L-024

仮想環境を用いたネットワーク QoS 検証を可能とするエミュレーションシステムの設計と実装

Design and Implementation of Emulation System Enabling Network QoS Verification Using Virtual Environment

藏内 将博† 辻本 幸徳† 井口 信和‡
Masahiro Kurauchi Yukinori Tsujimoto Nobukazu Iguchi

1. 序論

近年，VoIP を利用したインターネット電話やビデオ会議などの音声・映像を扱うリアルタイム系アプリケーションが増加し，通信のマルチメディア化が進んでいる．これらのリアルタイム系アプリケーションは，広帯域及び低遅延なネットワークを必要とする[1]．さらに，インターネットのビジネス利用が増加したことにより，高品質なネットワークへの要求が高まっている．しかし，インターネットはベストエフォート型であり，通信サービスの品質(QoS: Quality of Service)を保証するものではない．このような背景から，インターネット上で QoS 保証を実現する技術として IntServ や DiffServ，IEEE802.1p が提案されている．

高品質な通信サービスを実現するためには，ネットワークの計画，設計，構築，運用のサイクルの中で QoS を検証し，その結果をネットワークに反映する必要がある．しかし，実際に運用されているネットワーク上で QoS を検証しようとすると様々な問題が出てくる．例えば，アクティブ測定により QoS を検証しようとした場合，測定対象のネットワークに負荷をかける事になる．また，パッシブ測定の場合だと，運用中のサービスが生成するトラフィックの収集が必要なため，プライバシーの問題が発生する．

そこで本研究では，実運用ネットワークに影響を与えることなく，低コストで手軽な QoS 検証環境の提供を目的として，仮想マシンで構成する仮想的なネットワーク上で QoS を検証するためのシステム(以下「本システム」という)を開発する．

本稿では，まず 2 章において本研究が検証の対象とする QoS アーキテクチャと品質尺度について述べ，3 章で本研究の関連研究を概観する．4 章にて仮想環境上で QoS を検証するための要件を定義し，5 章で要件定義に基づいたシステムの設計と実装について述べる．6 章では実装したシステムを定量的に評価し，最後に 7 章で本稿をまとめる．

2. QoS 検証の対象

本研究では，DiffServ 及び IEEE802.1p により QoS を保証するネットワークを検証の対象とする．

DiffServ と IEEE802.1p はタイプの異なるトラフィックをクラス分けして，そのクラスごとの優先度によって異なるサービスを提供し，特定のトラフィックの品質を保証する．優先度情報を格納するために，DiffServ では IP ヘッダ中に 6 ビットの DSCP(Differentiated Service Code Point)フィールド，IEEE802.1p では VLAN タグ付きのフレームヘッダ中に 3 ビットの PCP(Priority Code Point)フィールドを規定している．

ネットワークの品質を表す尺度(QoS パラメータ)として，パケット遅延時間や遅延変動，スループット，パケット損失率がある．本システムでは，仮想ネットワーク上でこれらの QoS パラメータを測定の対象として，最終的にユーザに提供する．ユーザは本システムをネットワークの事前検証に利用し，導き出された QoS パラメータを判断材料として最適なネットワークを設計，構築する．

3. 関連研究

アプリケーションの動作検証や通信プロトコルの評価を目的に，仮想的なネットワークを実験環境として構築する研究が行われている[2-16]．この実験環境はネットワークシミュレータとネットワークエミュレータに大別できる．

シミュレータとして広く知られている ns-2[2,3]や OPNET[4]では，仮想ネットワークの回線に帯域幅や遅延を割り当てて実ネットワークをシミュレートし，様々なキューイング/スケジューリング方式を設定してトラフィックがどのように流れるかを観察できる．そして，その際の QoS パラメータを求めることもできる．文献[5]では QoS 検証を目的として，リアルタイム分散システムをモデル化し，確率的モデル検査ツールを用いてスループットや RTT，パケット損失率を解析している．文献[6]では，大規模ネットワークの性能評価を目的として，フローレベルでのシミュレーションを行っている．これらのシミュレータは，通信処理やルータのバッファ制御など QoS 制御以外の細かい点を設計しなければならず，ユーザに余分な作業負担を強いることになる．

