ソフトウェアネットワークエミュレータの パケット送信間隔高精度化

2009SE141 熊澤 秀俊
指導教員: 後藤 邦夫

1 はじめに

通信ネットワークを媒介とするアプリケーションや通信技術の開発には，動作試験用の通信ネットワークを用意する必要がある．教育や小規模の開発においては１台のＰＣで複数のホストをエミュレートできるためコストの安いソフトウェアネットワークエミュレータを用いる．

Linux 上で動作するソフトウェアネットワークエミュレータに Common Open Research Emulator[1](CORE) や Goto's IP Netrowk Emulator[2](GINE) などが存在する．

GINE はパケット送信の Queue 処理間隔を標準では Real Time Clock(RTC) に依存しており最短間隔がおよそ 122.07[usec] となる．これにより GINE の仮想ネットワーク上でパケットが Queue に到着する間隔がこれより短いと固め送り現象が起き通信の模倣精度が低くなる．

当研究では GINE のパケット送信間隔を短くし，固め送り現象を低減する．そのために既存のタイマの修正と，より高精度なタイマを実装する．

性能評価として，RTC などの既存のタイマと新たに提案するタイマが指定した時間間隔通りに動作しているか計測し，精度を比較する．また GINE の仮想ネットワークで固め送りが RTC に比べ低減し，模倣精度が向上していることを確認する．

この研究により Queue 処理を必要とする GINE の仮想ネットワークにおいてタイマに起因するパケットの遅延が低減し現実により近い通信ネットワークの実験環境が得られる．

2 GINE の概要

GINE のタイマに使うクラスについて述べる．

2.1 GINE と CORE

CORE は GUI を持つソフトウェアネットワークエミュレータである．あらかじめルータやホストなど，オブジェクトのひな形が用意されている．ユーザはキャンバスにオブジェクトを配置することで仮想ネットワークを作る．また各ノードをつなぐリンクに対して帯域制限や遅延，損失率などの通信特性を与えることができる．しかし上りと下りの双方に対して画一的な通信特性となる．

GINE も GUI を備えており，各ノード間のリンクに独自のキューを挟むことで同様の通信特性を与えることができる．このとき上りと下りそれぞれに別の通信特性を与えることができる．

2.2 Queue 処理に使用する GINE のクラス

GINE の Queue 処理間隔を制御する GINE のクラスは主に Timer クラス，FrameQueue クラス，ProcessQueue クラスの 3 つである．これらの相互関係を図 1 に示す．

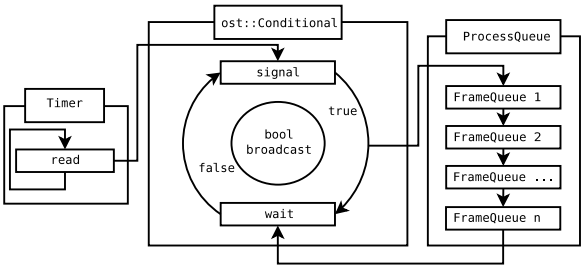

図 1 タイマの動作

Timer クラスは Common C++ Library の ost::Thread クラスを継承しており一つのスレッドとして動作する．ost::Conditional クラスの機能により他のスレッドと処理のタイミングを同期できる．Timer クラスは指定されたタイマを read() するたびに，ost::Conditional を介して待機中の ProcessQueue クラスに処理指示をだす．

FrameQueue クラスはユーザが設定した通信障害を保存し，指定した時刻に達すると後述の ProcessQueue クラスに処理を依頼する．

ProcessQueue クラスは FrameQueue のインスタンスを受け取り，queueList に追加する．また処理依頼を出した queueList の要素を順番に処理し，次の処理指示まで待機する．

3 改良タイマ

既存のタイマである NANOSLEEP の問題の修正と，新しいタイマの提案と実装について述べる．

3.1 NANPSLEEP

GINE には RTC 以外に NANOSLEEP，THREAD，AUDIO のタイマが存在する．NANOSLEEP は高精度にスリープしたが時間を指定できなかった．NANOSLEEP の実体である clock_nanosleep() への時間指定は相対時間であったが，関数は絶対時間として解釈していた．そのため不正な時間指定としてスリープせずに関数から返っていた．この問題を修正し，時間を指定できるようにした．しかし任意の指定時間に対しておよそ 60[usec] 長くスリープするという問題が新たにわかった．そのため新たにタイマを提案し，実装する．

3.2 Traffic Controled Timer

Linux kernel 2.2 以降で追加された QoS 機能と，そのインタフェースである iproute2[3] の一部である tc を用い新しくタイマを提案し，実装する．QoS 機能はパケット中継において帯域制限と優先制御を提供する．

GINE に TCSender クラスを追加する．これはホスト OS のローカルホストに対して短い UDP パケットを最高速で送信し続けるクラスである．Timer クラスでソケッ

