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

石野　正英†　　前田　香織††　　河野英太郎††　　石田　賢治†

† 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@v6.ipc.hiroshima-cu.ac.jp, ishida@ce.hiroshima-cu.ac.jp,
††{kaori,kouno}@ipc.hiroshima-cu.ac.jp

あらまし　ネットワークプロトコルやアプリケーションの開発時には，実ネットワークや仮想的なネットワークを用意してそのテストを行っている．我々は，このような目的で使うネットワークエミュレータ「Lenet」を開発している．Lenet はパケットをキャプチャするなどで得られたシナリオを再現することで，ネットワーク特性をより柔軟に表現する機能を持つ．本論文では，特に近年利用が増加しているリアルタイムアプリケーションに着目し，その制御情報を用いて，エミュレーションを行う方法について提案する．これを実現するために，RTP の制御情報である RTCP を用いてシステム設計を行った．また，エンドユーザの利便性を向上するための Lenet の GUI システムの開発についても述べる．

Design of Capture and Replay Function Using RTCP of A Network Emulator Lenet

Masahide ISHINO†, Kaori MAEDA††, Eitaro KOHNO††, and Kenji ISHIDA†

† Graduate School of Information Sciences, Hiroshima City University
†† Information Processing Center, Hiroshima City University
3-4-1, Ozuka-higashi, Asa-minami, Hiroshima, Japan
E-mail: †ishino@v6.ipc.hiroshima-cu.ac.jp, ishida@ce.hiroshima-cu.ac.jp,
††{kaori,kouno}@ipc.hiroshima-cu.ac.jp

1. はじめに

ネットワークプロトコルやアプリケーションを開発する際にはテスト環境として，シミュレータやエミュレータが利用される．我々もこのような目的でネットワークエミュレータ Lenet [1] [2] を開発している

近年では，エンドユーザがリアルタイムアプリケーションを用いて，ストリーム配信をする機会が増加している．こうした背景からシミュレータとして，ProLab [3] が開発されている．これは，SIP や H.323 の呼制御のシミュレーションだけでなく，RTCP レポートのモニタリングや RTP パケットの統計情報が取得できる．

そこで，我々は，このようなリアルタイムアプリケーションの制御情報を，主に IP 電話や動画伝送などのストリームアプリケーションの開発やテストのためのネットワークエミュレーションに使うことを考えた．本稿では，これを実現するために，RTP(Real time Transport Protocol) の制御情報である RTCP(Real time Transport Control Protocol) に着目し，それを用いたキャプチャとリプレイの方法を提案する．RTP と RTP の制御を行う RTCP は RFC3550 [4] で示されている，リアルタイムアプリケーションを利用する際に，広く用いられているプロトコルである．特に，近年では，VoIP(Voice over IP) などの音声通話アプリケーションに多く利用されている．また，この機能を Lenet に盛り込んだシステムの拡張について述べる．その詳細を 4 章で述べる．

さらに，Lenet はエンドユーザが利用することを目的としており，エンドユーザの利便性を高めるための GUI(Graphical User Interface) を用いたシステムを構築した．その詳細を 3 章

で述べる．

2. Lenet の概要

2.1 主な特徴

Lenet は以下のような特徴をもつソフトウェアエミュレータで Linux PC で動作する．

1) データリンク層でのエミュレーション

Lenet はパケットをブリッジのようにデータリンク層で処理する．

2) RTC を用いたタイマ管理

Lenet はハードウェアクロックである RTC（Real Time Clock）を用いて，正確で，CPU 負荷の少ないタイマ管理を行う．

3) ユーザランドアプリケーション

ユーザが Lenet を利用するのに必要な導入プロセスをできるだけ簡潔にすること，またシステムの可搬性，移植性などを考え，ユーザランドのアプリケーションとして開発した．

4) パケットキャプチャ・リプレイ機能

Lenet はキャプチャ機能を持っており，その時々のネットワークの特性を取得することが可能である．この機能を利用することで，モデルなどでは表現が難しいネットワーク特性をエミュレートできる．Lenet は汎用の PC(Linux) で動作し，1 台の PC でパケットの情報をキャプチャし，その情報をもとにネットワークの動きをリプレイすることができる．

