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

2005－ＱＡＩ－16（3）
2005／7／29

パケットロスパターン再現可能なネットワークエミュレータの開発

石野 正英† 前田 香織‡ 河野 英太郎‡ 岸田 崇志†

†広島市立大学大学院情報科学研究科 ‡広島市立大学情報処理センター

〒731-3194 広島市安佐南区大塚東 3-4-1

E-mail: †{ishino, takashi}@v6.ipc.hiroshima-cu.ac.jp, ‡{kaori, kouno}@ipc.hiroshima-cu.ac.jp

あらまし パケットロス，遅延，遅延の揺らぎ(ジッタ)，到着順序エラー，パケットの重複などは通信品質の低下の要因となり，アプリケーションなどに影響を与える．さらに，ネットワークの多様化により，様々なネットワークの状態を作り出すネットワークエミュレータは，アプリケーションの開発やマルチメディア端末の評価において重要な役目を果す．しかし，パケットロスのパターンに着目した時，既存のネットワークエミュレータで用いられている統計的なパラメータだけではネットワーク特性を再現することは難しい．そこで，我々は（1）モニタデータから得られるネットワークのパケットロスパターンを再現する（2）データリンク層でパケットを転送，エミュレートする，といった特徴を持つネットワークエミュレータ Lenet を開発した．Lenet はエンドユーザでも利用できるように Linux PC 上で動作するソフトウェアエミュレータである．本稿では Lenet の特徴や実装などについて述べる．また，実装した機能の実験的評価を行い，他のネットワークエミュレータと比較する．

キーワード ネットワークエミュレータ，ネットワーク特性，通信品質

Development of a Network Emulator Using Various Patterns of Lost Packets

Masahide Ishino† Kaori Maeda‡ Eitaro Kohno‡ Takashi Kishida†

†Information Science, Hiroshima City University ‡Information Processing Center, Hiroshima City University

E-mail: †{ishino,takashi}@v6.ipc.hiroshima-cu.ac.jp, ‡{kaori, kouno}@ipc.hiroshima-cu.ac.jp

Abstract Main factors of network quality are packet losses, packet order errors, jitter and delays on the Internet. Various applications are often affected by these factors. A network emulator to generate various network parameters is important in evaluations of applications, especially streaming. The existing network emulators can reproduce networks virtually by indicating parameters. Only statistical parameters like losses and jitter can not reproduce more real network behavior. Then we developed a network emulator "Lenet" which has two features. First is forwarding packets in a data link layer. Second is reproduction of packet losses patterns by log data, or user's scenario. In this paper, we describe and evaluate these functions by comparing and the existing network emulators.

Keyword Network emulator, Network behavior , Communication quality

1. はじめに

ネットワーク通信の品質低下の要因には，ネットワーク上のパケット損失(ロス)，遅延，遅延の揺らぎ(ジッタ)，到着順序エラー(入れ替え)，パケットの重複などがある．これらは様々なアプリケーションに大きな影響を与える．例えば，映像伝送などのアプリケーションでは，遅延やジッタが発生すると，再生する映像に乱れを生じる．アプリケーションの実装によっては，大きな遅延やジッタが起こった場合にパケットを廃棄してしまうこともある．

そこで，ネットワークのロスや遅延などをパラメータとして設定した値に従って，様々なネットワークの状態を作り出すネットワークエミュレータはアプリケーションマルチメディア端末などの評価において重要な役目を果す．既存のエミュレータとして，ソフトウェアで開発されたものには NISTNet[1], dummynet[2],

linee[3], 伊原らの開発したエミュレータ[4]などがある.

これらは，ユーザがパラメータを設定することによってネットワークの状態を表現しようとするが，それが必ずしも実ネットワークの状態を表しているとは限らない．これは，パラメータで設定するだけでは表現しきれない状態があるからである．このことから，モデルや統計的な値で表現できない様なネットワークの状態を再現する機能が必要であると考えた．このような機能を持つ製品として，NTTエレクトロニクスのネットワークエミュレータ[1]がある．このエミュレータはリアルタイムモニタとアナライザ機能を持ち，リアルタイムモニタで収集したデータからエミュレーションパラメータを自動生成する．しかし，高価でエンドユーザが気軽に利用するには現実的でない．