OS やネットワークスタックの仮想化技術を活用した多くのエミュレータが研究されている[7-16]．これらのエミュレータは，ネットワークの教育やトラブルシューティング，アプリケーションの動作検証，プロトコルの評価などを目的として開発されている．その中でも文献[11-16]に関しては，より実ネットワークに近い環境を目指して，通信遅延やパケット損失などのリンク特性を再現している．しかし，どのエミュレータも QoS 制御の設定や QoS パラメータの測定はできない．

本システムは仮想化技術を活用したネットワークエミュレータである．実際のルータやスイッチと同じ動作をする仮想ネットワーク機器を備え， 1 台の計算機上に仮想ネットワークを構築できる．そして，仮想ネットワーク上で QoS 制御の設定や QoS パラメータの測定を可能とする．

†近畿大学大学院総合理工学研究科
Interdisciplinary Graduate School of Science and Technology, Kinki University
‡近畿大学理工学部情報学科
Department of Informatics, School of Science and Engineering, Kinki University

4. 要件定義

本章では，仮想ネットワーク上で QoS を検証するための要件について述べる．

実ネットワーク上での QoS 検証では，アクティブ測定やパッシブ測定によって QoS パラメータを導出する．その際，測定用のトラフィックを生成したり，ネットワークを流れるトラフィックを観測したりする．本システムによって実ネットワークを用いた場合と同様の検証を可能とするために，以下のように要件を定義した．

要件1. 仮想ネットワークにおいて QoS 制御の設定が可能であること

要件2. 検証に再現性があること

ネットワークの再構築が必要になった場合や過去に設計したネットワークと新しいネットワークの動作の違いを検証したい場合，その都度仮想ネットワークを構築しなければならない．これでは，ユーザに作業負担をかけることになる．このため，繰り返して検証する作業が容易であることを要件とする．

要件3. 検証に柔軟性があること

実運用ネットワークを仮想ネットワークとして完全に再現することは困難なため，実運用環境に則った QoS パラメータの測定ができない．そこで，実運用環境で用いられるサービスのトラフィックを活用して柔軟な QoS 検証が可能なことを要件とする．また，実ネットワークでの検証と同様にアクティブ測定が可能なことも要件とする．

要件4. 仮想ネットワーク上のトラフィックを解析し QoS の評価が可能であること

5. 設計と実装

5.1 仮想ネットワークの構築

本システムでは，QoS 検証のための仮想的なネットワーク実験環境の構築に，ホスト OS 型の仮想化技術 User-mode Linux(UML)[17]を使用する．UML は Linux をユーザ空間で動作させる技術で，通常の Linux マシンと同様の仮想マシンを作成できる．UML による仮想マシンを複数動作させて，それらを相互に接続することで仮想ネットワークを構築する．本システムにおける，仮想ネットワークの構成図を図 1 に示す．仮想マシン間の通信には，UML に付属するツール uml_switch を用いる．uml_switch 起動時に生成される UNIX domain socket を通してパケットをやり取りすることで通信をエミュレートする．また，仮想ネットワークと物理ネットワークを接続する際には，TUN/TAP インターフェースとブリッジインターフェースを使用する．この外部ネットワーク接続部分から実運用のサービスが生成するトラフィックを取り込むことで，より実環境に近い柔軟な検証を可能とする．

本システムでは，複雑な仮想ネットワークを構築するために，様々なネットワーク機器の動作を模倣する仮想ネットワーク機器(以下「仮想機器」という)を用意している．利用可能な仮想機器としてはホスト，ルータ，スイッチングハブ，リピータハブがある．本稿では，それぞれを仮想ホスト，仮想ルータ，仮想スイッチ，仮想ハブと呼ぶ．これらの仮想機器は，コマンドラインインタフェースまたは GUI を通して操作する．

