に，ノード上のジョブ実行機能では，先に確立された NAT 越え通信の接続等を利用できること，そして，OS に非依存なこと，である．

3.2 NAT 越え通信機能

設計方針に従い，確実に NAT を越えるためには，中継方式を選択すべきであるが，2.1 に述べたとおり，条件によっては通信性能を損なう可能性が高い．そこで，優先順位としては，UDP hole punching，UPnP，中継方式とする．なお，各手法におけるプロトコルは，順に，UDP，TCP，TCP とする．

NAT 越えは，通信を行う通信ノードと，これらの通信ノードの仲介を行う仲介ノードの三台一組で行う．仲介ノードは，基本的に，2 台の通信ノードが共に接続可能なノードである．仲介ノードの主な役割は，(1) 通信ノード間のメッセージを中継する，(2) 各通信ノードの環境 (NAT タイプなど) を把握する，(3)NAT 越え通信の方法を決定する，の 3 つである．以下の流れで通信を確立する．

Step1 各通信ノードが仲介ノードに接続する．

Step2 仲介ノードを利用して，各通信ノードの NAT のタイプおよび UPnP 機能の有無を調べる．

Step3 UDP hole punching が適用できるのならば接続して終了．できないのであれば Step4 へ．

Step4 各通信ノードのどちらかの NAT に UPnP 機能があればポートフォワーディングを自動設定し，接続を行い終了．なければ Step5 へ．

Step5 直接通信をあきらめ，他のノードに通信の中継を依頼する (中継方式)．接続ステップを終了．

また，UDP hole punching の場合は，UDP を使用するため，信頼性が低い．そこで，ペイロード内に ID を付加して，パケット損失時には再送するなどの処理を行い，信頼性を付加する．

3.3 ノードの管理・検索機能

まず，3.2 の NAT 越えを行うために必要となる，NAT 越え以前の各ノード間のコミュニケーションやノード検索のための，ブロードキャスト型やユニキャスト型のメッセージ通信が行えるネットワークを構築する．基本的に，アプリケーションの実行時のみ，必要に応じて NAT 越え通信を行う．

具体的には，各ノードは，図 2 のような木構造のネットワークを構築する．各ノード間に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が成立している．なお，接続は TCP で行い，原則として NAT 越えは行わないように接続する (NAT の内側，すなわち，プライベートネットワーク側から外側のノードに接続する)．また，LampEye は，複数の異なるプライベートネットワーク内のノードを統合することを前提にしているため，各ノードを IP アドレスで管理することはできない．そこで，各ノードは，”自分のノード名. 親ノードの ID” という ID で管理される．また，最上位に位置する各ノードは，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．このような構造をとることにより，親あるいは子を知っているだけで，メッセージやデータ等のブロードキャストやユニキャストが可能になる．

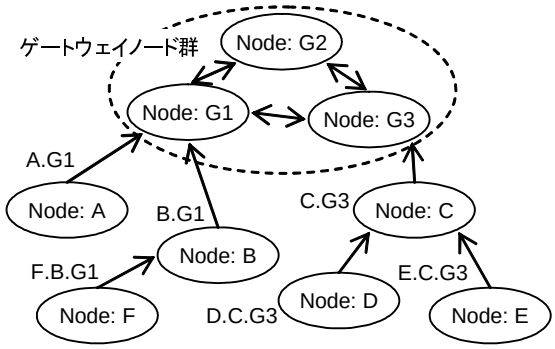

図 2: ノード組織の例

また，ノード情報に関しては，空きメモリなど基本情報に加え，各ノードの管理者が記述した追加情報を検索することができる．情報は型・フィールド名・値の 3 つで構成され，自由に記述できる．

3.4 ノード上でのジョブ実行機能

OS に非依存という点から，LampEye の実装には Java を選択する．これを前提にすると，ノード上でのジョブ (任意のクラス) を実行するには，まず，ジョブインターフェイスを実装したジョブクラスを記述

し，これから生成されたジョブオブジェクトを実行ノードに送信する．次に，ジョブオブジェクト内のジョブ処理用あるいはジョブ起動用のメソッドを呼び出し，実行する．

例えば，単純な分散処理に使用する場合はジョブ処理用のメソッド内に処理の記述を行い，その戻り値を処理結果として取得する．また，実行ノード上でアプリケーションを起動する場合は，ジョブ起動用メソッド内に起動に関する記述を行う．なお，ジョブ起動用のメソッドには，引数として実行ノード上のリソースを参照するためのオブジェクトがセットされており，これを使用することで，先に確立した NAT 越え通信等を利用できる．