そこで我々は，エンドユーザがより忠実にネットワークの状態を再現できるようなネットワークエミュレータを開発することを目指した．開発するエミュレータの設計思想はこの製品と同様だが，今後，マルチメディアアプリケーションをエンドユーザが利用する機会が増えることを想定し，エンドユーザでも利用できるソフトウェアによるエミュレータを開発した．

開発したエミュレータ Lenet はロスしたパケットのシーケンスを入力として読み込み，再現する機能を持っている．また，ブリッジタイプで開発することにより，エンドホストはエミュレータであるホストを特別に意識することなく通信することができる．

本稿では開発した Lenet について述べる．まず 2 章では，Lenet の特徴や動作概要などを述べる．3 章で Lenet の実装について述べ，4 章では，各機能の評価実験について述べる．5 章では，NISTNet や dummynet と Lenet を比較し，Lenet の有用性について考察した．最後に 6 章でまとめと今後の課題を述べる．

2. Lenet の概要

2.1. 特徴

Lenet は 3 つの大きな特徴を持っている．

1) パケットロス再現機能

ロスしたパケットのシーケンス番号を入力することにより，ネットワークのパケットロスのパターンを再現することができる．ロスパターンは，帯域測定ツールなどで採取したモニタデータ（ログファイル）を用いたり，使用者が任意に記述することが可能である．これにより，従来のエミュレータに比べ，多様なロスのパターンを再現することができる．

[1] NTTエレクトロニクスのネットワークエミュレータ　http://www.nel.co.jp/product/emulator/

2) データリンク層でのエミュレーション

ネットワークエミュレータにはネットワーク層でパケットを転送するルータタイプと，データリンク層で転送するブリッジタイプがある．NISTNet や dummynet の様なルータタイプでは，入出力パケットを IP レベルで制御する，エミュレータをルータとして動作させなければならないので，エンドホストではデフォルトゲートウェイなどのネットワーク層での変更が必要となる．

一方，ブリッジタイプではデータリンク層でパケットを処理することで，エンドホストの OS や使用するアプリケーション，プロトコルに依存しない．またブリッジと同様に動作する為，ホストのネットワーク層の設定も不要である．Lenetはブリッジタイプで動作する．dummynet はブリッジタイプで使用することもできるが，カーネルの再構築が必要となる．Lenet ではこうした前準備は不要である．

3) RTC を使用した時刻制御

Lenet はできるだけパケット送出の時刻の精度を上げるために，時刻の制御として RTC(RealTime-Clock)[6]を使用している．

RTC は計測目的等の特別な PC を除き，ほとんどの PC に搭載されている．時刻を制御するためのハードウェアクロックで 1/2 秒から 1/8192 までの 2 のべき乗刻みのタイミングで Linux から割り込みをかけることができる．C 言語で利用できる時間を刻む関数である，sleep や gettimeofday などを用いると，CPU への負荷が非常に高くなってしまうので，Lenet では，RTC を使用している．ただし，RTC を用いて 64KHz 以上の周波数を使用するため，Lenet の実行には管理者権限(root 権限)が必要となる．

2.2. Lenet で表現できるネットワーク特性

Lenet では図 1のようなネットワーク特性を表現することが可能である．

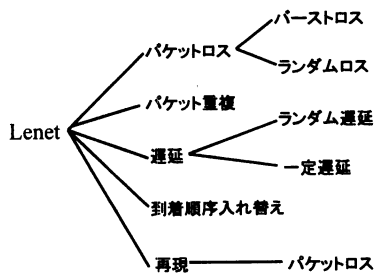

図1　Lenet で表現可能なネットワーク特性の一覧

2.3. 動作概要

2.3.1. 基本動作

Lenet の動作概要を以下に示す．Lenet の基本的な動作は図 2のようにデータリンク層でデータを転送する．

Lenet では設定ファイルにロス，遅延，入れ替え，重複の各値をパラメータとして設定し，その値に従ったネットワーク特性をエミュレートする．

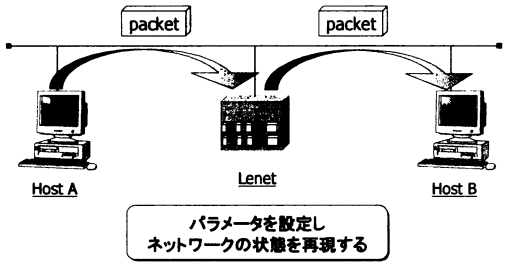

図2　Lenet の基本動作

2.3.2. パケットロスパターン再現機能の動作概要

パケットロスパターン再現機能の動作概要を図 3に示す．まず，a)でパケットロスのパターンを入力として与える．次に b)で Lenet は入力されたパターンに沿って受信ストリームを再現する

