グリッドアプリケーション評価のためのネットワークエミュレーション

笠井武史[†,☆] 西村元一[†] 前田高宏[†,☆☆]
大澤清[†] 合田憲人[†,☆☆☆]

グリッドソフトウェア開発者に定量的なソフトウェアの評価を可能にする，グリッドエミュレーションシステムが開発されている．しかし，現状のシステムでは時系列に変化する動的なネットワーク性能の挙動の再現に問題がある．本稿では，グリッドにおける遅延時間やスループット，パケットロスといったネットワーク性能の時系列な変化を簡単にエミュレーションするシステムを提案する．提案システムでは，ネットワークシナリオファイルに従ってエミュレーションを行い，さらにそのシナリオファイル作成を支援するツールの開発を行った．システムの性能評価を行った結果，ユーザによる効率的なソフトウェアの評価が可能になることが示された．

Network Emulation for Evaluation of Grid Applications

Takefumi Kasai,[†,☆] Motokazu Nishimura,[†]
Takahiro Maeda,[†,☆☆] Kiyoshi Osawa[†] and Kento Aida[†,☆☆☆]

Grid emulation systems, which enable grid software developers to conduct quantitative evaluation of their software, have been developed. However, the existing systems have a problem of difficulty in emulating dynamic and time series behavior of network performance. This paper proposes a network emulation system, which easily enables time series emulation of network behavior e.g. latency, throughput and packet losses, on the Grid. The proposed system emulates the network behavior following the network scenario file, which is provided by the user with file generation support tools. The experimental results show that the performance of the proposed tool to help users conducting effective evaluation of their softwares.

1. はじめに

近年，その利用が期待されているグリッドコンピューティングの分野では，グリッド上で動作するさまざまなアプリケーションやミドルウェアが開発されており，基盤技術が整備されつつある．このようなグリッド上で動作するソフトウェアを効率的に開発するには，ソフトウェアを複数回実行して性能を評価する必要があるが，ソフトウェア開発者が実グリッド実験環境で評価を行うには以下にあげるような問題点がある．

(1) 実グリッド実験環境を構築するためには，ネットワーク上の多くの計算資源を集める必要があり，計算機やネットワークに関する専門的な知識が要求される．

(2) 実グリッド実験環境は複数の研究機関が計算資源を互いに供与している場合がほとんどであり，計算機構成は固定的で，また個々の実験に許されるグリッド実験環境の利用機会は制限されることがある．

(3) 実グリッド実験環境では計算機の負荷やネットワークの性能が時々刻々と変化するため，再現性のある評価を行うことが困難である．

(4) ソフトウェア開発者が意図的にネットワークや計算機の障害を引き起こしソフトウェアの挙動を調べたい場合，実グリッド実験環境においてそのような障害を引き起こすことは困難である．

特に多くのグリッド環境に見られるように，グリッドを構築するネットワークとしてインターネットが用いられている場合は (3)，(4) の問題は顕著である．

これらの問題を解決するために，グリッド環境のシミュレーションを行う Bricks[1] や SimGrid[2] などのシミュレータに関する研究が進められている．これらのシミュレーションでは，ソフトウェアの挙動を簡略

† 東京工業大学
Tokyo Institute of Technology
☆ 現在，NTT コミュニケーションズ株式会社
Presently with NTT Communications Corporation
☆☆ 現在，東京大学
Presently with The University of Tokyo
☆☆☆ 現在，国立情報学研究所
Presently with National Institute of Informatics

化したモデルを用いて評価が行われ，スケジューリングなどのアルゴリズムの評価には適している．しかし，ソフトウェアの開発では，プログラムを実行できる評価環境が必要であり，シミュレーションでは不十分である．

グリッド環境のエミュレーションについては，MicroGrid[3]，Panda WAN Emulator[4] や，主にネットワーク技術の評価を目的とした StarBed[5]，EMPOWER[6]，Lenet[7] などのエミュレーションシステムが開発されている．MicroGrid は，グリッド環境エミュレーションソフトウェアであり，グリッド環境を再現した PC クラスタ上で実際のアプリケーションを動作させることができる．ネットワークに関しては，SSFNet[8] というエミュレータソフトウェアを利用しており，下層レイヤに至るまでネットワーク構成のより詳細な再現が可能である．しかし，MicroGrid では以下にあげる 2 点の問題点がある．

- アプリケーション実行中の計算機やネットワークの負荷の時系列な振舞いを再現することが困難である．
- 設定ファイルの記述が非常に複雑で，ユーザがこの記述法に慣れるためには多くの時間を要する．

他のシステムについても，同様の問題が残されている．たとえば，Panda WAN Emulator や EMPOWER では，実ネットワーク環境上で収集したトレースデータ（ログ）を用いて負荷の時系列な振舞いをエミュレーションする機能を有しているが，エミュレーション対象のネットワークのトレースデータをユーザが入手することは困難であり，ユーザの負荷が大きい．Lenet では，ネットワークパケットをキャプチャしてシナリオを作成する機能があるが，対象が RTP による通信であり，グリッド環境のエミュレーションに利用することが難しい．また，エミュレーション環境の設定，特に時系列な振舞いの設定を行うためにユーザが複雑な設定またはスクリプトを記述しなければならないという問題も残されている．

本稿では，グリッド環境のエミュレーションにおいて特に重要であるネットワークのエミュレーション技術に着目し，これらの問題を解決するためのグリッド上のネットワークのエミュレーションシステムを提案し，その実装について述べる．

提案システムは単一の PC クラスタ上に実装され，PC クラスタ上の計算ノードにエミュレーション対象であるグリッド環境の計算資源の役割が割り当てられる．また，計算ノードはルータ PC を経由して接続され，ルータ PC 上でエミュレーション対象であるグリッド環境のネットワーク（WAN）のエミュレーションが行われる．ネットワーク性能のエミュレーションでは，パケットの遅延やパケットロス，通信スループットの再現を行う必要があるが，これらの再現には NISTNet[9] を用いている．また計算機間の時系列的なネットワーク（WAN）の振舞いを再現するために，「ネットワークシナリオ」と呼ばれるネットワーク性能が時系列表記されたファイルを用いたエミュレーションを行う．ネットワークシナリオは XML により表記されたファイルであり，ユーザ自身が本ファイルを記述することも可能であるが，この方法のみでは負荷が大きい．そのため，提案システムでは，実グリッド実験環境上でのソフトウェア実行時のネットワーク性能を収集しネットワークシナリオファイルを生成するツールや，ユーザが GUI 上の描画や数学モデルを用いてネットワークシナリオファイルを生成する支援ツールが用意されている．

