DPDK を用いたネットワークエミュレータにおけるパケットロスの実装と評価

佐々木 伽音[1,a)]　広渕 崇宏[2]　山口 実靖[1]　高野 了成[2]
1 工学院大学　2 国立研究開発法人 産業技術総合研究所　a) c515046@ns.kogakuin.ac.jp

キーワード：ネットワークエミュレータ，DPDK，10Gb Ethernet，パケットロス，TCP 輻輳制御

1. はじめに

ネットワークエミュレータは，性能評価などにて広く用いられている．高精度なエミュレーション手法として専用のハードウェアを用意する方法があるが，コストが高いという課題がある．ソフトウェアと汎用 PC を用いて低コストで実現する手法として netem[1]があるが，これは高スループット環境で精度が低い問題がある[2]．我々は過去に，ソフトウェアを用いて低コストかつ高い精度で再現できる DEMU を提案した[2]．ただし，これまでの DEMU はパケット損失を再現することができない. 本稿では DEMU 上でパケットロスを再現する手法を提案し，評価によりその有効性を示す.

2. 関連研究

2.1 ネットワークエミュレータ

主に実験ネットワークにおけるパケットロスや遅延を再現する装置であり，ハードウェアによるものとソフトウェアによるものがある．ソフトウェアエミュレータとしては Linux の netem[1]や FreeBSD の dummynet[3]がある.

2.2 netem

netem は Linux に標準搭載されているネットワークエミュレータであり，遅延，パケットロスなどのエミュレーションが可能である．ただし，高スループット環境では遅延やロスのエミュレーションの精度が低下する問題がある[1] . 本稿ではさらに, パケットロスエミュレーションにおけるロスの分布にも問題があることを示す.

2.3 DEMU

我々が開発する DEMU はソフトウェアによる高精度なネットワークエミュレータである[2]. パケット処理フレームワークの DPDK[4]を用いることで，OS のネットワークスタックよりも高速なパケット処理が可能である．我々は先行研究において 10Gbps 環境下において高精度な遅延のエミュレートが可能であることを示した[2]．DEMU には，受信処理，遅延処理，送信処理をそれぞれ担う三つのスレッドがある．各スレッドでは最大 32 パケットをまとめて処理する．伝送遅延をエミュレーションする際には，受信スレッドで予め受信時刻を各パケットに記録し，リングバッファを介して遅延挿入スレッドにパケットを渡す．遅延挿入スレッドでは受信時刻を確認し，指定された挿入遅延時間が経過したパケットはリングバッファを介して送信スレッドへ渡される．送信スレッドはパケットを対応するネットワークインタフェースカードから送信する.

3. DEMU におけるランダムパケットロスエミュレーションの実現

提案手法では，受信スレッドがパケットを受け取るごとに、乱数を用いてパケットを破棄するか否かを判定する．パケットロスを行う場合はパケットの遅延挿入スレッドへの移動を行わないことによりパケットロスを再現する．設定できる最小のパケットロス確率は 10^{-5} とする.

4. 性能評価

4.1 パケットロス評価

図 1-①の環境にてパケットロスの精度に関する調査を行った．本評価ではパケット長 1500 バイトの UDP パケットを 10,000,000 個送信した．指定したパケット破棄率は，0, 0.0001, 0.001, 0.01, 0.1, 1[%]の 6 パターンで，すべての実験において 1 ミリ秒の遅延を挿入した．表 1 に評価結果を示す．まず，netem では DEMU に比べどの場合においてもラウンドトリップ時間が長い. これは netem が DEMU よりもパケット処理速度が遅いことが原因であると考えられる．パケットロスの精度については，DEMU では指定破棄率に対し誤差が非常に小さいが，netem では最大で 3 倍の誤差が生じている.

4.2 TCP 通信評価

本節にて TCP 通信における性能の評価を行う. 通信には iperf3 を用い 60 秒間通信を行った. 実験環境は図 1-②の通りである. 指定したパケット破棄率は, 0, 0.0001, 0.001, 0.01, 0.1, 1[%]の 6 パターンで，DEMU および netem を用いたときのそれぞれのスループットを調査した．実験の結果を図 2 に示す．横軸は指定した破棄率，縦軸は得られたスループットである．図を見ると破棄率 0.001[%]以下では DEMU と netem のスループットはほぼ同じである，破棄率 0.01~0.1[%]では，DEMU のスループットが netem に比べ低い．我々はこれをパケットロス発生のバースト性によるものであると考え，この仮説を検証するために以下の二点について調査を行った.

本研究は、産総研の技術研修制度によって実施された.

第一に，DEMU および netem における連続パケットロス回数の累計分布を調べた. 計測結果を図 3 に示す. 図より，DEMU では一回のパケットロスイベントにつき 1 つのパケットのみ破棄する場合がほとんどであるが，netem では一回のパケットロスイベントで多数のパケットがバースト的に破棄されていることがわかる．また 30 秒間の実験において，netem では総パケットロスイベント数は 94 回であったが総パケットロスは 2349 個，DEMU では総パケットロスイベント数は 1462 回で，総パケットロスも 1462 個であった.

