2008－OS－108 (24)
2008／4／24

TwinOSにおける送信パケット監視機能と基本性能評価

粟田 祐一 † 乃村 能成 †† 谷口 秀夫 ††
† 岡山大学工学部 †† 岡山大学大学院自然科学研究科

近年，OSに対する不正な攻撃が増加している．このような場合，攻撃対象となるOSを外部から監視することが有効である．我々は，1台の計算機上に2つのLinuxを独立に走行させるTwinOSを研究開発し，一方のOSが持つNICを他方のOSに提供する擬似NIC機能を実現している．本稿では，この擬似NIC機能を利用して一方のOSが行う送信を他方のOSから監視する機能の設計と実装について述べる．監視対象となるOSの外部から監視を行うため，攻撃者による監視記録の改ざんを防ぐことが期待できる．また，基本評価として，通信監視機能の実装による通信性能への影響について報告する．

Evaluation of Transmission Packet Monitor on TwinOS

Yuichi AWATA †, Yoshinari NOMURA †† and Hideo TANIGUCHI ††
† Faculty of Engineering, Okayama University
†† Graduate School of Natural Science and Technology, Okayama University

Recently, illegal attacks to OSs are increasing. A lot of intruder detection method have been proposed. In this paper, we propose a method for monitoring communication using TwinOS. TwinOS runs two Linux on a single PC, and has a virtual NIC that provides one Linux with the other Linux's NIC. Our proposed method monitors one Linux's transmission from the other Linux via virtual NIC, and keep intruders away from log files. In addition, as basic evaluation, evaluated influence on communication performance by the implementation of the monitor.

1. はじめに

近年，インターネットが普及したことにより，オペレーティングシステム (以降，OSと略す) に対する不正な攻撃が増加している．例えば，ウイルスに感染した計算機からファイル共有ソフトを通じて個人情報が流出するといった被害が発生し，大きな問題となっている[1]．加えて，DoS攻撃やDDoS攻撃[2)3)]も増加している．不正な攻撃への対策として，ウイルス対策ソフトやパーソナルファイアウォールソフトが用いられてきた．計算機への不正侵入監視手法も多く提案されている[4]．しかし，不正な攻撃を行うソフトウェアの巧妙化，多様化により，従来の対策では検知できない攻撃が増加している．不正な侵入を許し，OSが乗っ取られた場合，情報が改ざんされ，侵入者の追跡が困難になる．この問題への対処として，攻撃対象となるOSの外部から監視する方式が有効である．例えば，計算機の通信を監視する専用の計算機 (以降，監視計算機と略す) を用意することが考えられる．この場合，複数の計算機の通信をまとめて監視する監視計算機は，高機能であることが要求され，高価なものになってしまう．これに対し，被監視計算機と監視計算機を1対として構成すると監視計算機の性能を抑制できるものの，その数は多くなり，コスト高を招いてしまう．

一方，計算機ハードウェアの性能向上に伴い，1台の計算機上に複数のOSを走行させ，その上で多様なサービスを実現したいという要求がある．この要求を満たす方式として，仮想計算機方式 (以降，VMと略す) がある．VM方式は，1つのOSがホストOSとなり，他のOS(以降，ゲストOSとする) をカーネルコールレベル，あるいはハードウェアインタフェースレベルでエミュレートすることによって実現されている．このため，ゲストOSはホストOSの機能に制約を受けるとともに，ホストOSの停止がシステム全体の停止につながるという問題がある．また，この方式ではホストOSとゲストOSが入出力機器を共有することになり，単独のOSが走行する環境に比べ複数のOSが走行する環境の方がI/O性能が低下し

やすいという問題がある．これらを回避するために，VMware ではバルーニング等の技術を用いてゲスト OS の性能を向上させる方式も実現している[5]．しかしこの場合も，入出力機器は共有しているため，占有した場合と同等の性能を得ることは難しい．そこで，我々は，1 台の計算機上に 2 つの Linux を独立に走行させる TwinOS を提案している[6]．TwinOS は，2 つの Linux が 1 台の計算機上で動作する．各 OS はハードウェアを OS ごとに分割し，占有させることによって，互いの処理の影響を受けない．このため，VMware に比べ両 OS とも入出力性能を十分に利用できるという特徴を持つ．また，TwinOS は，一方の OS が占有する NIC(Network Interface Card) の機能を他方の OS に仮想化して提供する機能 (以降，擬似 NIC 機能と略す) を実現している[7]．

本稿では，TwinOS 上に実現した擬似 NIC 機能を利用した通信監視機能として，一方の OS が行う送信を他方の OS から監視する機能について述べ，基本評価を報告する．