提案システムを PC クラスタ上に実装し，性能評価を行った結果，本システムを用いることで，ネットワークの遅延やスループットといったネットワーク性能を時系列に変化させるエミュレーションが可能であることが確認された．これにより，開発者はより現実に即した，再現性のある評価を簡単かつ繰り返し行うことができる．また実グリッド実験環境では実現が困難であるネットワーク障害やパケットロスなどもユーザが意図したとおりに引き起こすエミュレーションが可能であることも確認された．

以降，2 章では提案システムの基盤となるエミュレーション環境について概説し，3 章では，提案システムの重要な機能であるネットワークシナリオを用いたネットワークエミュレーション手法について述べる．4 章では提案システムの実装について述べ，5 章では，性能評価結果を示す．最後に 6 章で本稿のまとめと今後の課題について述べる．

2. ネットワークエミュレーション環境

本章では，提案システムの基盤となるエミュレーション環境である擬似グリッド実験環境について概説する[10]．擬似グリッド実験環境では，エミュレーション対象の PC クラスタやネットワーク（以降それぞれ，仮想 PC クラスタ，仮想ネットワークと表記）の処理が PC クラスタ上の計算ノードに割り当てられ，エミュレーションが実現される．このうち仮想計算ノードの処理は，PC クラスタ上の計算ノードで実行される．また仮想ネットワーク間の通信，すなわち WAN 上の通信は，PC クラスタ上の PC ルータを介して行

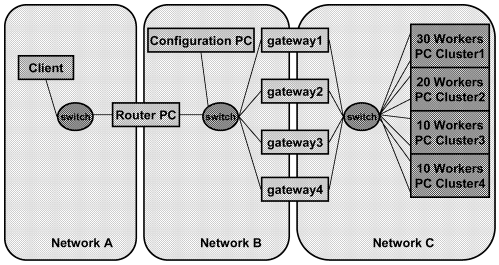


図 1 擬似グリッド実験環境のネットワーク構成図
Fig. 1 The network diagram of the emulated Grid testbed.

われ，本 PC ルータ上で仮想ネットワーク上の通信エミュレーションが実行される．

現在，**図 1** に示すようなマスタ・ワーカ方式のソフトウェアの評価を目的としたシステムが実装されている．図中では，エミュレーション対象であるグリッド上のローカル計算機（client）から遠隔の 4 台の仮想 PC クラスタに対してマスタ・ワーカ方式でジョブを割り当てる場合を想定している．図中，gateway は仮想 PC クラスタのゲートウェイノードを表しており，client から各 gateway （すなわち PC クラスタ）間の通信がルータ PC（Router PC）を経由することにより，ルータ PC 上での通信遅延，スループット，パケットロスのエミュレーションが行われる．

図中の Workers は仮想 PC クラスタの計算ノードを意味している．本システムではこれらの計算ノードは同一ネットワークセグメントに接続されているが，ソフトウェア的にグループ分けすることにより，各仮想 PC クラスタ上の計算ノード数を設定することができる．図中の例では，4 台の仮想 PC クラスタがあり，それぞれに 30 台，20 台，10 台，10 台の計算ノードがある環境を表している．

3. ネットワークシナリオを用いたグリッドネットワークエミュレーション手法

本章では，ネットワークシナリオを用いて時々刻々と性能が変化するグリッド上のネットワークのエミュレーション手法について述べる．

本システムを利用するソフトウェア開発者のニーズとして，以下の 2 点が考えられる．

(1) ユーザが意図した仮想的なネットワークの状況を再現したい．
(2) 実グリッド実験環境におけるネットワークの状況を再現したい．

ここでそれぞれの場合において想定される本システムの具体的な利用例とシナリオ作成支援の方法について述べる．

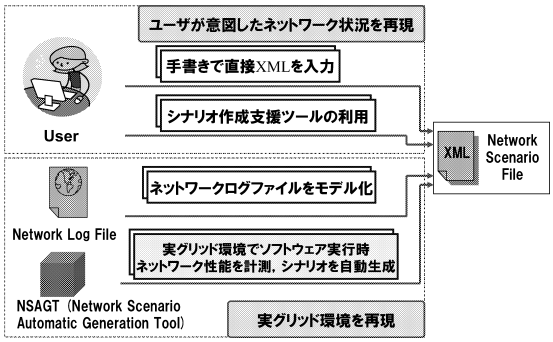


図 2 ネットワークシナリオの作成方法
Fig. 2 Generation of a network scenario file.

3.1 仮想的なグリッド環境上での実験

仮想的なグリッド環境上の実験を行う利用例として，以下のような状況が想定される．

- グリッドにおける負荷分散手法を評価するため，グリッドソフトウェア実行中にネットワーク障害を意図的に引き起こして，ソフトウェアの挙動を調べ，正しく負荷分散が行われているか調べたい．
- パケットロスが恒常的に数%生じるようなネットワーク環境において，開発したソフトウェアが正しく動作するかどうか検証したい．

すでに商業利用の段階にあるインターネット上に構築された実グリッド実験環境でこれらの実験を行うことは困難である．しかし，ネットワークシナリオを用いた擬似グリッド実験環境においては，どのくらいの期間，どの程度のネットワーク障害やパケットロスを引き起こすか，シナリオファイルに記述するだけで実現することが可能である．

ユーザが意図したネットワーク状況を再現する場合，**図 2** にあるようにユーザは，手書きで直接 XML を入力し，シナリオファイルを作成することも可能であるが，長時間にわたるシナリオになるとそのサイズが大きくなるため，ユーザの負荷が大きい．そのため本システムでシナリオ作成を容易にする「シナリオ作成支援ツール」が用意される．本ツールでは，GUI 上でグラフを操作することで視覚的にネットワークシナリオを作成することが可能である．