Lenet は 3 つの方法でリプレイを行う．1 つ目が，パケットをキャプチャし，それをシナリオとしてリプレイするものである．2 つ目はユーザが作成する時間毎にパラメータが切り替わるタイムテーブルをシナリオとしてリプレイする．3 つ目が統計的なパラメータを用いてリプレイするものである（遅延のみ）．

5) ユーザが利用しやすい GUI

ユーザの利便性を向上するため，コマンドによる操作だけではなく，GUI を備えている．

2.2 システムモデル

Lenet のシステムモデルを図 1 に示す．Lenet のエミュレーションは大きく分けて，2 つのモードを持っている．1 つは emulation モードで，ネットワークエミュレータの一般的なパケット損失率や遅延などのパラメータに従って，エンド-エンド間のネットワーク特性を表現するものである．

もう 1 つの replay モードではパケットをキャプチャし，その情報を基にネットワーク特性を再生するものである．このモードでは，図 1 の Host A から送出されたパケットは Lenet でキャプチャされ，その情報はキャプチャファイルに書き出される．その後，Lenet で扱う為のフォーマットに変換され，シナリオファイルとして出力される．このシナリオファイルはパケットのキャプチャによる生成以外に，ユーザが作成可能である．再生する時は，シナリオファイルに従って，送出制御モジュールにおいて，アプリケーションのトラフィックをコントロールする．

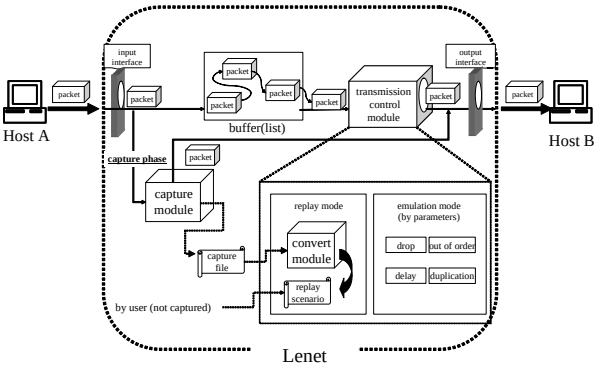


図 1　システムモデル．

表 1　Lenet の動作が確認できた PC の仕様．

CPU	memory	OS	kernel version
PentiumIII 1.0GHz	512MB	Vine Linux 2.6	2.4.22
Pentium4 2.2GHz	512MB	Debian/GNU Linux sid	2.4.27
			2.6.8
Pentium4 2.6GHz	512MB	Debian/GNU Linux sid	2.4.27
			2.6.12-smp
			2.6.11
ARM920Tid 200MHz	64MB	Debian Linux 3.4.3-13	2.6.12.3 -a9-1
AMD AU1550(TM) 400MHz	128MB	Debian Linux sarge	2.6.12
VIA Eden 667MHz	512MB	Vine Linux 3.0	2.4.26
		Linux Knoppix	2.4.27-knx-2

2.3 アーキテクチャと開発環境

図 2 に示すのは Lenet のアーキテクチャである．Lenet は GUI などから作成された設定ファイルからパラメータを取得する．リプレイシナリオはキャプチャもしくは，ユーザによって作成される．Lenet は C 言語によって開発している．(gcc のバージョンは 3.3.6)

