Mintを用いたデバッグ支援環境の評価(修正版)

2015/10/13

藤田将輝

1 はじめに

本資料は前回の資料で測定方法を誤っていたため，これを再考し，再度測定を行った結果と考察を前回の資料に加えたものである．また，本デバッグ支援環境におけるデバッグ支援機構がどの程度の時間で処理をするかを測定し，これについても追記している．さらに，実 NIC で測定を行い，この結果を掲載している．

2 前回との差異

前回の資料との差異を以下に示す．

(1) パケットサイズによるデバッグ支援機構の処理時間の測定
本デバッグ支援環境ではデバッグ支援機構を動作させることで，デバッグ支援 OS からデバッグ対象 OS へ割り込みを発生させる．パケットのサイズを変更させた際の，デバッグ支援機構 1 処理の時間をそれぞれ測定した．

(2) 測定 2 の測定方法の再検討
前回の測定 2 の測定方法はデバッグ支援機構を指定した間隔で連続で動作させ，どの程度の間隔とサイズならば NIC ドライバが正常にパケットを処理できるかを測定していた．しかし，この方法では，デバッグ支援機構の処理時間を下回る間隔では測定できない．このため，測定方法を再度検討し，測定した．

(3) 実 NIC での測定
本評価では，実 NIC との比較がなかった．本デバッグ支援環境の有用性を示すため，実 NIC がどの程度の処理速度を実現しているかを測定し，比較した．

3 評価目的

本評価の目的は，Mint を用いたデバッグ支援環境を使用して，デバッグ対象 OS にパケット受信割り込み処理をさせた際に，どの程度の割り込み間隔とパケットのサイズならば正常にパケットを処理できるかを測定し，実現できる処理速度を算出することにより，実際の NIC を再現することに十分な性能を実現できているかを評価することである．

4 評価環境

本評価における評価環境を表 1 に示す.

表 1 測定環境

項目名	環境
OS	Fedora14 x86_64(Mint 3.0.8)
CPU	Intel(R) Core(TM) Core i7-870 @ 2.93GHz
メモリ	2GB
Chipset	Intel(R) 5 Series/3400
NIC	RTL8111/8168B
NIC ドライバ	RTL8169
ソースコード	r8169.c

5 測定対象

本評価で行う測定は，以下の 5 つである.

【測定 0】 パケットのサイズを変えた際のデバッグ支援機構の処理時間の変化

【測定 1】 コアが反応可能な IPI の受信間隔の測定

【測定 2】 NIC ドライバがパケットを処理可能な IPI 送信間隔とパケットサイズの測定

【測定 3】 デバッグ対象 OS 上で動作する UDP の受信プログラムがパケットを処理可能な IPI 送信間隔とパケットサイズの測定

【測定 4】 実 NIC でパケットを受信した際の処理速度の測定

6 測定 0

6.1 測定方法

パケットのサイズを変化させ，デバッグ支援機構を動作させた際の，処理時間の変化を測定した．デバッグ支援機構の以下の処理にどの程度の時間がかかるかを測定した.

(1) パケットを NIC の受信バッファに配置する.
(2) NIC の受信ディスクリプタを受信状態に更新する.
(3) IPI を送信する.

測定したパケットのサイズを以下に示す.

(1) 1KB
(2) 1.5KB
(3) 4KB
(4) 8KB
(5) 12KB
(6) 16KB

この中で 1.5KB はネットワークインタフェースにおけるデフォルトの MTU のサイズである.

6.2 測定結果

測定結果を表 2 に示し，分かったことを以下に示す.

(1) パケットのサイズが増加するに連れて，処理時間は増大している.

表 2　パケットのサイズを変化させた際のデバッグ支援機構の処理時間の変化

サイズ	処理時間
1KB	356ns
1.5KB	380ns
4KB	697ns
8KB	2200ns
12KB	3983ns
16KB	4797ns

7 測定 1

7.1 測定方法

IPI の間隔を調整し，連続で IPI を送信し，割り込みハンドラの動作確率を測定する．具体的には，デバッグ支援 OS からデバッグ対象 OS に 5000 回連続で IPI を送信し，デバッグ対象 OS の割り込みハンドラを動作させた際に，カウンタをインクリメントすることで，動作した割り込みハンドラの回数を測定し，動作確率を算出する．これらの動作を 1 サイクルとし，IPI の送信間隔を一定時間増加させながら何サイクルも行う．これにより，全ての IPI にコアが反応できる送信間隔を調査する.