3.2 実グリッド実験環境の再現

実グリッド実験環境の再現を行う例として，以下のような状況が想定される．

- 大規模な実グリッド実験環境で 1 度だけソフトウェアの評価を行った．しかし，その実験環境はさまざまな研究機関が計算機資源を互いに供与して構築されているため，個人単位に許される利用

期間は制限される．そのため，評価を行ったときと同種のネットワーク，計算機の状況を再現したうえで細かいパラメータを変えつつ繰り返しソフトウェアの性能評価を行いたい．

この場合，図2にあるようにユーザはAPAN（Asia-Pacific Advanced Network）[11] などから提供されているネットワークログファイルをモデル化することによりネットワークシナリオを作成することができる．

また，本システムではネットワークシナリオ自動生成ツール “NSAGT: Network Scenario Automatic Generation Tool” を提供する．本ツールでは，実グリッド実験環境でグリッドソフトウェア実行時に同時にネットワーク性能を測定し，ネットワークシナリオファイルを自動的に生成することが可能である．本ツールによって測定可能なネットワーク性能指標は遅延時間とスループットである．

ネットワーク遅延時間測定機能は，ping を用いて該当する複数ホスト間の遅延時間を同時に測定する．ネットワークスループット測定機能は，グリッドソフトウェア実行中に該当ネットワークのスループットを測定し，ネットワークシナリオを自動的に生成する．NSAGT のそれぞれの機能の詳細については，4.3 節で述べる．

4. 実　　装

ネットワークシナリオを用いたグリッドネットワークエミュレーションを実現するためのシステムの全体設計図を**図 3** に示す．本システムは，「シナリオ作成部」，「ネットワークシナリオシステム部」，「擬似グリッド実験環境部」の 3 つの部分から構成されている．

ユーザがネットワークシナリオを用いて実際にネットワークエミュレーションを行うまでの手順は以下のようになる．

(1) **ネットワークシナリオの作成**

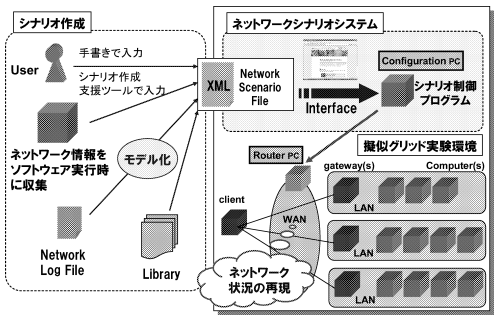


図 3 グリッドネットワークエミュレーション手法の全体設計
Fig. 3 An overview of Grid emulation system.

ユーザはまず「ネットワークシナリオファイル」を用意する．

ネットワークシナリオファイルは，可読性を保った XML で記述されているため，ユーザは手書きで書くこともできるが，図中にあるようなネットワークシナリオ作成支援のためのツールを用いて用意することもできる．

(2) **ネットワークシナリオシステムへファイルの読み込み**

ネットワークシナリオファイルが用意された後，それを Web インタフェースを通してネットワークシナリオシステムへ入力する．

シナリオファイルをブラウザを通してアップロードするだけで，ファイル入力後のシナリオファイルの解析やネットワークエミュレータの制御などの動作は，図中の “Configuration PC” 上にある「シナリオ制御プログラム」がすべて自動的に行うため，ユーザはシナリオファイルを用意する以外の煩雑な設定ファイルを記述する必要がなく，簡単な操作性を実現している．

(3) **グリッドネットワークのエミュレーション**

ユーザが Web インタフェース上でエミュレーション実行を指示することにより，ネットワークシナリオに基づくエミュレーションが実施される．この状態でユーザ評価対象のソフトウェアを 2 章で説明した擬似グリッド実験環境上で実行する．

次節以降は，細かい実装に関して詳説する．

4.1 ネットワークシナリオファイルの記述仕様

ネットワークシナリオファイルは，ネットワークの動的な性能変化をソフトウェア実行開始時間を 0 とした時系列で記述する．現在のバージョン（ver.1.0.2）では以下の 3 つのネットワーク性能項目に関して記述することができる．

- ネットワーク遅延時間 [ms]
- スループット [Byte/sec]
- パケットロス率 [%]

シナリオファイルは XML を用いた記述仕様が定められている．XML を用いることで，ファイルの可読性を保つことができ，エディタを用いて容易に簡単なシナリオや細かいパラメータの変更を記述することができる．また，XML でシステムとしての汎用性も増す．

ネットワークシナリオファイルの記述例を**図 4** に示す．この例では，2 台のホスト間（g0.alab.ip.titech.ac.jp → g1.alab.ip.titech.ac.jp）での，各シナリオ時間（0, 10, 20 [s]）それぞれにおける遅延，スループット，パ

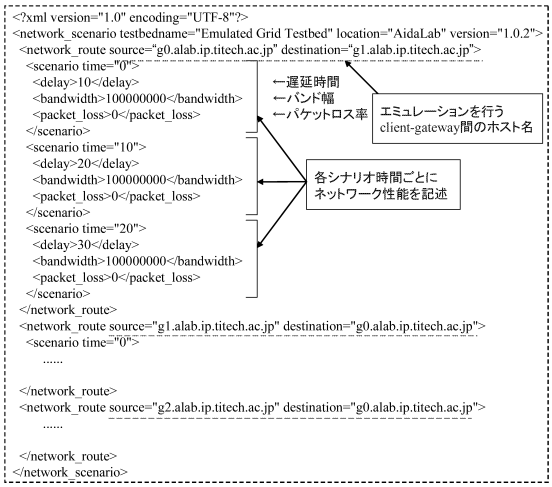


図 4 ネットワークシナリオの記述例
Fig. 4 An example of a network scenario.