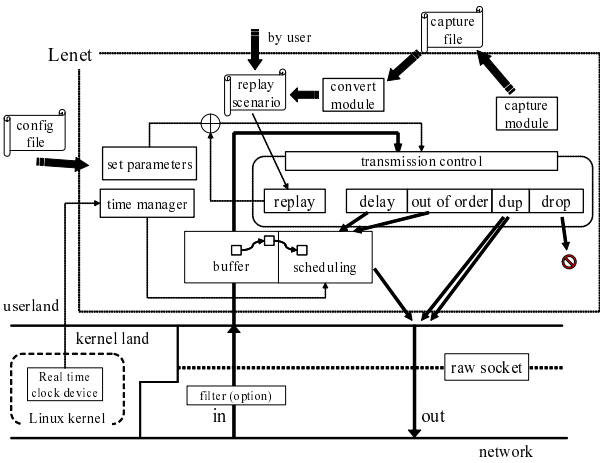


図 2　アーキテクチャ．

表 1 に Lenet の動作を確認した PC の仕様を示す．

3. Lenet の GUI システム

3.1 概　　要

Lenet の GUI を利用するシステムの概要を図 3 示す．

このシステムはコマンドを GUI から送信するクライアント，それを受け取り Lenet への指示等を担当する manager デーモ

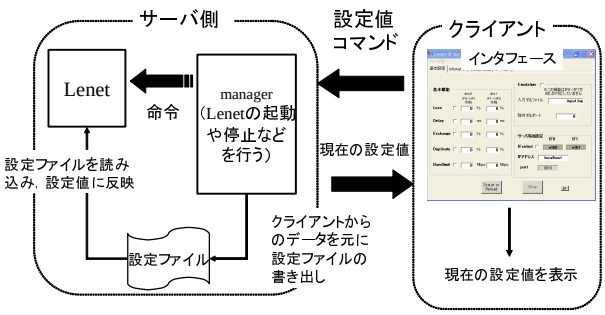


図 3　GUI を利用するシステムの概要.

ンと Lenet 本体で構成されている．クライアントはサーバ側と manager を介して通信を行う．

このシステムで用いるプロトコルは一部，遠隔機器制御プロトコル RACP [8] の設計を基とし，図 4 に示すようなパケットフォーマットとした．また，このシステムで用いる命令を表 2 に一覧を示す．

この GUI によって出来ることは以下のようなものである．

1)　Lenet の遠隔オペレーション

遠隔にある Windows PC から，ネットワーク越しに Lenet の起動，停止，設定ファイルの再読込等の操作ができる．これは，コマンドラインからではなく，視覚的に操作することができ，Unix に対する知識がなくても操作可能である．

2)　設定ファイルの生成

テストを繰り返していると，以前利用した設定ファイルを再利用したいという場面がある．このような時に，GUI による設定内容をローカルのファイルに書き出したり，サーバにアップロードする機能が利用できる．また，今現在アップロードされているファイルのリストも取得できる．こうすることで，様々な場面のパラメータを保存し，必要な時に利用することが可能となる．

図 4　Lenet 制御パケットのフォーマット.

表 2　コマンド一覧.

object	command	第 1 引数	第 2 引数	説明
LENET	INIT	/		クライアントをサーバーのリストに登録
	LIST	/		自分がサーバにアップしている ファイルの一覧を取得
	START	/		Lenetの起動要求
	RELOAD	/		設定ファイルの再読込要求
	UPLOAD	file名	file	ファイルのアップロード
	TRAP	/		Lenetから設定変更の通知
	STOP	/		Lenetの停止
	QUIT	/		クライアントをサーバのリストから削除 登録されたファイルもすべて削除

3.2　動 作 詳 細

図 3 の動作を詳細に述べる．まず，設定値をクライアントから manager にアップロードする．manager は，この設定値を設定ファイルとして出力する．このとき，クライアントの IP アドレスとアップロードされた設定ファイル名が manager の利用者データベースに登録される．アップロードしたファイルはクライアントごとに管理され，クライアントからの要求に応じて manager はファイルの一覧を送る．

次に，クライアントは表 2 に示すような命令を manager に出すことができる．manager はその命令に従って，Lenet の起動，再起動，設定ファイルの再読込などをおこなう．また，Lenet は現在の設定値を manager を介して，リアルタイムにクライアントに知らせる．