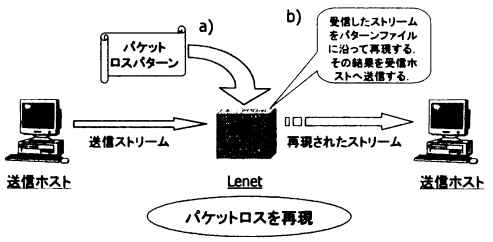

図3　パケットロスパターン再現機能の動作

3. 実装

3.1. 構成概要

Lenet の内部構成図を図 4に示す．Lenet は入力インタフェースに到着したパケットをバッファリングし，出力インタフェースからの送出を．エミュレーションパラメータに従って制御する．

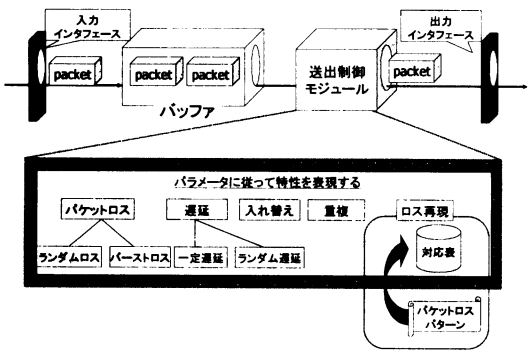

図4　Lenet の内部構成図

3.2. エミュレーションパラメータの実装

1)　パケットロス

a)ランダムロス

Lenet では送出制御モジュールで一様分布に従った乱数を発生する rand を用いてパケット毎に乱数を発生させている．ランダムロスは，Lenet で発生させた乱数がユーザが設定したパケットロス率未満の場合パケットを廃棄する．

b)　バーストロス

ランダムロスの場合と同様に，Lenet で発生させた乱数がユーザが設定したバーストロスの発生確率未満の場合，連続してパケットを廃棄する．連続して廃棄するパケットの個数は，0 からオプションパラメータとして設定された最大連続廃棄個数の間のランダムな個数である．

2)　遅延

a)　一定遅延

一定遅延は Lenet の入力インタフェーイスに到着したパケットをバッファリングし，ユーザが設定した遅延時間が経過するまで待った後，パケットを送出するように実装されている．

b)　ジッタ（ランダム遅延）

ランダム遅延は，一定遅延と同様に，到着したパケットはバッファリングして送出される．バッファリング時間はユーザが設定した一定遅延時間(tm)を中心として，0 からユーザが設定した振れ幅の最大値(⊿t)の間のランダムに決定された値分だけ上下に振れるように実装されている．各パケットに対するバッファリング時間は tm＋rand(⊿t*2)－⊿t である．

3)　順序入れ替え

ランダムロスの場合と同様に，Lenet で発生させた乱数が，ユーザが設定した入れ替え発生確率未満であれば，そのパケットをバッファに入れ，次のパケットの送信を待ってバッファにあるパケットを送出する．これにより順序入れ替えを実現している．

なお，Lenet ではランダム遅延を発生させる際に，到着するパケット間隔に対してバッファリング時間が大きいと，先に到着したパケットを後から到着したパケットが追い越してしまうため，入れ替えが発生する．

4)　パケット重複

ランダムロスの場合と同様に，Lenet で発生させた乱数が，ユーザが設定した重複発生確率未満であれば，そのパケットを 2 回送出することにより，パケット重複を実現している．

5)　パケットロスパターン再現

ロスパターンファイルを読み込み，その数字列から Lenet のプログラム内にパケットのシーケンス番号との対応表を作成する．パケットが到着するとシーケン

ス番号を割り当てた後，その対応表を上から順番に見て，そのシーケンス番号があれば，そのパケットを廃棄する．見つからなければ受信ホストへ送信する．対応表が終端になれば，対応表の始めに戻るというサイクルを繰り返す．