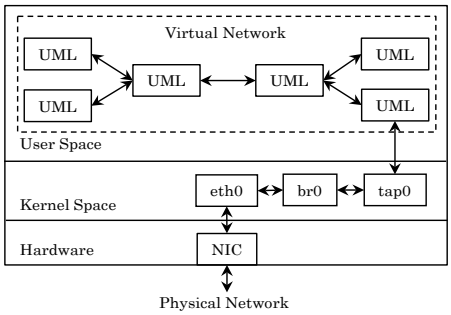

図 1　仮想ネットワークの構成図

仮想ホストは Debian GNU/Linux により標準的な Linux コマンドを扱える．仮想ルータはルーティングソフトウェアスイートの Quagga[18]を仮想マシン上で動作させることで，スタティック及び RIP，OSPF，BGP によるダイナミックルーティングを可能としている．仮想スイッチは，仮想マシン上にブリッジインターフェースを作成することで，MAC アドレスに基づくフレームのスイッチングを行う．その他に Spanning Tree Protocol や VLAN を設定できる．仮想ハブは，ハブモードで起動した uml_switch により実現する．

5.2 仮想ネットワークへ QoS アーキテクチャの適用

仮想ネットワーク上で QoS を検証するためには，本システムの仮想ルータと仮想スイッチを QoS 制御に対応させる必要がある．

5.2.1 仮想ルータにおける QoS 制御

ルータは OSI 参照モデルのレイヤ 3 のヘッダ情報をもとにトラフィックを分類し，それぞれに合ったサービスレベルを提供する．

ルータにおける QoS 制御機構は，パケットのクラス分類，マーキング，シェーピング/ポリシング，キューイング/スケジューリングといったコンポーネントから構成されている．Linux カーネルはこれらの QoS 制御に関する実装が充実しているため，これを利用して仮想ルータを QoS 制御に対応させる．クラス分類及びパケットマーキングに関しては，iptables[19]を用いる．そして，スケジューリング方式として代表的な FIFO(First-In, First-Out)や CBQ(Class-Based Queuing)，PQ(Priority Queuing)をサポートし，トークンバケットアルゴリズムによりシェーピング/ポリシングを行う．これらのトラフィック制御は tc(Traffic Control) [20]を用いて設定する．

5.2.2 仮想スイッチにおける QoS 制御

OSI 参照モデルのレイヤ 2 に相当するスイッチでは，IEEE802.1p において規定される VLAN タグ中の優先度情報に従って QoS 制御を行う．

仮想スイッチの QoS 制御機構は，トラフィックのクラス分類及びマーキングに使う情報が異なる事以外，仮想ルータと同じものとする．

5.2.3 QoS ポリシー管理インターフェース

QoS を保証する仮想ネットワークを構築するためには，QoS ポリシーを作成し仮想ルータ及び仮想スイッチに適用する．本システムは，QoS ポリシーを作成・編集・適用するための GUI を提供する．ポリシー作成時の設定項目としては，以下のものがある．

- ポリシーを適用する機器及びネットワークインターフェースの設定
- トラフィック分類基準(PCP，DSCP，プロトコル番号など)の設定
- トラフィッククラスの作成
- トラフィッククラス毎の割り当て帯域の設定
- パケットマーキングの設定
- シェーピング/ポリシングの設定
- キューイング/スケジューリング方式の設定

5.3 仮想ネットワークの保存と復元

仮想機器の設定や機器同士の接続，QoS 制御の設定など仮想ネットワークに関わるすべての情報を XML(eXtensible Markup Language)によって定義する．これにより，仮想ネットワークの情報を XML ファイルとして保存できるようになる．本システムで保存した仮想ネットワーク定義ファイルを読み込むと，XML のタグを解釈し仮想ネットワークを自動的に構築する．

図 2 に QoS 制御の設定情報を含む XML ファイルの例を示す．この例では，1 つの仮想ルータが 1 つの仮想ネットワークインターフェースを持ち，CBQ を用いた QoS 制御の設定情報が記述してある．