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図 3 のような 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ノードの制御等を行える．

4.1 LampEye パッケージ

パッケージを利用することで，NAT 越え通信やジョブの実行等を簡潔に記述することができる．NAT 越え通信に関しては，java.io.OutputStream と java.io.InputStream と同じインターフェイスをもつオブジェクトを利用するため，既存の Java アプリケーションを容易に拡張することができる．

4.2 GUI

Java3D を用いて各ノードを 3D で視覚化しており，この機能を利用してマウス操作で直感的にノードの選択，ジョブの投入等を行うことができる．

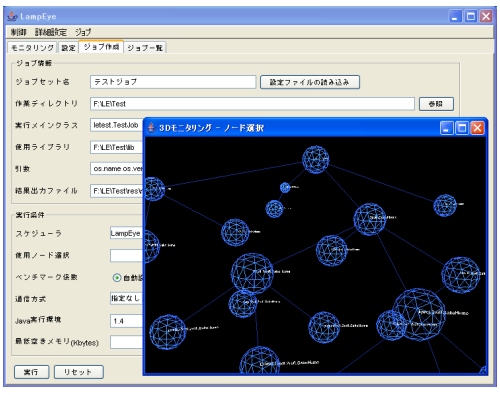

図 3: LampEye ジョブ実行 GUI およびモニタリング

5 評価

5.1 評価実験

本実験では，4 つのプライベートネットワークを含むグリッドにおいて各ノードが同時に通信する場合を想定する．中継方式の場合と UDP hole punching の場合の通信性能を比較することで，NAT 越え手法の選択優先順位の妥当性を評価する．具体的には，(1)100MB のファイルを一方向へ送信する場合と，(2)1000 バイトのメッセージを交互に 1000 回ずつ送信する場合において，実行する通信プログラムの数を変更して，通信性能の変化を比較する．各ノードは図 4 のように接続されており，スペックは，CPU: Pentium4 2.8〜3.0GHz，メモリ: 512MB〜1GB である．また，LAN 環境は 100BASE-TX である．本実験では，B，C と D，E でそれぞれペアをつくり，測定は BC 間で行う．BC 間で起動する通信プログラム数は 1 とし，DE 間の通信プログラムを増加させていく．中継方式の場合は，BC，DE の各ペアがともに A を中継ノードとして使用する．

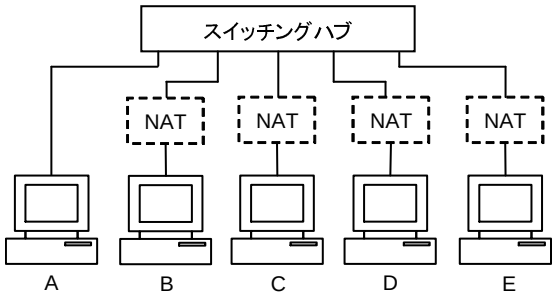

図 4: 評価実験環境

5.2 結果および考察

ファイル送信とメッセージ通信の結果は，それぞれ，図 5，6 のようになった．

まず，ファイル送信に関して述べる．DE 間の通信プログラム数が 0 の時点 (負荷が 0) において，85Mbps 程度の通信速度がでており，基本性能は十分であると考えられる．通信プログラム数が増加した場合，中継方式は，A への負荷が集中するため，通信性能が低下する．一方，UDP hole punching は，A を使用せずに直接通信を行うため，ハブの処理能力を超えない限り，通信性能に影響はない．

次に，メッセージ通信に関して述べる．DE 間の通信プログラム数が 0 の時点における通信時間の差は UDP と TCP の差であると考えられ，UDP に信頼性を付加したプロトコルであっても TCP よりも高速であるといえる．なお，通信プログラム数を増加さ

せた場合でも，中継方式の通信時間が途中までほとんど増加しないのは，通信プログラムが必要とする通信速度が遅く，DE 間の通信プログラムと合わせても，A の帯域幅をすべて使い切れなかったためと考えられる．