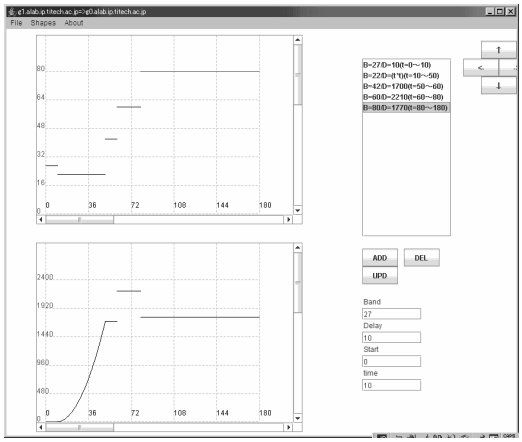


図 5 シナリオ作成支援ツールの GUI 画面
Fig. 5 A snapshot of the scenario generation support tool.

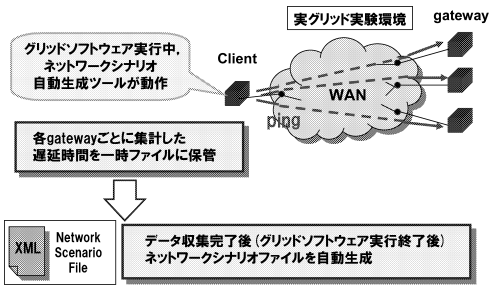


図 6 ネットワークシナリオ自動生成ツール（遅延時間測定機能）
Fig. 6 The automatic scenario generation tool (delay measurement).

ケットロスを記述している．この場合は遅延時間のみ 10，20，30 [ms] と変化させている．

4.1.1 ネットワーク障害発生の記述方法

ユーザが意図的にネットワークの障害を発生させたい場合は，

<bandwidth>0 </bandwidth>

または，

<packet_loss>100 </packet_loss>

と，シナリオに記述することによって実現できる．

4.2 シナリオ作成支援ツール

本システムでは，ユーザが意図したネットワークの状況を再現する際にネットワークシナリオの作成を簡単化するシナリオ作成支援ツールを提供している．本ツールでは，**図 5** に示すような GUI 上でグラフを作成することで，ネットワークの性能変化を視覚的に表現しながら，シナリオを作成することができる．また，想定するシナリオが特定の数学モデルに従う場合は，数式を入力することも可能である．

4.3 ネットワークシナリオ自動生成ツール

ネットワークシナリオ自動生成ツール（NSAGT）は，実グリッド実験環境上で実行させることによりネットワークの遅延およびスループットを測定し，ネットワークシナリオファイルを自動生成する．

4.3.1 ネットワーク遅延時間測定機能

ネットワーク遅延時間測定機能は，**図 6** に示すように実グリッド実験環境において “client PC” でグリッドソフトウェアを実行している際，同時に本ツールを動作させるだけで該当する複数ホスト間のネットワーク遅延時間を測定し，自動的にネットワークシナリオファイルを生成する．

遅延時間の測定には ping を用いている．ユーザは本ツールの設定ファイルに ping を実行する時間間隔，シナリオ時間間隔，各 gateway のホスト名（または IP アドレス）を記述する．シナリオ時間間隔とは，シナリオファイル中にシナリオとして記述される時間間隔のことである．つまり，シナリオ時間間隔ごとに実際のエミュレーション時には動的な変化が行われることになる．ツールが実行されると，各 gateway ごとに遅延時間測定結果を一時ファイルに保存し，遅延時間データの収集が完了後，それらをまとめて 1 つのネットワークシナリオファイルとして生成する．

4.3.2 ネットワークスループット測定機能

本機能については，実グリッド実験環境でグリッドソフトウェア実行中にソフトウェアが行うデータ通信におけるデータ容量と Ninf-G [12] の API より得られる通信時間の情報からスループットを自動算出する機能を実装中である．

4.4 ネットワークシナリオシステム制御プログラム

Web インタフェースを通して入力されたネットワー

クシナリオファイルは図 3 における “Configuration PC” 上に置かれたネットワークシナリオ制御プログラムに読み込まれる．シナリオ制御プログラムはシナリオファイルを解析し，エミュレータの設定を動的に変更するスクリプトを生成する．エミュレーションを行うまでの制御プログラム動作は以下のようになる．

(1) **ネットワークシナリオファイル（XML）の解析**
入力されたネットワークシナリオが XML 仕様記述に沿ったものであるかどうか，解析する．誤った文法で書かれていた場合にはエラーを表示する．

(2) **ネットワーク性能値の補正**
ユーザは，本システムを構築した PC クラスタ間の物理的なネットワーク性能値（遅延時間，スループット，パケットロス率）や NISTNet のオーバヘッドによるスループット損失率をシステムの設定ファイルにあらかじめ記述しておくことができる．シナリオ制御プログラムはこれらの値を用いて，シナリオに書かれた遅延時間から物理的な遅延時間を差し引いたり，NISTNet のオーバヘッド分，スループットの設定値を増加させたりするなどの補正を自動的に行う．また，シナリオに書かれたネットワーク性能値が物理的な性能値を超えている場合は，エミュレーションは行われない．これらの補正を行うことでより精度の高いネットワークエミュレーションを実現することができる．4.1.1 項で述べたネットワーク障害発生がシナリオに記述されていた場合，NISTNet での実際のエミュレーションにおいては，packet loss を 100%と設定し，“Router PC” を通るすべてのパケットを破棄させることで障害を実現する．

(3) **NISTNet 制御スクリプトの生成**
補正された値をもとにネットワークエミュレータ “NISTNet” の設定を動的に変更するスクリプトが生成される．NISTNet 制御スクリプトの例を**図 7** に示す．それぞれのホスト間に対する NISTNet の設定変更コマンドの間に sleep コマンドを挿入することで動的なネットワーク性能変化を実現している．