この機能により，仮想ネットワークの構築と QoS 制御の設定をくり返し実施する必要がなくなり，ユーザの作業負担を軽減できる．また，QoS 検証の中断・再開が可能になり，過去の QoS 制御設定と新しい設定を適用した際の比較も容易になる．

5.4 仮想ネットワーク上における QoS 検証

5.4.1 トラフィック解析による QoS パラメータ測定

ここでは，仮想ネットワーク上における QoS パラメータの測定方法について述べる．なお，測定対象項目はスループット，パケット遅延時間，遅延変動，パケット損失率であり，最終的にこれらの結果をグラフ化してユーザに提供する．

本システムは，1 台の計算機上に測定対象ネットワークを構築するため，一括してトラフィックを管理でき，スループットや End-to-End の遅延及び遅延変動，パケット損失の測定が容易である．また，仮想ネットワークという閉じた環境で QoS 検証をするため実運用環境に影響を与えることがない．

QoS パラメータを測定するために，仮想ネットワークを流れるパケットをキャプチャする．本システムは，UML 上でキャプチャプログラムを動作させることにより，指定した仮想機器のネットワークインターフェースを流れるパケットをタイムスタンプ付きでキャプチャする(図 3).

ネットワーク上の 2 つのノード間で片道遅延時間を計測するには，送信側と受信側で一意のパケットを識別するための情報をタイムスタンプとともに記録する．そして，送信側と受信側が同期した時計によってタイムスタンプを付加することができれば，2 点間の片道遅延を測定することができる．UML は *gettimeofday()*システムコールを発行してホスト OS 側の時刻を取得する(図 3)．そのため，ホスト OS 上のすべての UML は同一の時刻を共有することになる．この性質を利用して 2 点間の仮想機器の時刻を同期する．そして，受信側と送信側の仮想機器でキャプチャしたデータ中のタイムスタンプの差を取ることで片道遅延の値を算出する．

```
<vm name="Router0" type="Router">          } UML name
 <if name="eth0" net="ethline0"></if>      } I/F name
  <configuration>                            Socket name
   <interface id="0" type="eth">
    <ip>
     <address>10.0.0.1</address>           } IP Address
     <mask>255.0.0.0</mask>                } Netmask
    </ip>
    <if_qos>
     <bandwidth unit="Mbps">100</bandwidth>
     <scheduling type="cbq" class="10">
      <rate unit="Kbps">1500</rate>        } QoS
      <match type="dscp">46</match>
     </scheduling>
    </if_qos>
   </interface>
  </configuration>
</vm>
```

図 2　仮想ネットワーク定義ファイルの例

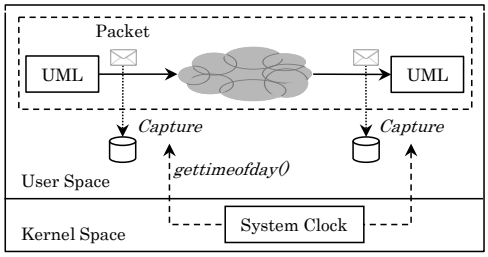

図 3　時刻を同期したパケットキャプチャ

遅延変動の計算には片道遅延の測定値を用いる．遅延変動の一般的な算出方法として，遅延時間の分散または標準偏差，IETF RFC1889 の Jitter，ITU-T 勧告 Y.1541 の IPDV(IP Packet Delay Variation)がある[21]．今回，これらすべての算出方法を実装した．ユーザが使用する際は任意の算出方法を選択する．

パケット損失率に関しては，送信側と受信側で記録したキャプチャデータを比較し，通信経路中にパケットが破棄されたかどうかを求める．

スループットは，キャプチャしたデータのタイムスタンプとパケット長から，単位時間当たりに流れたデータ量を求める．

5.4.2 トラフィック生成によるアクティブ測定

実際のネットワークではリアルタイム性のあるトラフィックやそうでないものなど様々なトラフィックが流れている．柔軟に QoS を検証するためには，仮想ネットワーク上でも様々なトラフィックを利用できることが望ましい．