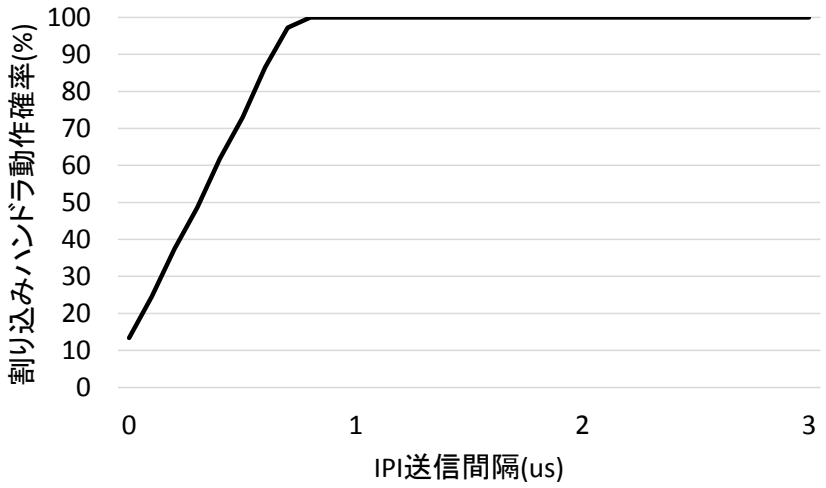

図 1　IPI の反応限界

7.2　測定結果

測定結果を図 1 に示し，分かったことを以下で説明する．

(1) IPI の送信間隔が 0us から 1us の間，割り込みハンドラの動作確率は一次関数的に増加する．
(2) IPI の送信間隔を 1us 以上にすると動作確率は 100% となる．

8　測定 2

8.1　測定方法

本デバッグ支援環境を用いて，連続でパケットを送信した際の NIC ドライバでパケットを処理できる確率を測定する．測定に用いたパケットは UDP パケットの Ethernet フレームであり，パケットのサイズは Ethernet フレームのサイズである．また，パケットの受信成功とは，NIC ドライバがパケットをソケットバッファに格納することである．測定を行う際の具体的な処理フローを図 2 に示し，以下で説明する．

デバッグ支援 OS

(1)　パケットジェネレータを動作させる．
(2)　NIC ドライバの受信バッファの全てのエントリ (256 エントリ) にパケットを配置する．
(3)　受信ディスクリプタを更新する．
(4)　IPI を送信する．
(5)　指定した時間だけ処理を待たせる．
(6)　5000 回繰り返したかを判定する．5000 回に満たなければ (3) へ．5000 回であれば (7) へ．