トを read() する際，パケットが到着するまでブロックする性質を利用して時間間隔を得る．

また時間間隔の制御にはローカルホストに対して tc を用いて任意の帯域制限 b[bps] を設け，TCSender クラスから Timer クラスへ UDP パケットを送信し続ける．この時パケット長を l[octets] に固定するとパケットが流れる間隔 c[sec] は式 1 で得られる．

$$c = \frac{8b}{l} \tag{1}$$

パケットが流れる間隔 c[sec] が Timer クラスでパケットを受信する間隔となる．この tc による帯域制限を用いたタイマを Traffic Controled Timer(TCTimer) と呼び，さらに仮想ネットワークを起動する直前にタイマの実験を行い指定値と実験値の誤差を考慮して，帯域制限を自動で調節するように実装を変更した TCTimer を Boosted TCTimer と呼ぶ．

4 実験と評価

実験とその結果の評価について述べる．実験に使用した PC は kernel が linux 3.8.0 x86_64 であり，CPU が Intel(R) Core(TM) i5-3337U 1.80GHz である．

4.1 タイマの比較

TCTimer，Boosted TCTimer，NANOSLEEP，RTC に対して 10[usec] から 300[usec] まで 10[usec] きざみで時間指定した際に，1000 回の試行にかかった時間をそれぞれ 30 回計測した．その結果の平均値を図 2 に示す．

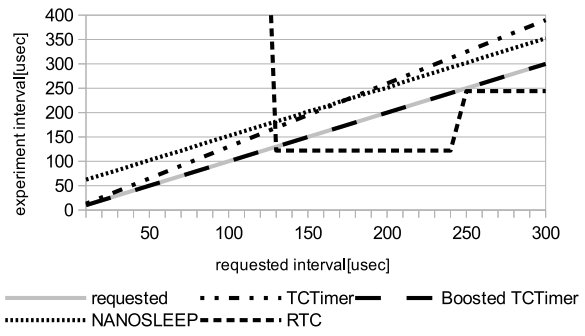

図 2 各種タイマの実験値と指定時刻の比較

RTC は局所的には指定値に近づくものの，全体としては他のタイマと比べ精度が低いと言える．また RTC は 8192Hz より高い周波数を指定されている区間では，指定より低い周波数で動作しており，指定時間に沿った結果が得られなかった．TCTimer はおよそ 120[usec] 未満の指定時間で RTC や NANOSLEEP などの既存のタイマより指定値に近い結果が得られた．さらに Boosted TCTimer は指定時間とグラフが重なっており指定時間に対して高精度な計測結果を示している．

4.2 固め送り抑止

hostA と hostB の間にスイッチを一つ挟んだ仮想ネットワークを作る．hostA から hostB に対して ping を送る．このとき応答を待たずに次のパケットを送信する preload 機能を用いて高速に 15 個のパケットを送信する．hostA が送信した時刻と hostB が受信した時刻を図 3 に示す．(a) に Boosted TCTimer を用いた実験の結果を示し，(b) に RTC を用いた実験の結果を示す．スケールが違うため，RTC の実験結果に Boosted TCTimer の実験結果の最初と最後のパケットを破線で表し，重ねてを示す．

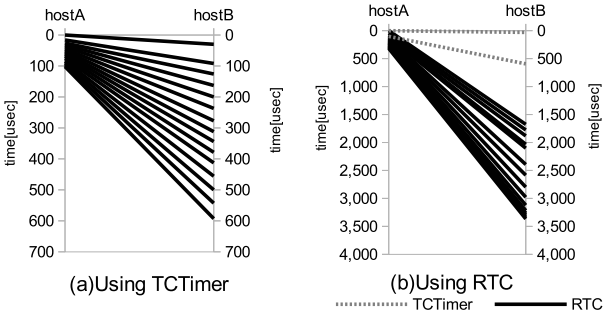

図 3 固め送り抑止の比較

RTC を用いた実験結果と TCTimer を用いた実験結果を比較すると後者は前者と比べパケットの到着のタイミングが均一であり，一つ一つのパケットが送信されてから到着するまでの時間が短い．そのため，仮想ネットワークの経路上にある Queue 処理に起因するパケット伝送の遅延が低減さていると言える．

これにより現実のパケット伝送にはない，GINE の Queue 処理に起因する遅延が低減し，模倣精度が向上したと考えられる．

5 おわりに

当研究により，スリープタイマの高精度化が実現された．それにともない，GINE の独自キューにおけるパケット送信間隔の精度が向上した．これにより GINE の仮想ネットワークは模倣精度が向上し，現実により近いものとなったと言える．

当研究では精度の異なる TCTimer を同時に使用出来ない．今後ループバックインタフェースを Timer インスタンスごとに独立させることで，精度の異なる複数のタイマを同時に使用できるようになると予想される．

参考文献

[1] Ahrenholz, J.: Comparison of CORE Network Emulation Platforms, *Proc. of IEEE MILCOM Conference*, IEEE, pp. 864–869 (2010).

[2] Goto, K.: Network Emulator with Virtual Host and Packet Diversion, *Cyber Journals: Multidisciplinary Journals in Science and Technology, Journal of Selected Areas in Telecommunications (JSAT)*, Vol. 3, No. 3, pp. 13–20 (2012).

[3] The Linux Foundation: iproute2 (accessed Jan. 2014). http://www.linuxfoundation.org/collaborate/workgroups/networking/iproute2.