また，一連の動作を行うときのクライアントの処理のフローチャートを図 5 に示す．

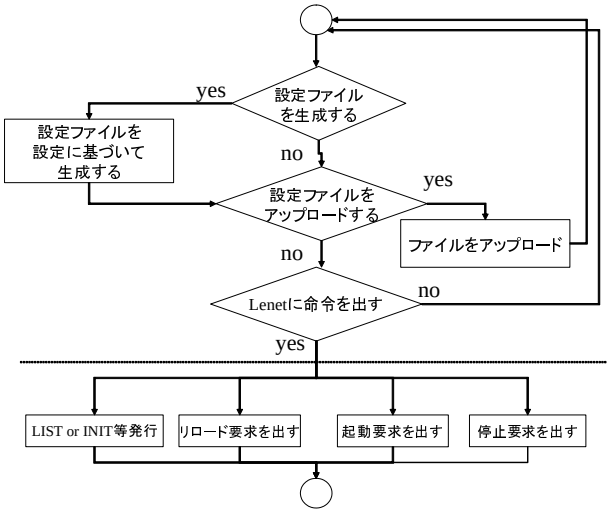


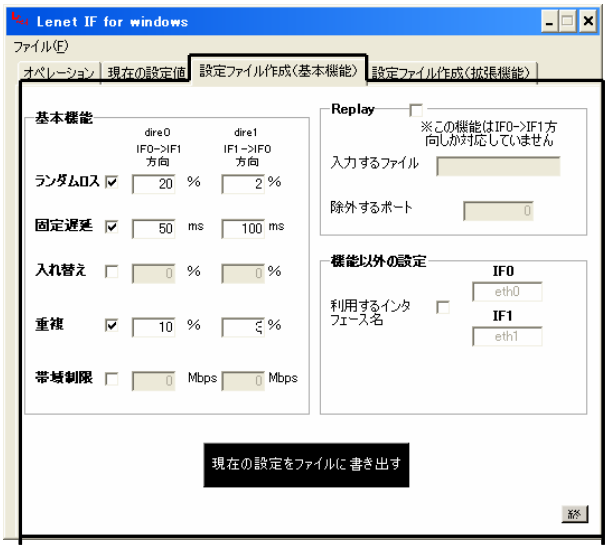
図 5　クライアントの処理.

3.3　実　　装

図 6 に示すような GUI を Microsoft VB.NET(注1)によって開発した．Windows によって動作する．ただし，.NET Framework(注2)が必要である．図 6(a) はパラメータを設定する画面である．この画面でパケット損失率や遅延などのパラメータをセットし，設定ファイルを生成する．図 6(b) は manager への命令を発効する画面である．生成した設定ファイルを manager にアップロードしたり，Lenet の起動，停止などの命令を発効する．また，図 6(b) にも示すように,「現在の設定値」のタブを選択すると，現在 Lenet で設定されている値を確認することができる．

これにより，Lenet が動作する Linux PC そのものではなく，遠隔から設定できる．また，Windows PC から容易に操作できる．

(注1)：Microsoft Visual Basic .NET は Microsoft の商標登録
(注2)：Microsoft .NET Framework は Microsoft の商標登録



(a) 設定ファイル生成時の画面

(b) 命令発効時の画面

図 6　Lenet の GUI.

4.　キャプチャ・リプレイ機能

4.1　現行の機能

2.1 節の 4) の中で述べた，キャプチャ・リプレイ機能に関して，以下の 2 つの実装が完了している．各々の手法について述べる．

1)　ユーザ作成のリプレイシナリオを用いるリプレイ

ユーザはパケット損失率や遅延などのパラメータとその切替時間を記述して，リプレイシナリオを作成する．Lenet はそのシナリオに従って，時間毎にパラメータを切り替える．

2)　統計的なパラメータを用いる遅延の分布のリプレイ

静的な遅延の分布の生成の手法は，Lenet では NISTNet [5] とほぼ同様である．NISTNet は前もって生成してある，逆関数テーブルをランダムに参照する．この手法は，累積分布関数を基に考えられたものである [5]．事前準備として，インターネットなどで，ping の RTT(Round Trip Time) を測定した結果のテーブル (以下 ping テーブル) を用意しておく．Lenet での逆関数テーブルを生成するまでのプロセスは NISTNet と同様である．