2. TwinOS[6]

2.1 TwinOS の特徴

TwinOS は、以下の特徴を有する．

(1) 相互に他 OS の処理負荷の影響を受けない

(2) 両 OS とも入出力性能を十分に利用できる

このために，1 台の計算機ハードウェアにおいて，プロセッサやメモリ，入出力機器といった各資源について効果的な共有と占有を行っている．まず，共有するハードウェア資源を最小限とすることが有効であるため，プロセッサのみを共有する．一方，メモリや入出力機器は 1 つの OS が占有する．具体的には，プロセッサの時分割制御方法については，起動時には起動処理を順次行うこととし，割込み/例外処理においてはタイマ割込みを利用して OS を切替え，時分割することとしている．実メモリの空間分割方法については，起動時に若番アドレス空間と老番アドレス空間に 2 分割し，共有部分を有しないこととしている．入出力機器の占有方法については，各 OS 毎に指定された入出力機器のみを占有制御するように起動時に環境設定する．このため，割込み/例外処理では，当該割込みを発生した入出力機器を占有制御している OS の割込み処理が呼出されるように制御する．したがって，割込み発生時に走行している OS (AP が走行している場合は，その AP が利用している OS) が当該割込みを発生した入出力機器を占有制御している OS と異なっている際には，OS の切替えを行う．また，各方式の実現においては，各 OS の改修を最小化するとともに，改修量が同等な場合は 1 つの OS に改修を集中させ局所化することとしている．これにより，開発工数の削減だけではなく，将来における Linux 以外の OS への本手法適用の際の工数削減を可能にしている．

2.2 擬似 NIC 機能

擬似 NIC 機能は，一方の OS が占有する機器を他方の OS に提供する機能によって実現されている．これによって TwinOS の利用法を拡大させることができる．OS はデバイス NIC ドライバを経由し I/O 命令を発行することによってハードウェアの制御を行う．このため，2 つの OS で 1 つのハードウェアを使用する場合には I/O 命令の監視が必要となる．

図 1 に擬似 NIC の構成と I/O 命令および割込み処理の流れを示し，以降に説明する．

TwinOS 方式において，擬似 NIC は NIC を占有する OS が持つ．OS1 の NIC ドライバが NIC に対して I/O 命令を発行しようとすると，OS1 内における実装により，OS 切替えが発生する．そして，擬似 NIC が I/O 命令の内容を引き継ぎ，I/O 命令の種類に応じた処理を行う．また，処理終了等にともなう NIC からの割込みを契機に図 1 の (A)〜(D) の処理が行われる．割込み処理の内容を以下に説明する．

(A) 割込みを受ける処理，OS 切り替えの前処理，および OS を切り替える処理．

(B) NIC ドライバで本来行われる処理と，I/O 命令発行時に OS を切り替える処理．

(C) タイマ割込みが発生するまでの共存 OS での処理と OS を切り替える処理．

(D) 共存 OS からの復帰処理

3. 送信処理

TwinOS は，NIC:3com 3c905-TX，NIC ドライバ:3c59x.c を対象とし，擬似 NIC 機能を実現している．本章では，この NIC と NIC ドライバの送信処理について述べる．

NIC ドライバは，送信パケットを管理する表 (以降，送信管理表と略す) によってパケット単位で管理し，複数の送信管理表をリスト構造で持つ．NIC ドライバは，このリストの先頭アドレスを I/O 命令によって NIC に通知することで，送信を依頼する．これらの関係を図 2 に示し，NIC の処理を以下に説明する．

(1-1) 送信管理表を参照し，パケットの情報 (物理アドレス，サイズ) を取得する．

(1-2) パケットを NIC 内の送信バッファへ DMA 転送で読込む．

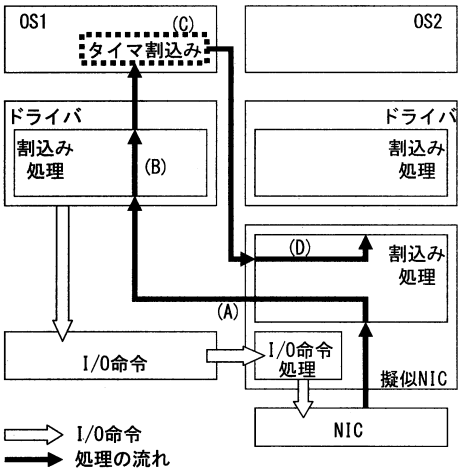

図 1 擬似 NIC の構成と割込み処理の流れ

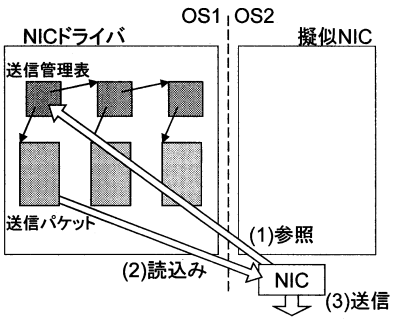

図 2 送信処理の流れ

(1-3) パケットの読込みが完了したことを NIC ドライバに通知する割込み (以降，複写完了割込みと略す) を発生させるか判断する．これは，送信管理表にセットされたフラグから判断する．フラグがセットされていれば，複写完了割込みを発生させる．