(4) **NISTNet 制御スクリプトの実行**
ユーザが Web インタフェース上でエミュレーション実行を指示すると，制御スクリプトが図 3 における “Configuration PC” から NISTNet がインストールされた “Router PC” に向けて実行され，ネットワーク時系列なエミュレーションが行われる．なお，スクリプトの実行には rsh を利用している．

```
#!/bin/sh

/usr/local/bin/cnistnet -u
/usr/local/bin/cnistnet -a g0.alab.ip.titech.ac.jp g1.alab.ip.titech.ac.jp --delay 9.75
/usr/local/bin/cnistnet -a g1.alab.ip.titech.ac.jp g0.alab.ip.titech.ac.jp --delay 9.75
/usr/local/bin/cnistnet -a g0.alab.ip.titech.ac.jp g2.alab.ip.titech.ac.jp --delay 99.75 --bandwidth 10000000 --drop 1
/usr/local/bin/cnistnet -a g2.alab.ip.titech.ac.jp g0.alab.ip.titech.ac.jp --delay 99.75 --bandwidth 10000000 --drop 1
sleep 10
/usr/local/bin/cnistnet -a g0.alab.ip.titech.ac.jp g1.alab.ip.titech.ac.jp --delay 19.75
/usr/local/bin/cnistnet -a g1.alab.ip.titech.ac.jp g0.alab.ip.titech.ac.jp --delay 19.75
sleep 10
/usr/local/bin/cnistnet -a g0.alab.ip.titech.ac.jp g1.alab.ip.titech.ac.jp --delay 29.75
/usr/local/bin/cnistnet -a g1.alab.ip.titech.ac.jp g0.alab.ip.titech.ac.jp --delay 29.75
/usr/local/bin/cnistnet -a g0.alab.ip.titech.ac.jp g2.alab.ip.titech.ac.jp --delay 149.75 --bandwidth 10000000 --drop 1
/usr/local/bin/cnistnet -a g2.alab.ip.titech.ac.jp g0.alab.ip.titech.ac.jp --delay 149.75 --bandwidth 10000000 --drop 1
sleep 20
/usr/local/bin/cnistnet -a g0.alab.ip.titech.ac.jp g2.alab.ip.titech.ac.jp --delay 99.75 --bandwidth 10000000 --drop 1
/usr/local/bin/cnistnet -a g2.alab.ip.titech.ac.jp g0.alab.ip.titech.ac.jp --delay 99.75 --bandwidth 10000000 --drop 1
```

図 7 NISTNet 制御スクリプトの例
Fig. 7 An example of configuration script for NISTNet.

5. 性能評価

本章では，シナリオ再現性の基礎評価を遅延時間，スループット，パケットロス率に対して行った．なお，パケットロス率については他と同様に再現性が確認できたため，紙面の都合上，結果の掲載を省略する．

図 1 に示した提案システムの実装を行ったエミュレーション環境の計算機構成の詳細を**表 1** に示す．

5.1 シナリオ再現性評価実験

5.1.1 遅延時間シナリオの再現性評価

ネットワーク遅延時間シナリオの再現性評価は，ping を用いて行った．

まず，client（1 台）–gateway（1 台）間において，シナリオ時間間隔を 1 秒とし，遅延時間を 10 → 20 → 30 → 40 → 50 → 10 →… [ms] と変動させたときのシナリオと測定結果を**図 8** に示す．図では，横軸を時間 [s]，縦軸を遅延時間 [ms] として，“scenario” の実線がネットワークシナリオファイルに書かれた想定した遅延時間の変動を表しており，“Result” の点が ping 実行による遅延時間の実測値を表している．図より，1 秒間隔で変動するシナリオを NISTNet において十分に再現できることが確認できる．

次に，gateway ホストを複数台にして同様の実験を行った．client（1 台）–gateway（4 台）間（4 経路）においてシナリオ時間間隔を 5 秒にした際の用いたシナリオと遅延時間測定結果を**図 9** に示す．また，同じく 4 経路において 6 時間程度の長時間シナリオを用いて遅延時間の再現性を評価した結果を**図 10** に示す．図の見方は図 8 と同様である．gateway が複数台になっても，また長時間にわたるシナリオになっても，シナリオの記述内容に沿って遅延時間の動的変動が再現されていることが確認できる．

表 1 擬似グリッド実験環境の計算機構成
Table 1 The configuration of the emulated Grid testbed.

	CPU	Memory	OS	ノード数	ネットワーク仕様
client	Pentium4 (2.4 GHz)	1 GB	Linux2.4.18	1	1000BASE-TX
Router PC	Pentium4 (2.4 GHz)	1 GB	Linux2.4.7	1	1000BASE-TX
Configuration PC	Pentium4 (2.4 GHz)	512 KB	Linux	1	1000BASE-TX
gateway	Pentium4 (2.4 GHz)	512 KB	Linux2.4.7	4	1000BASE-TX
Worker	Pentium3 (1.4 GHz) Dual	512 KB	Linux2.4.10	36	1000BASE-TX

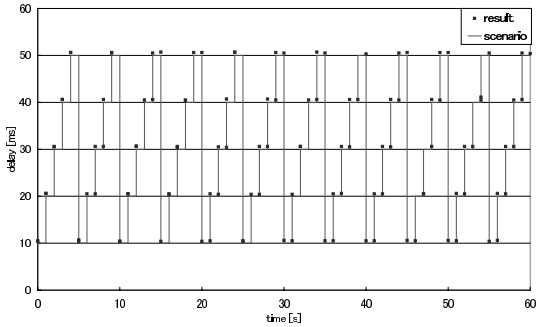


図 8 client（1 台）–gateway（1 台）間の遅延時間の再現性評価結果
Fig. 8 The results of delay time reproducibility between a client and a gateway.

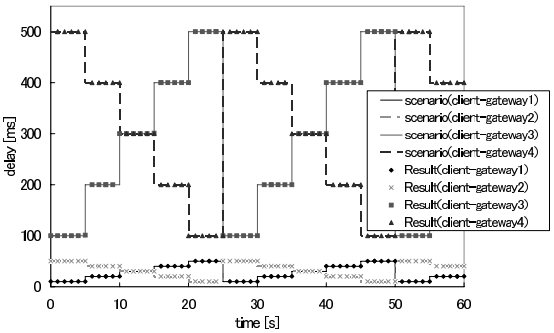


図 9 client（1 台）–gateway（4 台）間の遅延時間の再現性評価結果
Fig. 9 The results of delay time reproducibility between a client and four gatways.