このような要件を満たすために，仮想ネットワーク上で動作するトラフィックジェネレータを実装する．このトラフィックジェネレータは，パケットの IP ヘッダフィールドやパケット長などを変更することで，任意のトラフィックを生成する．これによって，アクティブ測定が可能になりより柔軟な QoS 検証を実現する．

6. 性能評価

本章では，仮想機器のメモリ使用量と仮想機器間のスループットを評価する．なお，実験には表 1 に示す環境を用いた．

表 1　実験に使用した環境

CPU	Intel Core i7 930 @ 2.8Ghz
Memory	4GByte
M/B	Asus P6T-SE
OS	Ubuntu10.04 Linux 2.6.31-41-generic-pae
UML	Debian GNU/Linux 5.0 Linux2.6.18

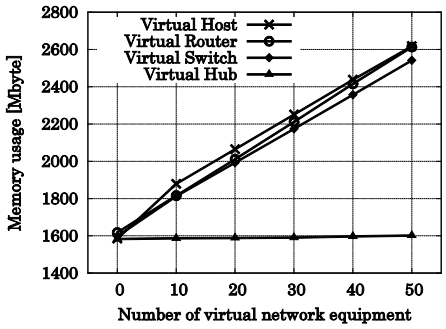

図 4　仮想ネットワーク機器のメモリ使用量

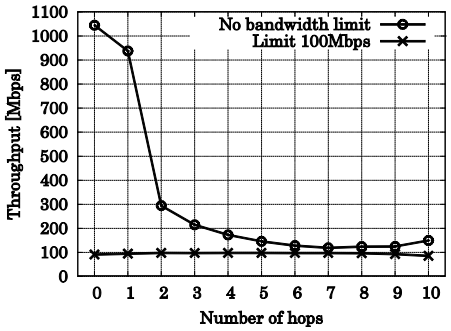

図 5　仮想ネットワーク上のパケット転送性能

6.1.1 仮想ネットワーク機器のメモリ使用量

仮想機器を複数起動させた際の，ホスト OS 側のメモリ使用量を計測した(図 4)．グラフから分かるように，仮想ハブ以外の仮想機器のメモリ使用料は起動台数に比例しており，1 台当たり約 18～20Mbyte 占有していた．仮想ハブは，パケット転送機能のみを有する uml_switch プロセスであるため，1 台当たり数百 Kbyte と少ない使用量であった．

6.1.2 仮想ネットワーク上のパケット転送性能

仮想ルータ上に nuttcp[22]を導入し，パケット転送性能を評価した．実験では，仮想ルータを直列に繋ぎ RIP による仮想ネットワークを構築し，徐々にホップ数を増やしてスループットを計測した(図 5)．結果からは，直接接続した 0 ホップでは 1Gbps を超える数値が出ているが，ホップ数が増えるにつれ，スループットが減少していることがわかる．

tc-tbf[20]によって仮想ネットワーク全体の帯域幅を 100Mbps に制限したところ，ホップ数が増加しても安定したスループットが出ることを確認した．そのため，実際に本システムを安定して動作させるには，仮想ネットワークの帯域幅を 100Mbps 程度に制限するのが妥当であると考えられる．

7. おわりに

本研究では，仮想マシンで構成する仮想的なネットワークを用いて，ネットワーク QoS を検証するシステムを開発した．本システムは，1 台の計算機上に DiffServ，IEEE802.1p を適用した仮想ネットワークを構築し，QoS パラメータを測定することができる．

今後の予定として，より汎用的な QoS 検証環境の構築を目的に，IntServ アーキテクチャの適用や IPv6 及び MPLS ネットワークにおける QoS 制御，QoS ルーティングを可能とするエミュレーションシステムを開発していく．