第二に，送信者における輻輳ウィンドウサイズの推移を調べた．まず送信者で使用されている輻輳制御アルゴリズムは，ロスベース手法を用いた CUBIC TCP である．ロスベース手法は，パケットロスを一回検出するとロスした個数に依存せず輻輳ウィンドウサイズを 1（タイムアウト時）またはスロースタート閾値（重複 ACK,SACK 時）まで減少させる．図 4 に指定破棄率 0.01[%]に設定した DEMU および netem を用いた時の送信者の輻輳ウィンドウサイズを示す. 図から DEMU ではパケットロスイベントによる輻輳ウィンドウサイズの低下が多く発生しているのに対し，netem では輻輳ウィンドウサイズの大きな低下が少ないことがわかった.

以上のことから，両者は個々のパケットが一様乱数に従い独立に破棄される設定ではあるが netem では DEMU に比べバースト的な破棄が行われていることが分かった．この原因としては，Linux カーネルの処理が間に合っていない可能性などが挙げられる.

5. おわりに

本稿では，DPDK を用いたネットワークエミュレータ DEMU にパケットロスエミュレーション機能を実現する手法を提案し，評価より有効性を示した．性能評価より提案手法は 10Gbps などの広帯域環境においても既存手法よりも高い精度でパケットロスをエミュレーションできること、および精度の違いに起因する TCP 通信性能への影響を確認した．今後はマルコフ連鎖などを用いたバースト的パケットロスや，パケットリオーダリングなどの機能を追加する予定である.

参考文献

[1] "networking:netem [Linux Foundation Wiki]", https://wiki.linuxfoundation.org/networking/netem (accessed Oct. 30, 2018).
[2] S. Aketa, T. Hirofuchi and R. Takano, "DEMU: A DPDK-based network latency emulator," *2017 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN)*, Osaka, 2017, pp. 1-6. doi: 10.1109/LANMAN.2017.7972145
[3] dummynet: Luigi Rizzo, Dummynet: a simple approach to the evaluation of network protocols, ACM SIGCOMM Computer Communication Review Volume 27 , Issue 1 , PP.31-41 JAN 1997
[4] Intel DPDK: Data Plane Development Kit. URL http://dpdk.org.

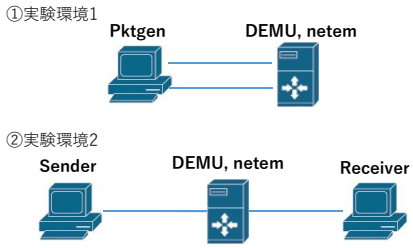

図 1　実験環境図

表 1　遅延挿入・パケットロス精度

	DEMU			NETEM		
指定破棄率	loss[%]	ave_latency[us]	遅延標準偏差[us]	loss[%]	ave_latency[us]	遅延標準偏差[us]
0	0	1757	123	0.03061	2832	191
0.0001	0.00012	1761	126	0.23633	2746	283
0.001	0.00092	1748	126	0.93112	2493	125
0.01	0.01013	1733	123	0.93358	2432	269
0.1	0.10028	1720	119	2.53091	2416	204
1	1.00171	1711	119	3.65741	2386	275

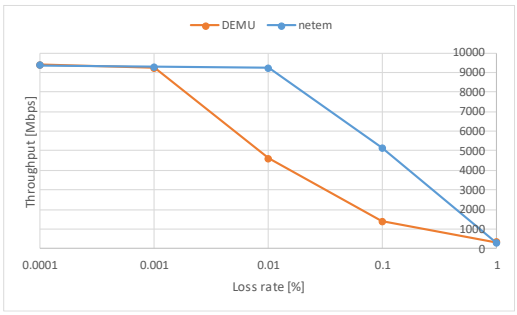

図 2　パケットロスによるスループット変化

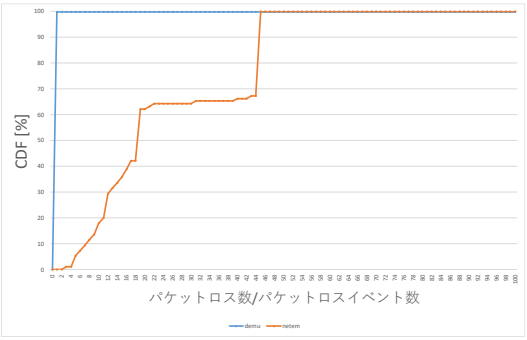

図 3　連続パケットロス数

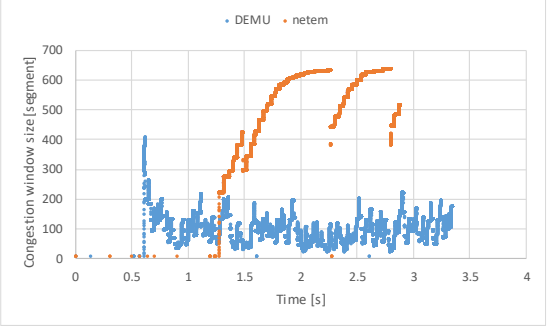

図 4 輻輳ウィンドウサイズ推移