3.3. 開発環境と動作実績

Lenetの開発環境は，OS Linux，C言語で開発した．gccのバージョンは3.3.5である．Lenetの動作が確認できたマシンの仕様を表1に示す．

表1　動作実績

CPU	メモリ	OS	カーネルVer
PentiumⅢ 1.0GHz	512MB	Vine Linux 2.6	2.4.22
Pentium4 2.2GHz	512MB	Debian/GNU Linux sid	2.6.8
Pentium4 2.2GHz	512MB	Debian/GNU Linux sid	2.4.27
VIA Eden 667MHz	512MB	Vine Linux 3.0	2.4.26

4. 評価実験

Lenetの動作確認や精度を調べるため，いくつかの評価実験を行なった．

実験は図5のような構成で行なった．また，HostA，HostB，Emulator PCのスペックを表2に示す．

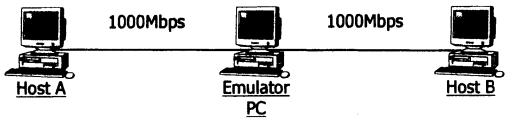

図5　実験構成

表2　マシンスペック

CPU	Host A	Emulator PC	Host B
OS	Debian GNU/Linux sid	Debian GNU/Linux sid	Debian GNU/Linux sid
CPU	Pentium4 3GHz	Pentium4 2.2GHz	Pentium4 3GHz
Memory	1GB	512MB	1GB

4.1. 最大スループット

Lenetがどの程度のスループットを出すことができるのかを調べるために，各パラメータを設定した時と転送のみの時の最大スループットをIperf[5]を用いて測定した．

ここで，最大スループットの定義は，受信側でパケットが30秒間欠落せずに受信できる帯域とした．測定の結果を表3示す．

表3 最大スループットの測定結果

測定条件	最大スループット (Mbps)
転送のみ	355
ロス 1%	350
遅延 10ms	320
重複 100%	220
入れ替え 50%	340

本測定環境では，HostAを送信，Emulator PCを受信として，Iperfを用いて最大スループットを測定したところ，560Mbpsであった．このことから，Lenetを用いた場合には最大スループットが最低でも約63%に低下する．しかし，ネットワーク特性を表現する場合でも，大きく性能が低下しないことがわかった．

4.2. 精度に関する実験

パケットロス，遅延，重複，到着順序入れ替えについてLenetがどの程度正確に動作しているかを調べるために評価実験を行なった．ロス，遅延に関してはNISTNet, dummynetと比較し，重複に関してはNISTNetと比較した．

本実験ではユーザが設定した各設定値がエミュレータを通過後の測定値とが一致しているかを評価する．このとき，一致率を以下の様に計算し，評価値として用いた．

一致率 = (設定値 − |設定値 − 測定値|)/設定値*100

一致率が100に近ければ近いほど，測定値が設定値に近い値をとったと言える．

4.2.1. パケットロス

ユーザが設定したパケットロス率でパケットロスを発生しているかを調べるために，設定ロス率を10%から90%までの10%刻みで変化させ，また，各ロス率でHostAからの送信帯域を10Mbpsから90Mbpsまで10Mbps刻みで変化させ，各設定値においてHostBでのロス率を測定した．測定にはIperfを用いた．また，同様の方法で，NISTNetやdummynetを使用して測定した．約20万パケットについて測定した際の各帯域の平均一致率を表4に示す．

表4 パケットロスの精度

帯域(Mbps)	一致率 (%)		
	dummynet	NISTNet	Lenet
10	99.73	99.94	99.72
20	99.86	99.98	99.78
30	99.77	99.92	99.81
40	99.83	99.91	99.86
50	99.79	99.95	99.79
60	99.63	99.94	99.84
70	99.74	99.95	99.89
80	99.85	99.92	99.86
90	99.81	99.92	99.83

表4より，Lenetは他のエミュレータとほぼ同様の精度を持っていることがわかった．また，Emulator PCに90Mbpsの負荷をかけ，帯域が大きくなり処理をするパケットの数が増えても精度を落とすことなく処理できていることを示す

4.2.2. 遅延

ユーザが設定した遅延を正しく発生しているかを調べるために，Lenetで10msから100msまでの遅延を

10ms きざみで発生させ，ping コマンドを用いて HostA から HostB へ ICMP の ECHO_REQUEST パケットを送り，RTT から遅延を測定した．この時，測定値と設定値が一致するかを確認する．同様の方法で NISTNet と dummynet でも測定し，各遅延時間ごとの 50 回の平均を表 5 に示す．