参考文献

[1] Y. Chen, T. Farley, and N. Ye, “QoS Requirements of Network Applications on the Internet”, *Information Knowledge Systems Management*, IOS Press, Vol.4, No.1, pp.55-76 (2004).
[2] The Network Simulator ns-2, http://www.isi.edu/nsnam/ns/ (Online).
[3] D. Mahrenholz and S. Ivanov, “Real-Time Network Emulation with ns-2”, *8th IEEE International Symposium on Distributed Simulation and Real-Time Applications (DS-RT)*, pp.29-36 (2004).
[4] OPNET Modeler, http://www.opnet.com/ (Online).
[5] N. Takeshi, I. Akihiko, O. Kozo, and K. Shinji, “QoS Analysis of Real-Time Distributed Systems Based on Hybrid Analysis of Probabilistic Model Checking Technique and Simulation”, *IEICE Transactions on Information and Systems*, Vol.E94-D, No.5, pp.958-966 (2011).
[6] Y. Sakumoto, R. Asai, H. Ohsaki, and M. Imase, "Design and Implementation of Flow-Level Simulator for Performance Evaluation of Large Scale Networks", *15th Annual Meeting of the IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, pp.166-172 (2007).
[7] X. Jiang and D. Xu, “vBet: a VM-based Emulation Testbed”, *ACM SIGCOMM Workshop on Models, Methods and Tools for Reproducible Network Research (MoMeTools)*, pp.95-104 (2003).
[8] M. Pizzonia and M. Rimondini, "Netkit: Easy Emulation of Complex Networks on Inexpensive Hardware", *4th International Conference on Testbeds and Research Infrastructures for the Development of Networks & Communities (TridentCom)* (2008).
[9] VNUML, http://www.dit.upm.es/vnuml/ (Online).
[10] F. Galán, D. Fernández, W. Fuertes, M. Gómez, and J. E. López de Vergara, “Scenario-based virtual network infrastructure management in research and educational testbeds with VNUML”, *Annals of Telecommunications*, Vol.64, No.5-6, pp.305-323 (2009).
[11] J. V. Loddo and L. Saiu, “Marionnet: A Virtual Network Laboratory and Simulation Tool”, *1st International ICST Conference on Simulation Tools and Techniques for Communications, Networks and Systems (SIMUTools)* (2008).
[12] Y. Benchaïb and A. Hecker, "VIRCONEL: A Network Virtualizer", *19th Annual Meeting of the IEEE International Symposium on Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS)*, pp.429-432 (2011).
[13] A. M. Mukwevho, J. A. van der Poll, and R. M. Jolliffe, "A Virtual Integrated Network Emulator on XEN (viNEX)", *2nd International Conference on Simulation Tools and Techniques (SIMUTools)* (2009).
[14] D. Schwerdel, D. Hock, D. Günther, B. Reuther, P. Müller, and P. Tran-Gia, "ToMaTo - a network experimentation tool", *7th International ICST Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities (TridentCom)* (2011).
[15] J. Ahrenholz, “Comparison of CORE Network Emulation Platforms”, *Military Communications Conference (MILCOM)*, pp.166-171 (2010).
[16] K. Goto, “Network Emulator with Virtual Host and Packet Diversion”, *Cyber Journals: Multidisciplinary Journals in Science and Technology, Journal of Selected Areas in Telecommunication (JSAT)*, Vol.2, No.2, pp.13-20 (2012).
[17] J. Dike, *User Mode Linux*, Prentice Hall (2006).
[18] Quagga Software Routing Suite, http://www.quagga.net/ (Online).
[19] The netfilter.org project, http://www.netfilter.org/ (Online).
[20] Linux Advanced Routing & Traffic Control, http://lartc.org/ (Online).
[21] S. Ohta, “Storage Space Saving and Error Avoidance of Passive Delay Variation Measurement Employing a Hash-Based Technique for IP Networks”, *IEICE Transactions on Communications*, Vol.J89-B, No.1, pp.10-21 (2006).
[22] nuttcp, http:// www.nuttcp.net/ (Online).