4.1.1　ユーザ作成のリプレイシナリオを用いるリプレイ

4.1 節の 1) の場合の動作確認実験の結果を図 8 に示す．実験は図 7 における Host A から Host B に ping コマンドを 100ms 間隔で実行した結果を示している．図 7 の PC のスペックは表 3 に示す．このとき，Lenet では用意したシナリオに従って遅延を変化させた．そのシナリオは，1 秒間隔で，(1ms, 10ms, 100ms, 1000ms, 100ms, 1ms, 1000ms, 1ms) の順に遅延を変化させるものである．NISTNet，dummynet [6] に関しても，スクリプトを用いて，同様に実験を行った．この実験で，図 8 に示すように dummynet(図 8(a)) は，Lenet(図 8(b)) と比較すると，遅延の大きな変化に対応できていないことがわかった．NISTNet に関しては，Lenet と同様の結果であった．

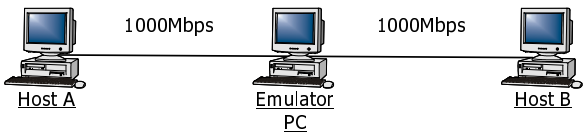


図 7　実験環境.

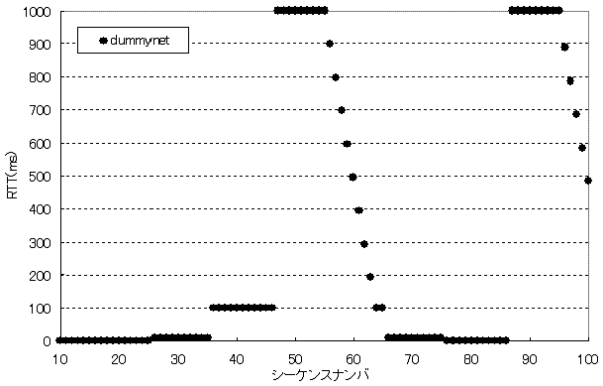
表 3　実験 PC の仕様

	HostA	EmulatorPC	HostB
CPU	Pentium4		
	3.0GHz	2.6GHz	3.0GHz
OS	Debian/GNU Linux sid		
memory	1GB	512MB	1GB

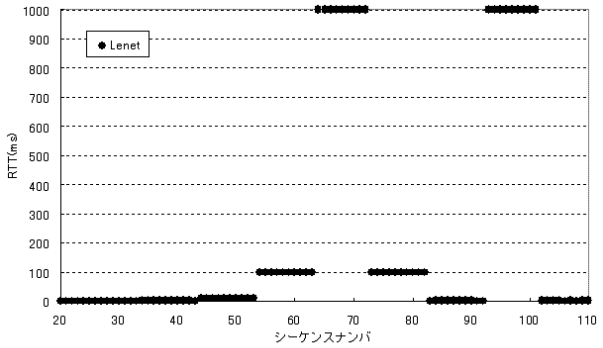
4.1.2　統計的な遅延の発生

4.1 節の 2) の場合の実装と評価実験結果を以下に示す．NISTNet は逆関数テーブルを参照するために，乱数を生成するが，その際に，パケットの前後間の遅延時間の相関性を示す値，相関係数を重みとしてユーザが与えることができる（デフォルトは 0）．Lenet の実装ではこの部分が異なっており，相関係数は利用せず，常に 0 とした上で，乱数の生成に一様乱数生成関数 (rand ()) を利用している．その理由として，逆関数法の定義から，逆関数テーブルの参照には一様乱数を用いることが前提なので，逆関数法を用いて分布を再現するのであれば，rand() で十分であると考えたからである．

Lenet では，ユーザはパラメータに平均値と偏差を入力する．特定の遅延の分布を得たいときは，ユーザが目的に合った ping テーブルを用意する必要がある．しかし，もし，ユーザが用意しなかった場合にはデフォルトで用意されている ping テーブ



(a) dummynet



(b) Lenet

図 8　リプレイシナリオを用いた遅延の動的変化.