以上より，NAT 越え手法の選択において，通信性能を理由に UDP hole punching を優先することは妥当であると考えられる．

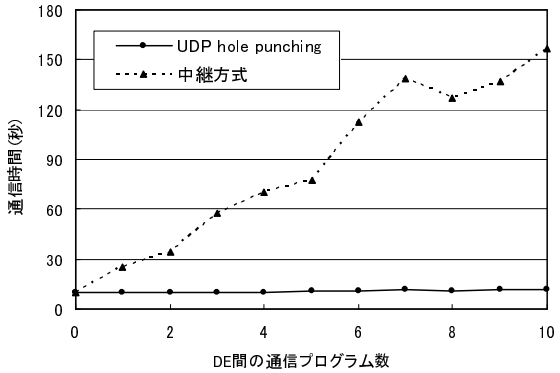

図 5: BC 間のファイル送信における通信時間

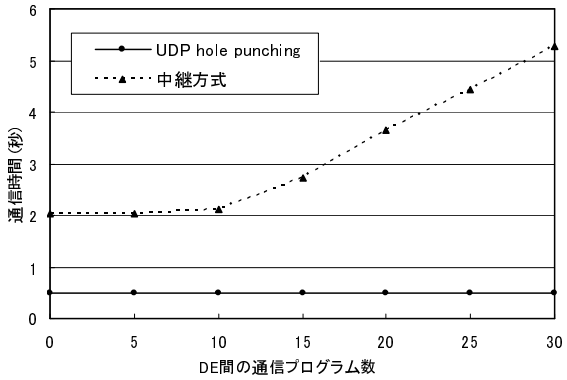

図 6: BC 間のメッセージ通信における通信時間

5.3 実環境での動作検証

東京理科大学野田キャンパス内の 2 拠点と千葉県内の学外の 2 拠点にある各計算機 (計 21 台) において，LampEye を用いてグリッドを構築し，正常に動作すること確認した．

6 応用事例

LampEye を，P2P 型の接続を必要とする，コラボレーション環境を構築するアプリケーションに適用した．図 7 の遠隔計算機操作システムは，VNC[6] のように，遠隔計算機のデスクトップ画面を取得し，マウスやキーボートのイベントを通知することで，遠隔計算機を操作することが可能である．また，ファイルをフレーム内にドラッグアンドドロップすることで遠隔計算機のデスクトップへ送信し，逆に，フレーム内のファイルをドラッグしたままフレームの外に出すことで遠隔計算機から取得することができる．

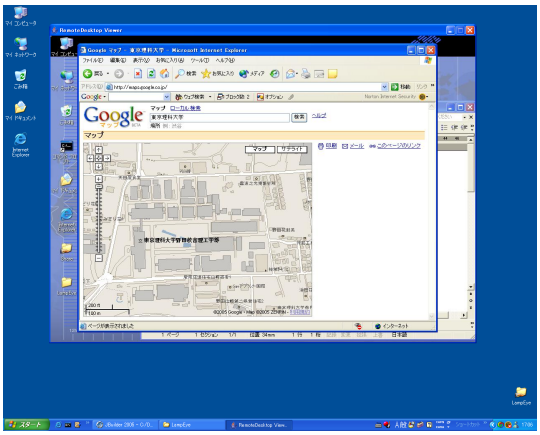

図 7: 遠隔計算機操作システム

7 おわりに

本研究では，異なる複数のプライベートネットワーク間における直接通信が可能な，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 の設計と実装を行った．LampEye を利用することで，従来のグリッドミドルウェアよりもネットワーク構成の制約を受けにくくなり，分散処理だけでなく，大量のデータ通信を必要とするデータグリッドや，地理的に分散した利用者間でのコラボレーション環境を容易に構築することが可能となる．

参考文献

[1] Ian Foster, Carl Kesselman. Globus: A metacomputing infrastructure toolkit. The International Journal of Supercomputer Applications and High Performance Computing, pp.115-128, 1997.

[2] Dietmar W. Erwin, David F. Snelling. UNICORE: A Grid Computing Environment, Proceedings of the 7th International Euro-Par Conference Manchester on Parallel Processing, pp.825-834, 2001.

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

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

[5] J. Rosenberg, J. Weinberger, C. Huitema, and R. Mahy. Simple Traversal of UDP through NATs, RFC3489, 2003.

[6] Tristan Richardson, Quentin Stafford-Fraser, Kenneth R. Wood and Andy Hopper. Virtual Network Computing, IEEE Internet Computing Vol. 2, No. 1, pp. 33-38, 1998.