表5　遅延の精度に関する測定結果

設定遅延時間(ms)	平均値(ms)		
	dummynet	NISTNet	Lenet
10	9.92	10.38	10.41
20	19.67	19.67	20.41
30	29.82	30.44	30.44
40	39.90	40.42	40.45
50	49.65	50.51	50.51
60	59.93	60.45	60.44
70	69.85	70.32	70.40
80	79.75	80.35	80.35
90	89.87	90.39	90.43
100	99.80	100.37	100.37
平均一致率(%)	99.49	98.86	98.76

表 5から，Lenet の持つ遅延発生機能は，NISTNet，dummynet と同等の精度を持っている事を確認した．

4.2.3. パケット重複

ユーザが設定したパケット重複率でパケット重複が発生できているかを調べるために，HostA から HostB へ Iperf を用いて 40Mbps で測定を行なった．このとき，Emulator PC でパケット重複率を 10%から 50%の間の 10%刻みの値に設定して，Lenet と NISTNet の各設定値ごとの重複率を測定した．それぞれ 20 万パケットについて測定した結果を図 6に示す．

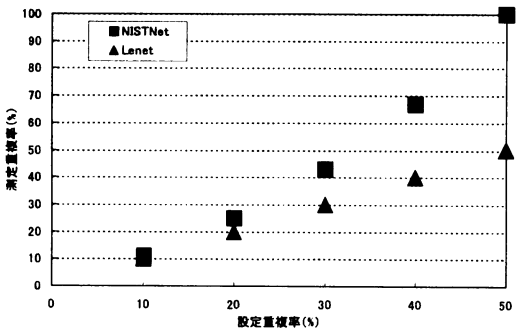

図 6　重複発生の設定値と測定値

図 6から，Lenet は設定値と測定後の重複率がほぼ一致しているのに対して，NISTNet ではずれている．設定したパラメータは Lenet と同様であるにもかかわらず，測定値と設定値が異なる原因は不明であり，現在考察中である．

4.2.4. 順序入れ替え

ユーザが設定した入れ替え発生確率通りにパケットの到着順序入れ替えが発生しているかどうかを調べるために，HostA から HostB へ Iperf を用いて 90Mbps で測定を行なった．このとき，Emulator PC で入れ替わる確率を 10%から 50%の間の 10%刻みの値に設定して，それぞれの設定値の時の入れ替え発生確率を測定した．測定はそれぞれ 20 万パケットに関して行なった．結果を表 6に示す．

表6 各設定値ごとの一致率

設定値(%)	10	20	30	40	50
一致率(%)	99.50	99.85	99.77	99.90	100.00

この項目に関しては，NISTNet，dummynet ともに実装されていないので，比較は行っていない．

4.3. パケットロスパターン再現の動作確認

Lenet に入力するパケットロスパターンとしてロスが規則的に発生した場合とランダムに発生した場合のデータを用いた．なお，規則的に発生した場合には，0，5，10…の様に 5 ずつ増加するデータを，ランダムに発生した場合のデータには perl の rand 関数で生成したデータをそれぞれ用いている．この時のパケットロスパターンはロスしたパケットのシーケンス番号列である．ロスしたパケットのシーケンスを出力する機能を持つ Sperf[7]を用いて，送信ホストから 10Mbps でストリームを送信し，受信ホストでそのストリームのロスしたパケットのシーケンスを記録する．送信ホストから送信されたストリームの中の 10000 個のパケットについて，入力したパケットロスパターンと受信ホストでの記録を比較し，一致するかを調べた．その結果を図 7 に示す．

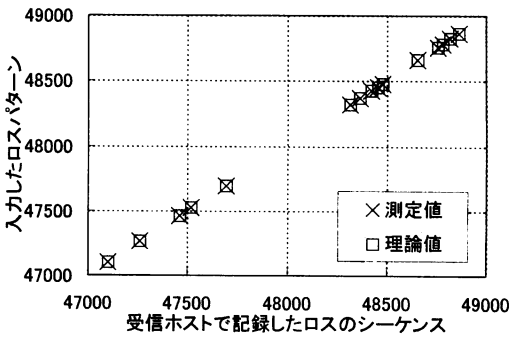

図7　入力データがランダムな場合の測定結果

結果から，10000 個すべての一致が確認できた．また，実ネットワークを用いた実験も行ない，入力したロスパターンが再現できることを確認した．

5. 考察

4.における評価実験から，Lenet の機能が NISTNet, dummynet とほぼ同等の精度を持っていることがわかった.また,パケットロスパターン再現機能に関して，実ネットワークを用いて,再現できることを確認した.