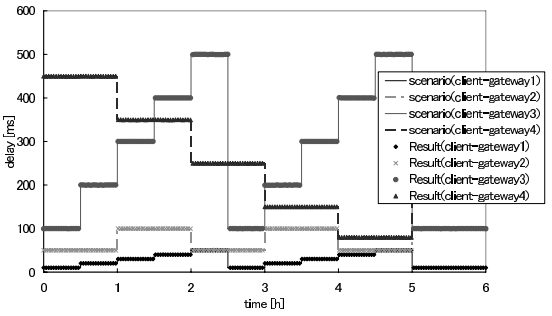


図 10 長時間シナリオを用いた遅延時間の再現性評価結果
Fig. 10 The results of delay time reproducibility in a large time scale.

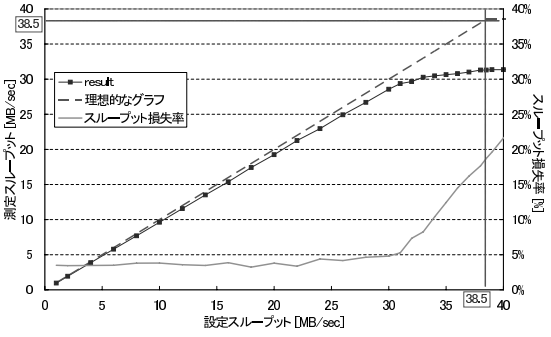


図 11 NISTNet オーバヘッドによるスループット損失率
Fig. 11 The throughput loss ratio by the overhead of NISTNet.

5.1.2 スループットシナリオの再現性評価

スループットシナリオの再現性評価は，Netperf[13] を用いて該当ホスト間のスループットを測定することで行った．

実験に先立ち，NISTNet のオーバヘッドによってスループットの設定値と実測値との間に差異が生じることが分かっていたため，NISTNet オーバヘッドによりどの程度の損失が生じるかをシナリオを用いずに評価を行った．**図 11** に評価結果を示す．図において，横軸は NISTNet スループット設定値を，縦軸は Netperf によるスループット実測値と損失率を表している．提案システムを実装したホスト間の最大スループットを測定したところ，およそ 38.5 [MB/sec] であったため，それを超えるスループットは再現することができない．図よりこの最大スループットに近い値を設定値とすると，スループット設定値と実測値との間に大きな差異が見られ，最大で 20%程度実測値の方が小さくなった．また，本実装での最大スループット 38.5 [MB/sec] に対し，およそ 30 [MB/sec] 以下の値をスループットとして設定した場合にはエミュレーションの精度は高まる．しかし，この場合においても設定値と実測値との間に 4〜5%程度の誤差が生じるが，この範囲においてはスループット損失率がほぼ一定であることが本評価

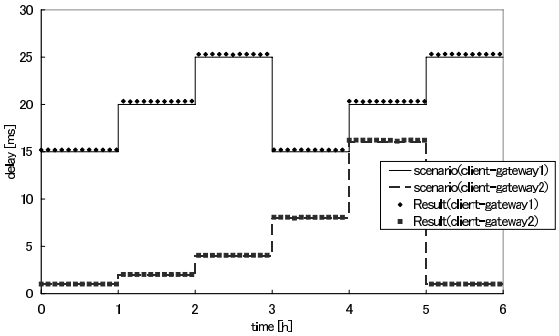


図 12 長時間シナリオを用いたスループットの再現性評価結果（損失補正後）
Fig. 12 The results of throughput reproducibility in a large time scale with the throughput correction.

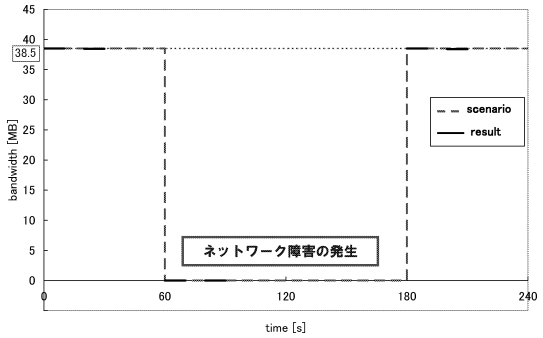


図 13 ネットワーク障害の発生
Fig. 13 The results for a scenario with network failure.

実験から明らかになったため，シナリオ制御プログラムがその恒常的な損失率を勘案して自動的に補正することによってより精度の高いスループット再現性が実現できる．

次に，cilent–gateway（2 台）間（2 経路）において長時間シナリオを用いてスループットを変動させた場合の評価結果を**図 12** に示す．図において，“scenario” の線が想定したスループット変動シナリオを表しており，“Result” の点が Netperf を用いたスループットの実測値を表している．図 11 で明らかになったスループット損失分 5% を図 12 では補正している．図から長時間におけるスループットシナリオにおいても，スループット設定値の補正を行うことによってシナリオに沿ったスループット再現性が実現できていることが確認された．

5.1.3 ネットワーク障害発生シナリオの再現性評価

最後に，ネットワーク障害を意図的に発生させるシナリオの再現性評価実験を行った．**図 13** に用いたシナリオと実験結果を示す．本実験では，ネットワークシナリオファイルに bandwidth を 0 と記述すること

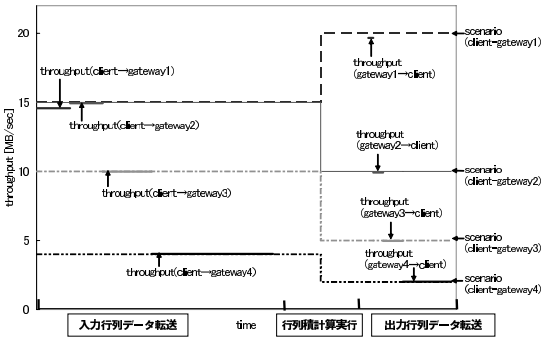


図 14 ベンチマークプログラムを用いた再現性評価実験結果
Fig. 14 The results of throughput reproducibility with the benchmark.