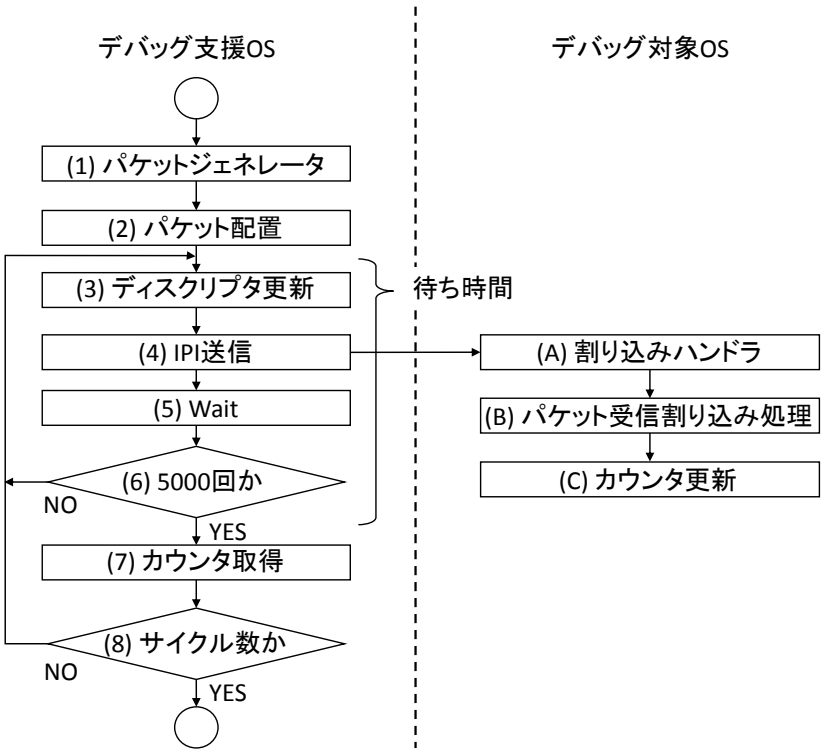

図 2　測定フロー

(7)　共有メモリのカウンタを取得し，出力する．

(8)　(3)〜(7) を 1 サイクルとして，待たせる時間を増加させながら指定したサイクル数になるまで繰り返す．

デバッグ対象 OS

(A)　IPI を受信すると割り込みハンドラが動作する．

(B)　割り込みハンドラがパケット受信割り込み処理を呼び出し，処理が開始する．

(C)　パケットを取得し，ソケットバッファに格納すると，カウンタをインクリメントする．

(3)〜(6) のループの中で，wait 処理を除いた (3)，(4) の処理時間は約 100ns であった．このため，IPI の送信間隔は正確であると言える．これらの操作を以下の 6 つのサイズのパケットで行うことで，各パケットのサイズにおける受信可能な動作間隔がわかる．

(1) 1KB
(2) 1.5KB
(3) 4KB
(4) 8KB
(5) 12KB
(6) 16KB

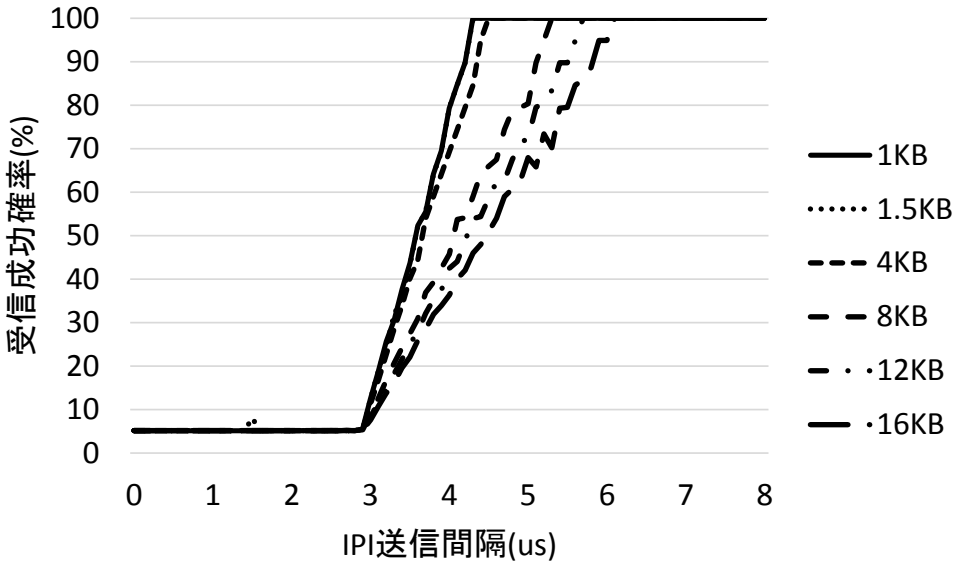

図 3　NIC ドライバにおける受信成功確率

さらに，1 サイクルにかかった時間を測定する．この時間と，処理したデータ量からどの程度の処理速度を実現できているかが分かる．

8.2　測定結果

測定結果を図 3 に示し，読み取れることを以下に示す．

(1) 0us から 3us の間，どのサイズでもほとんどの割り込みでパケット受信に失敗している．
(2) 3us を超えると，一次関数的に受信成功確率は増加し始める．この際，パケットサイズが小さいほど傾きが急になっている．
(3) 受信成功確率が 100% になってからはどれだけ動作間隔を増加させても受信成功確率は 100% となる．

9　測定 3

9.1　測定方法

本デバッグ支援環境を用いて，デバッグ支援 OS からデバッグ対象 OS へパケットを送信した際の，デバッグ対象 OS 上で動作する UDP の受信用プログラムでどの程度の間隔とパケットサイズならば，パケットを受信できるかを測定する．また，デバッグ支援 OS からデバッグ対象 OS 上のプログラムまでで実現できている通信量を測定する．具体的な方法は，8.1 節とほぼ同様である．カウンタをインクリメントする箇所が 8.1 節では NIC ドライバがパケットを受信した時であったのに対し，本測定では UDP の受信プログラムがパケットを受信した時であることが差分である．また，ここでの受信成功とはデバッグ対象 OS 上で動作する UDP の受信プログラムがパケットを受信することである．