ルに基づいて，逆関数テーブルが生成される．

この方法を用いた場合の動作確認実験の結果を以下に示す．図 9 に示すのは著者のうちの 1 人の自宅と大学の間のマシンとの RTT を ping によって，3 時間測定した遅延の分布を NISTNet と Lenet に入力し，その分布を再現した結果である．自宅側は ADSL(24Mbps)，大学はダークファイバ (100Mbps) で接続されている．その間のインターネットには少なくとも 16 個のルータがあり，少なくとも 4 つの ISP を経由している．図 9 の (1) が取得した ping データの分布，(2) が NISTNet による再現，(3) が Lenet による再現である．ここで，入力したパラメータは取得した ping データの平均値と偏差でそれぞれ，56.90ms と 3.01ms である．また，NISTNet に入力する相関係数は 0 である．

実験結果より (2) と (3) について，平均値と偏差を計算すると，(2) は平均値 57.18ms，偏差 2.49ms，(3) は平均値 57.09ms，偏差 2.55ms であった．これらと (1) を比較すると，Lenet の分布は NISTNet よりも測定分布に近い結果になっていることがわかる．その理由として，逆関数法を利用する上で，乱数を生成するときに，Lenet では rand() を用いており，これが一様乱数により近いためではないかと考える．ただし，この分布はあくまでも，遅延発生の統計的な度数分布であり，遅延の発生状況そのものを表現しているわけではない．

4.2　RTCP によるキャプチャ・リプレイ機能

本節では現行の Lenet に追加する，RTCP を用いるキャプチャ・リプレイ機能について述べる．初めに RTCP を用いたキャプチャ・リプレイの動作を図 10 に示す.ping などのコマンドで得られる RTT や，RTP のパケットのシーケンスナンバからパケット損失を，また，RTCP レポートからジッタなどの情報を得て，リプレイシナリオが生成される．そして，リプレイ機能によってリプレイシナリオを何度でも再生することができる．

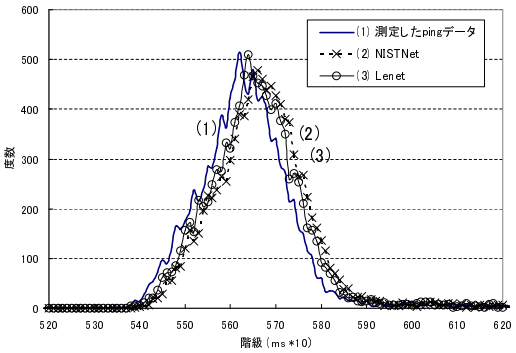


図 9　遅延分布の生成：実測値, Lenet，NISTNet.

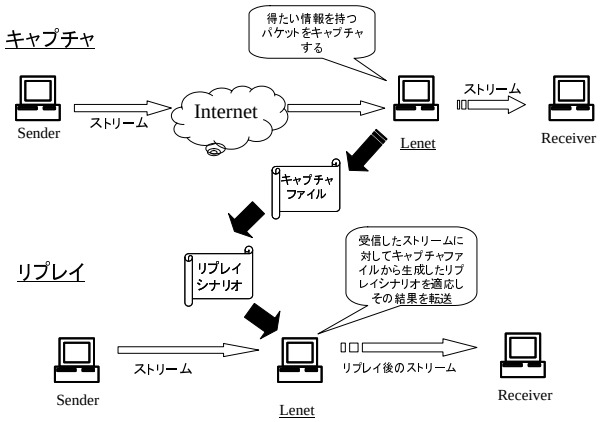


図 10　キャプチャ・リプレイ機能の動作.

4.2.1　RTCP の情報の利用について

RTP はメディアデータとして送信者から受信者に送出される．このプロトコルは，メディアデータをできるだけ遅延を少なく再生しようとして，リアルタイム性に重きを置いているため，同期に必要な時間情報（タイムスタンプ）しか持っていない．

一方，RTP の制御を行 RTCP は RTP の持つ個々のパケットの時間情報以外の，例えば，リップシンク（映像上の人物の口の動きとセリフや歌の音声の同期をとること）等の動作に必要な時間情報やジッタ，パケット損失率等の情報を提供する．RTCP はリアルタイムアプリケーションの品質を示す多くの情報を持っており，その情報を利用することはアプリケーションにとって，非常に効果的と考える．また，RTP の情報を Lenet のキャプチャ・リプレイ機能で正確に利用すると，情報量が膨大となり，ハードウェアの制約等からエミュレーション可能な時間が制限されてしまう問題もある．制御情報である RTCP レポートは RTP パケットに比べて，データ量としては小さい