で，意図的に障害を発生させている．図よりプログラム実行開始 60 秒後，障害が発生しパケットが 1 つも通らないことが確認された．

5.2 ベンチマークプログラムを用いたスループットシナリオ再現性評価実験

本節では，行列積計算を行う実際のソフトウェアを用いたスループットシナリオの再現性評価実験について述べる．本ベンチマークプログラムは，Ninf-G [12] を用いて書かれている．計算に用いる入力行列データを client から各 gateway へ送信する場合と計算された出力行列データを gateway から client へ返す場合のそれぞれにおいて転送データサイズと転送時間からスループットを算出した．データ転送時間の取得には Ninf-G の API を利用した．

本実験に用いたシナリオと結果を**図 14** に示す．図では，行列積計算を各計算ノードで行っている最中に 4 経路の client-gateway 間のスループットが変化するシナリオを想定している．各 gateway に送信する入力データのデータサイズは 409.6 [MB] で，client に送信する出力データサイズは，81.92 [MB] である．本実験では 5.1.2 項で述べたスループットの損失分（5%）は，シナリオ制御プログラムにおいて NISTNet 制御スクリプトを生成する際，自動的に補正されている．図より提案システムを用いてシナリオに沿ったスループットが再現できていることが確認された．

5.1.2 項および本節の結果より，スループット補正については，本稿で扱った規模の環境では 4〜5%程度に抑えられることが確認された．しかし，この値は用いるルータ PC やエミュレーションソフトウェアの性能に依存すると考えられるため，今後より詳細な評価を行っていく予定である．

5.3 ネットワークシナリオ自動生成ツールを用いた評価

本節では，4.3 節で述べたネットワークシナリオ自動生成ツール（NSAGT）遅延時間測定機能を用いた評価について述べる．本ツールの評価実験には，合田研究室内の client PC において以下の 4 つの gateway に向けてツールを実行することで，実際のネットワークの遅延時間の変動を反映したネットワークシナリオファイルを生成できるかどうか調べた．本実験では 5 [s] おきに ping を実行，シナリオ時間間隔は 20 [s] に設定し，計 5 分間の測定を朝，昼，夜の 3 回にわたり行った．

- gateway1：笠原研究室（早稲田大）
- gateway2：松岡研究室（東工大，大岡山）
- gateway3：小野研究室（東工大，すずかけ台）
- gateway4：合田研究室（東工大，すずかけ台，client PC と同一 LAN に接続）

図 15 は，朝（2007 年 1 月 29 日 AM9:21）に本ツールを実行して得られたネットワークシナリオファイルである．また，図 15 で示したネットワークシナリオを用いて，ネットワークエミュレーションを行った結果を**図 16** に示す．図より，評価を行った経路間の遅延時間はもともと数 [ms] と非常に小さいため，実測値に多少ばらつきが見られるが，ネットワークシナリオに記述した実グリッドの遅延時間の挙動が，擬似グリッド実験環境において再現されていることが確認できる．

このように本ツールを用いると，client 側でツールを実行するだけで容易に実グリッド環境のネットワークの挙動を反映してシナリオファイルを生成し，本シナリオファイルを用いた実験を行うことができる．他方，1 章で関連研究として紹介した Panda WAN Emulator は，実ネットワーク環境のトレースに NWS [14] を用いているため，トレースをとりたいすべてのマシンに NWS をインストールしなければならないが，それぞれ異なるポリシにより管理されている特定のサイトにおいては，それが許されない恐れがある．また，EMPOWER はネットワークトレースデータを用いることを想定しているが，エミュレーション対象のグリッド環境においてユーザが必要とするすべての情報，具体的には end-to-end の計算機間のネットワーク性能をすべて入手することは難しい．

提案システムが提供するネットワークシナリオ自動生成ツール（NSAGT）は，ユーザのアプリケーション実行時に容易に end-to-end のネットワーク性能を測定し，ネットワークシナリオファイルを生成できるため，これら既存ツールに比べて容易なエミュレーションが可能といえる．

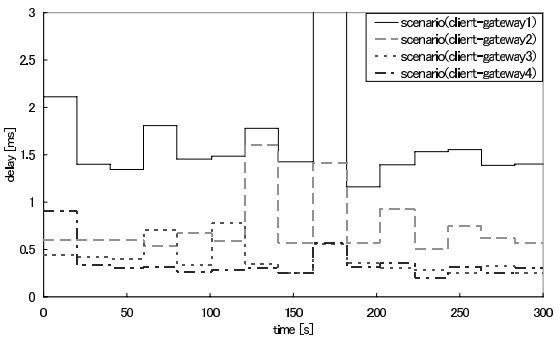


図 15 ネットワークシナリオ自動生成ツール評価結果（シナリオ）
Fig. 15 A network scenario created by the automatic scenario generation tool.

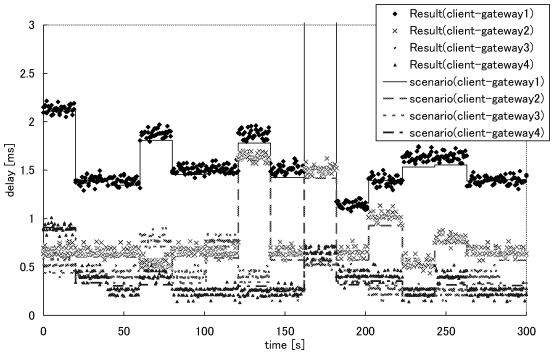


図 16 ネットワークシナリオ自動生成ツール評価結果（シナリオとエミュレーション結果）
Fig. 16 The results of delay time reproducibility of a scenario created by the automatic scenario generation tool.