ここで，NISTNet，dummynet と Lenet の比較を行った．使用する OS やどの層でパケットをエミュレートするかなどをまとめたのが表 7である．また，表 8 はネットワーク特性を表現する際の各エミュレータでのパラメータの設定方法をまとめたものである.

表7 ネットワーク特性の表現以外での比較

比較項目	NISTNet	dummynet + ipfw	Lenet
処理をする層	ネットワーク層	ネットワーク層（ブリッジも可）	データリンク層
必要な作業	モジュールとしてコンパイル	カーネルのコンパイルが必要	特になし
OS	Linux	BSD	Linux
IPv6対応	×	ブリッジタイプの場合は○	○

表 7から他のネットワークエミュレータに比べると，Lenet や dummynet が IPv6 をエンドホスト間で使用可能である．これはデータリンク層でパケットを処理していることによる．ネットワーク層以上ならば，別のプロトコルであっても，エンドホストは特に気にすることなく使用することができる.

また，表 8から Lenet にはキューイングアルゴリズムが実装されていないが，パケットロスパターンや順序入れ替えなどの NISTNet や dummynet が設定できないネットワーク特性を設定できる．Lenet は NISTNet や dummynet が表現していない部分を補完していると言える.

6. おわりに

本稿では，エンドユーザが利用すること想定したネットワークエミュレータ Lenet の実装とその評価について述べた.

現在，再現機能の中でロスを再現する部分を実装が完了している．今後は遅延の再現機能を実現して行く予定である.

謝辞

Lenet の開発において，ご助言を頂きました南山大学後藤邦夫教授,松下電器産業株式会社 ネットワーク開発センター小西一暢氏，北陸先端科学技術大学大学院の宮地利幸氏に感謝いたします．また，日頃から御指導頂く広島市立大学情報科学部石田賢治教授に感謝します.

文　献

[1] National Institute of Standards and Technology (NIST),"NISTNet",http://www.antd.nist.gov/tools/nistnet/index.html/, 2005.

[2] LuigiRizzo,"dummynet",http://info.iet.unipi.it/textasciitilde luigi/ip_dummynet/, 2005.

[3] NTTPCcomunications,"linee",http://www.pc-view.net/Network/030110/

[4] 伊原亜希，村瀬進哉，“ネットワーク障害の模倣を目的とする仮想ルータの設計と実現”，南山大学数理情報学部情報通信学科 2004 年度，卒業論文要旨集，2005. (http://www.seto.nanzan-u.ac.jp/msie/gr-thesis/it/proc/2004/)

[5] A. Tirumala,F. Qin, J. Dugan,J. Ferguson,and L.Gibbs,“Iperf”, http://dast.nlanr.net/Projects/Iperf/, 1999.

[6] 近堂徹，大塚玉記，西村浩二，相原玲二，“MPEG2 over IP 伝送システム mpeg2ts の開発と性能評価”,DICOMO 2002 シンポジウム論文集,pp.157-160, 2002.

[7] 川上貴宏，岸田崇志，河野英太郎，前田香織，“広帯域ネットワークにおける IP ストリーム伝送性能評価ツールの開発”，信学技報，pp.25-30, IA2004-5, 2004.

表8　エミュレーションパラメータの設定方法

比較項目	NISTNet	dummynet + ipfw	Lenet
ランダムロス	廃棄率	廃棄率	廃棄率
バーストロス	△※	△※	発生確率＋最大連続損失個数
固定遅延	遅延時間	遅延時間	遅延時間
ジッタ	発生させるジッタの平均値	×	固定遅延＋最大振れ幅（ms）
順序入れ替え	×	×	入れ替え発生確率
パケットの複製	重複発生確率	×	重複発生確率
帯域制限	限界帯域	限界帯域	限界帯域
パケットロス再現	×	×	パケットロスパターンファイルを入力
キューイングアルゴリズム	DRD	RED, GRED(gentle red)	×

※キューイングアルゴリズムによる最大連続損失個数は指定可能