ので，そのような制約を受けることは少ない．

また，パケットダンププログラムである tcpdump を用いてパケットのシーケンスナンバやタイムスタンプを取得したとしても，それは IP 層レベルでの測定値であって，トランスポート層以上の測定値ではない．さらに，ジッタなどの情報は IP 層レベルでは計算することが困難である．そのため，取得した情報をアプリケーションで利用する場合には，アプリケーションにより近い層の情報を利用することが良いと考える．

このようなことから，リアルタイムアプリケーションの制御情報として，特に RTCP に着目し，それをリプレイシナリオとして利用することとした．

4.2.2 RTCP によるキャプチャ・リプレイの設計

本節では現行機能に加えて，Lenet に追加した RTCP の情報を使って，エミュレーションを行うシステムの設計を述べる．

RTCP レポートの内の RR と SR を使って Lenet は以下に示す情報を生成する．ここで，RR(Receiver Report) は RTCP レポートの内，受信者から送信者に送信されるレポートで，受信者での受信品質に関する情報を伝えるために利用される．SR(Sender Report) は RTCP レポートの内，送信者から受信者に送信されるレポートで，送信者からのストリームに関する情報を伝える．

1) パケット損失率
2) 累積損失パケット数
3) ジッタ
4) RTT

1) から 3) は RTCP の RR のそれぞれのフィールドから取得することができる．4) のに関しては，本来，送信者の PC で計測するため，通常 Lenet の PC で，キャプチャによって RTT を計算すると，送信者の PC と Lenet の PC のクロックスキュー（PC 毎の時間を刻むタイミングのずれ）が生じてしまう．

また，RTCP は RTP のようなシーケンス番号に該当するものを持っていない．そこで，インターネット上で発生する RTCP パケットの入れ替わりに対処するため，Lenet では SR の NTP タイムスタンプフィールドと RR パケットの最新の SR のタイムスタンプが格納されている LSR(last SR) フィールドから取得した値をそれぞれ SR と RR の識別子（ID）として利用する．

Lenet では以下の方法で RTT を測定する．

図 11 に SR と RR の通過と RTT の測定のための時間計測のタイミングを示す．図 11 の TST は SR パケットの送信時刻 (NTP タイムスタンプフィールド)，TRT は RR パケットの受信時刻，TSR は Lenet での SR パケットの通過時刻，TRR は Lenet での RR パケットの通過時刻をそれぞれ示している．また，DLSR（delay since last SR）は受信側で SR パケットを受け取ってから，RR パケットを送出するまでの時間であり，その値は RR パケットに格納されている．Lenet は SR，RR パケットの各フィールドから DLSR などの必要な情報を読み出し，利用する．

本提案の前提条件は，Sender と Lenet が同一 LAN 上にあり，Sender と Lenet の間に遅延がほぼ無いことを想定している．これにより，SR の送出時刻や RR の到着時刻を知らなくても RTT を算出可能である．RTT の算出方法としては，

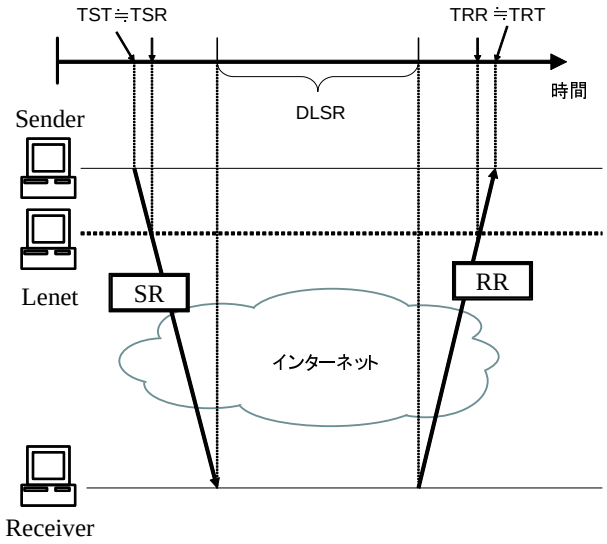


図 11　RTT の測定.