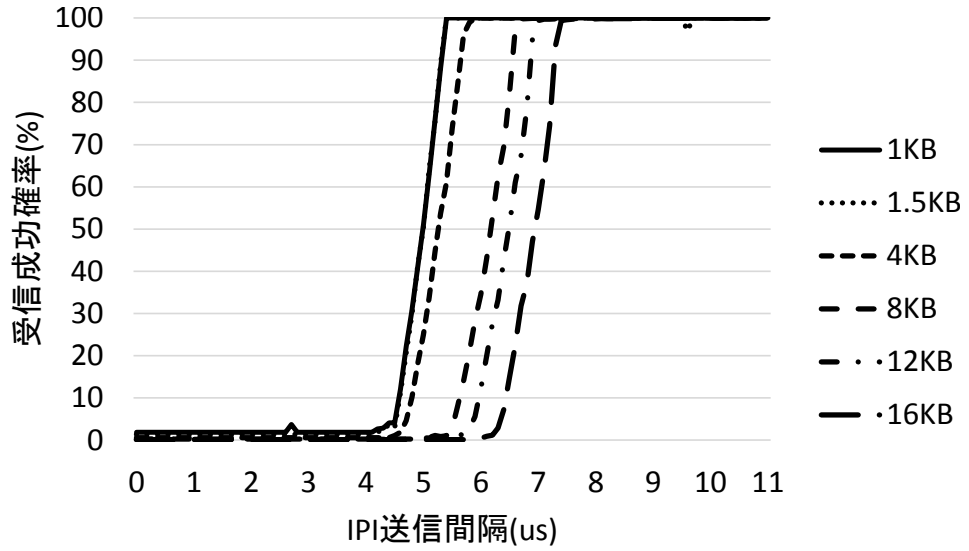

図 4　UDP 受信プログラムにおける受信成功確率

9.2　測定結果

測定結果を図 4 に示し，読み取れることを以下に示す．

(1) どのパケットサイズでもある時点まではほとんど受信に失敗している．
(2) ある時点から一次関数的に受信成功確率が増加している．
(3) 受信成功確率が増加し始めるある時点は，パケットサイズが増加するに連れて増大している．
(4) 受信成功確率が一次関数的に増加する際，パケットのサイズに関わらず，傾きは一定である．
(5) 受信成功確率が 0 回になる IPI 送信間隔以上に間隔を増加させても受信成功確率は 100% となる．

10　測定 4

10.1　測定方法

2 台の計算機を用いて，一方の計算機から他方の計算機へパケットを送信した際の，処理速度を測定した．2 台の計算機は LAN ケーブルで接続されている．具体的には以下の流れで測定を行った．

(1) 計算機 0 から計算機 1 へ 1.5KB の UDP の Ethernet フレームを連続で 5000 回送信する．1.5KB とは計算機 1 の NIC の MTU である．
(2) 計算機 1 がパケットを受信する．この際，計算機 1 の NIC ドライバには 1.5KB のパケットを受信するとカウンタを回すように改変を加えている．
(3) 計算機 0 で 5000 回パケットを送信する際に時間を測っており，5000 回の送信を終えるとかかった時間を出力する．

(4) 5000 回の送信を終えると，計算機 1 でカウンタの値を取得する．
(5) (3) と (4) の値，およびパケットのサイズである 1.5KB から処理速度を算出する．

また，同様の流れで，パケットが計算機 1 上で動作する UDP の受信プログラムまで届きいた際の処理時間も測定した．

10.2 測定結果

測定結果を表 3 に示し，分かったことを以下に示す．

(1) NIC ドライバでの測定結果とプログラムでの測定結果は同じである．

表 3　NIC ドライバにおけるパケットのサイズと割り込み間隔の関係

測定箇所	サイズ	処理時間	処理速度
NIC ドライバ	1.5KB	60378us	0.92Gbps
プログラム	1.5KB	60378us	0.92Gbps

11 考察

11.1 測定 0

測定結果である 6.2 節から，パケットのサイズが増加するに連れ，デバッグ支援機構の処理時間も増加していると言える．また，デバッグ支援機構を動作させて連続でパケットを処理させる際，結果に示した表 2 以上の間隔を取る必要があると言える．