6. まとめと今後の課題

本稿では，グリッド上のネットワークのエミュレーションシステムを提案し，その実装について述べた．ネットワーク性能が時系列に記述されたネットワークシナリオを用いて，ネットワークエミュレータ NIST-Net の設定を動的に変化させることで，グリッドネットワークの時系列な振舞いを再現できることが確認され，本提案システムが有効であることが確認された．また，本システムでは，ユーザのニーズごとにより簡単なシナリオ作成手法とその支援ツールを提供した．本システムを用いることで，グリッドソフトウェア開発者は簡単かつ効率的にソフトウェアの評価を行うことができる．

今後の課題としては，現状の実装において最大 4 経路までとなっているネットワークエミュレーションを

より大規模な環境でシステム実装しさらに詳細な評価を行うことや，NISTNet 以外のソフトウェアエミュレータやハードウェアエミュレータを用いてシステムの評価を行うことをがあげられる．ネットワーク性能値の補正に関しては，現状では実装したシステムの物理的なネットワーク性能やエミュレータのオーバヘッドをユーザがあらかじめシステム設定ファイルに記述し，その値をもとにシナリオ制御プログラムが補正を行っているが，これらの部分を完全に自動化すべく検討していきたい．また，ネットワーク性能のエミュレーションに限らず，帯域予約やルート指定を勘案したネットワークエミュレーションシステムの開発も検討課題としてあげられる．

さらに，現在のネットワークシナリオに合わせて，計算機の動的な性能変化（CPU 利用率やメモリ利用率など）を実現するための計算機シナリオとそのシステム実装を進めている．現状では物理的な性能を超えるシナリオが記述された場合，それを再現することができないが，今後は VM と仮想時間を用いて，たとえば物理的な計算機数よりも多い計算機を想定した実験環境でのソフトウェアの評価を行えるよう検討中である．

謝辞 本研究における評価に際して，ご協力していただいた笠原博徳教授（早稲田大），松岡聡教授，小野功助教授（東工大）に感謝の意を表します．

本研究の一部は，科学研究費補助金基盤研究（B）(課題番号 18300018）による．

参 考 文 献

1) Takefusa, A., Matsuoka, S., Nakada, H., Aida, K. and Nagashima, U.: Overview of a Performance Evaluation System for Global Computing Scheduling Algorithms, *Proc. 8th IEEE International Symposium on High Performance Distributed Computing* (*HPDC8*), pp.97–104 (1999).
2) Casanova, H.: Simgrid: A Toolkit for the Simulation of Application Schduling, *Proc. 1st IEEE/ACM International Symposium on Cluster Computing and the Grid* (*CCGrid2001*), pp.430–437 (2001).
3) Song, H.J., Liu, X., Jakobsen, D., Bhagwan, R., Zhang, X., Taura, K. and Chien, A. A.: The MicroGrid: A Scientific Tool for Modeling Computational Grids, *Supercomputing* (2000).
4) Kielmann, T., Bal, H., Maassen, J., van Nieuwpoort, R., Eyraud, L., Hofman, R. and Verstoep, K.: Programming environments for high-performance grid computing: The albatross project (2002).
5) Miyachi, T., Chinen, K. and Shinoda, Y.: Automatic Configuration and Execution of Internet Experiments on an Actual Node-based Testbed, *Tridentcom 2005*, pp.274–282 (2005).
6) Zheng, P. and Ni, L.M.: EMPOWER: A Cluster Architecture Supporting Network Emulation, *IEEE Trans. Parallel and Distributed Systems*, Vol.15, No.7, pp.617–629 (2004).
7) 石野正英，前田香織，河野英太郎，石田賢治：ネットワークエミュレータ Lenet の RTCP を用いたキャプチャ・リプレイ機能の設計，*IC2006*, pp.59–65 (2006).
8) SSFNet: network simulator. www.ssfnet.org
9) Carson, M. and Santay, D.: Tools: NISTNet: A Linux-based network emulation tool, *ACM SIGCOMM Computer Communication Review*, Vol.33 (2003).
10) 大角知孝，合田憲人：ネットワークエミュレーションを用いたグリッドアプリケーション負荷分散手法の評価，シミュレーション，pp.104–111 (2005).
11) APAN: Asia-Pacific Advanced Network. www.apan.net
12) Tanaka, Y., Nakada, H., Sekiguchi, S., Suzumura, T. and Matsuoka, S.: Ninf-G: A Reference Implementation of RPC-based Programming Middleware for Grid Computing, *Journal of Grid Computing*, Vol.1, No.1, pp.41–51 (2003).
13) NetPerf: NetPerf WWW pages and NetPerf manual included in archive. www.netperf.org/netperf/NetperfPage.html
14) Wolski, R., Spring, N.T. and Hayes, J.: The network weather service: A distributed resource performance forecasting service for metacomputing, *Future Generation Computer Systems*, Vol.15, No.5–6, pp.757–768 (1999).

(平成 19 年 1 月 22 日受付)
(平成 19 年 5 月 15 日採録)

笠井 武史（正会員）

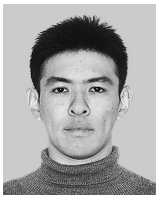
昭和 56 年生．平成 17 年早稲田大学理工学部電気電子情報工学科卒業．平成 19 年東京工業大学大学院総合理工学研究科物理情報システム専攻修士課程修了．同年 NTT コミュニケーションズ株式会社入社，現在に至る．

西村 元一（学生会員）

昭和 59 年生．平成 18 年神奈川大学理学部情報科学科卒業．現在，東京工業大学大学院総合理工学研究科物理情報システム専攻修士課程に在学中．

前田 高宏

平成 19 年東京工業大学工学部情報工学科卒業．現在，東京大学大学院情報理工学系研究科創造情報学専攻修士課程在学中．



大澤 清（学生会員）

平成 10 年東京工業大学工学部情報工学科卒業．平成 13 年東京大学大学院理学系研究科情報科学専攻修士課程修了．現在，東京工業大学大学院総合理工学研究科博士課程在学中．大規模並列数値計算に関する研究に従事．日本オペレーションズ・リサーチ学会会員．

合田 憲人（正会員）

昭和 42 年生．平成 2 年早稲田大学理工学部電気工学科卒業．平成 4 年同大学大学院修士課程修了．平成 8 年同大学院博士課程退学．平成 4 年早稲田大学情報科学研究教育センター助手，平成 8 年同特別研究員．平成 9 年東京工業大学大学院情報理工学研究科助手，平成 11 年同大学院総合理工学研究科専任講師，平成 15 年同助教授，平成 19 年国立情報学研究所特任教授，現在に至る．博士（工学）．並列・分散計算方式，グリッドコンピューティング，スケジューリング技術の研究に従事．電子情報通信学会，電気学会，ACM，IEEE-CS 各会員．