$$RTT = TRR - TSR - DLSR \tag{1}$$

である．

4.2.3 RTCP 利用の制限

RFC3550 に示されているように，RTCP をマルチメディアアプリケーションに実装する場合，RFC に従う実装は義務づけられていない．例えば，DV ストリームを扱う DVTS [9] では RTCP は実装されているが，その RR のパケット損失率と累積損失パケット数の情報しか利用されていなかった．さらに，DVTS をエンドシステムとして利用し，Lenet を介して通信を行う際に，Lenet でジッタを発生させたところ，そのパケット損失率と累積損失パケット数も受信側での映像品質とかけ離れた値をとっていたことを確認した．これは DVTS がエラーコンシールメント（ロスしたところの差分を前の画像で埋める処理）を行っていることが想定される．この処理によって，RTCP レポートの結果が悪くても，アプリケーションで映像品質のエラー回復を行っている．このようにアプリケーションでエラー回復などを行う場合は RTCP レポートが通信の品質をフィードバックする情報として利用しにくい．このことから，DVTS は RTCP をフィードバックし，通信品質の改善の目的に利用するのではなく，ネットワーク状況をユーザおよび DVTS が把握する目的で利用されていると考えられる．

このように，すべてのアプリケーションで，

a) 必ずしも RTCP が RFC3550 の定義に沿って実装されているわけではなく，アプリケーションの実装に依存する
b) RTCP レポートの値がすべての場合で利用可能なものではない．

Lenet において RTCP を利用することの提案は，RTCP が RFC3550 の定義に沿って，実装されたアプリケーションに対して，効を奏するものである．既に開発されているアプリケー

ションの RTCP の実装状況の調査を引き続き進め，上述の 2 点も考慮した上で，RTCP パケットのキャプチャとリプレイの実装を進めるとともに，RTCP 利用の有効性を検証してゆく．

5. むすび

本論文では，我々が開発した，ネットワークエミュレータ Lenet について示し，RTCP レポートの情報をシナリオとして利用するキャプチャ・リプレイの方法を提案した．実装する際のアプリケーションと RTCP の現状から，提案手法の問題点を明確にした．また，エンドユーザにも利用しやすいように，GUI を利用したシステムを構築したので，その動作概要を述べた．

今後の課題としては，RTCP によるキャプチャ機能の実装やその他の機能の開発，またその評価である．

Lenet は次の URL にて公開している．
http://lab.ipc.hiroshima-cu.ac.jp/lenet/.

文　献

[1] 石野 正英, 前田 香織, 河野 英太郎, 岸田 崇志, 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, 信学技報, Vol. 105, No. 219, pp. 13–18, 2005.

[2] Kaori Meda, Masahide Ishino, Eitaro Kohno, A Userland with Replay and Capture, The 2007 International Symposium on Applications and the Internet (SAINT 2007), 2007.(to be appeared)

[3] Prolab, NTT AT, http://ngw.ntt-at.co.jp/cti/radvision/prolab.html, 2006.

[4] H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson, RFC 3550,“ RTP: A Transport Protocol for Real-Time Applications, ” Jul., 2003.

[5] Mark Carson and Darrin Santay, NIST Net - A Linux-based Network Emulation Tool, ACM SIGCOMM Computer Communications Review, 33(3), pp.111-126, 2003.

[6] L. Rizzo, Dummynet: a simple approach to the evaluation of network protocols, Computer Communication Review, 27(1), pp.31–41, 1997.

[7] A.Tirumala, F.Qin, J.Dugan, J.Ferguson, and L. Gibbs. *Iperf: The TCP/UDP bandwidth measurement tool*. http://dast.nlanr.net/Projects/Iperf/, 1999.

[8] 西村 浩二, 前田 香織, 相原 玲二, 遠隔機器制御プロトコル RACP のフレームワークとその応用, 情報処理学会論文誌, Vol. 42, No. 12, pp. 2689–2877, 2001.

[9] A. Ogawa. *DVTS: Digital Video Transport System*. http://www.sfc.wide.ad.jp/DVTS/, 2003.