11.2 測定 1

測定結果である 7.2 節から，現在の測定環境では IPI の送信間隔が約 1us 以上であれば連続で IPI を送信した際に，全ての IPI に割り込みハンドラが動作できることが分かる．したがって，本実験環境下では，連続で割り込みを発生させる際，最低でも 1us 以上の間隔は必要であると言える．

11.3 測定 2

測定結果である 8.2 節から，連続で割り込みを発生させた際，パケットのサイズを増加させるほど全てのパケットを処理することにかかる時間が増加することが分かる．なぜグラフのような概形になるかは考察できていない．また，NIC ドライバで測定した受信成功回数，パケットのサイズ，および 1 サイクルにかかった時間から各パケットにおける通信量を算出した．各パケットサイズにおける初めて受信成功確率が 100% になった際の IPI 送信間隔，1 サイクルにかかった時間，総データ量，通信量, および

実 NIC の測定結果を表 4 に示す.

表 4　NIC ドライバにおけるパケットのサイズと割り込み間隔の関係

サイズ	割り込み間隔	1 サイクルの時間	総データ量	処理速度
1KB	4.3us	21689us	5000KB	1.7Gbps
1.5KB(NIC)	-	60378us	7324KB	0.92Gbps
1.5KB	4.4us	22203us	7324KB	2.5Gbps
4KB	4.6us	23195us	20000KB	6.5Gbps
8KB	5.5us	27788us	40000KB	10.9Gbps
12KB	5.9us	29770us	60000KB	15.3Gbps
16KB	6.4us	32491us	80000KB	18.7Gbps

11.4　測定 3

測定結果である 9.2 節から，連続で割り込みを発生させた際，NIC ドライバでの結果と比べて，より多くの時間がかかっていることが分かる．このことから，NIC ドライバですべてのパケットを処理できていても，プログラムまでパケットが届くにはより多くの時間がかかることが分かる．なぜグラフのような概形になるかは考察できていない．また，UDP の受信プログラムで測定した受信成功回数，パケットのサイズ，および 1 サイクルにかかった時間から各パケットサイズにおける通信量を算出した．各パケットサイズにおける初めて受信成功確率が 100% になった際の IPI 送信間隔，1 サイクルにかかった時間，総データ量，通信量，および実 NIC の測定結果を表 5 に示す.

表 5　UDP 受信プログラムにおけるパケットのサイズと割り込み間隔の関係

サイズ	割り込み間隔	1 サイクルの時間	総データ量	処理速度
1KB	5.5us	27677us	5000KB	1.3Gbps
1.5KB(NIC)	-	60378us	7324KB	0.92Gbps
1.5KB	5.5us	27700us	7324KB	2.0Gbps
4KB	6.1us	30705us	20000KB	4.9Gbps
8KB	7.7us	38649us	40000KB	7.8Gbps
12KB	9.4us	47241us	60000KB	9.6Gbps
16KB	11.1us	55789us	80000KB	10.9Gbps

11.5 測定 4

測定結果を表 4，5 に挿入し，比較すると，本デバッグ支援環境を用いると，実 NIC よりも非常に高速に処理できていることがわかる．また，MTU を超えた大きなパケットも問題なく処理できている．

12 評価

上記の結果と考察から，NIC ドライバで最大約 18.7Gbps，デバッグ対象 OS 上で動作する UDP 受信プログラムで最大約 10.9Gbps の処理速度を実現できることが分かった．また，実機の測定から本実験環境では，処理速度が約 1Gbps であることがわかる．また，PCI Express 1.1 の 1 レーンの転送量が 2.5Gbps であることから，実際の NIC が実現できる最大の処理速度は 2.5Gbps であると言える．本デバッグ支援環境で実現できる処理速度と実際の NIC が実現できる処理速度を比較すると，本デバッグ支援環境は実際の NIC の処理速度を大きく超えた処理速度を実現できていることがわかる．このことから，本デバッグ支援環境を用いることで現在は実現できていない高処理速度の NIC のエミュレートができると考えられる．高処理速度の NIC やバスが開発されたと想定し，これらに対応するドライバを開発できる．

13 おわりに

本資料では，再考した測定方法とその結果，デバッグ支援機構の処理時間，および実 NIC での測定結果について記述した．測定結果から本デバッグ支援環境は実 NIC に比べ，非常に高速な処理速度を実現できてると言える．