(1-4) リストをたどって次の送信管理表の処理に移る．

NIC は上記 (1-1)〜(1-4) をリスト終端まで繰り返し，リスト中のすべてのパケットを読込み，送信する．

次に，NIC ドライバによるパケットの通知処理の流れを図 3 に示し，以下に説明する．

(2-1) 新しいパケットの情報 (物理アドレス，サイズ) を格納した送信管理表を作成する．その後，NIC の複写完了割込みを発生できるように，送信管理表に割込みフラグをセットする．

(2-2) 割込みを禁止し，ロックをかけ，NIC の送信パケット読込み処理を停止させる．(OUT 命令 A)

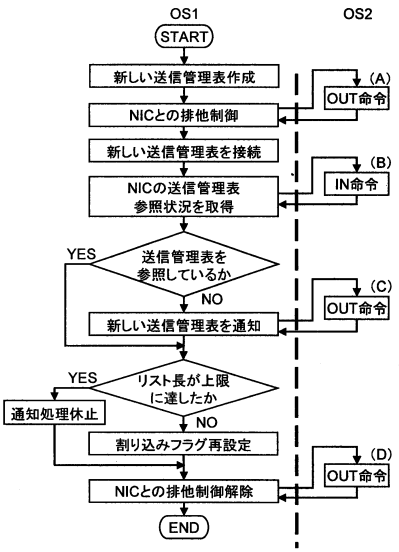

図 3 パケット通知処理

(2-3) 以前のパケット通知処理で NIC に通知した送信管理表に新しい送信管理表を繋ぐ．

(2-4) NIC の送信管理表の参照状況を取得する．(IN 命令 B)

(2-5) NIC が送信管理表を参照しているか確認する．これは，NIC による送信の流れにおける (1-1)〜(1-4) の状態にあるかどうかを確認することである．参照していなければ，新しい送信管理表の先頭アドレスを通知する．(OUT 命令 C)

(2-6) リストで繋いだ送信管理表の数が上限に達しているか確認する．NIC ドライバが持つ送信管理表には限りがあり，これを超えて使用することはできない．このため，送信管理表を上限まで使用した場合は，以降のパケット通知処理を一時的に休止する．処理が再開されるのは，読込み済みとなったパケットが解放されて，パケットを管理していた送信管理表の再利用が可能になったときである．上限に達していない場合は，送信管理表の割込みフラグを再設定する．新しい送信管理表の接続によりリストの終端が変わるため，これまで終端だった送信管理表の割込みフラグをクリアする．

(2-7) 割込みの禁止とロックを解除し，NIC の送信パケット読込み処理を再開させる．(OUT 命令 D)

最後に，複写完了割込み時に行うパケット解放処理について，様子を図 4 に示し，以下に説明する．

(3-1) 未解放のパケットがあるか確認する．なければ，(3-5) を行う．

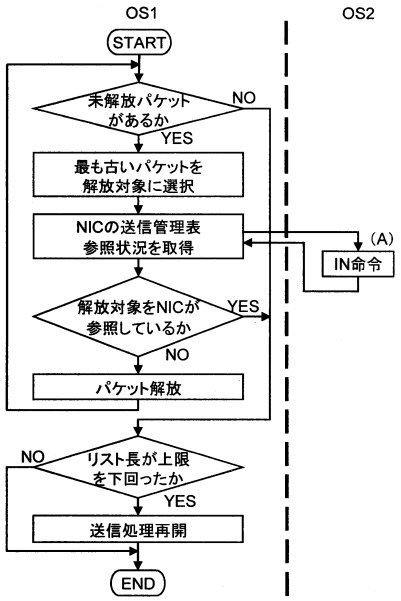

図 4 パケット解放処理

(3-2) 未解放のパケットの中から，最も古いパケットを解放対象とする．

(3-3) NIC の送信管理表の参照状況を取得する．(IN 命令 A)

(3-4) NIC が参照している送信管理表は，解放対象に選択されたパケットの送信管理表と同じであるかどうかを確認する．同じであれば，NIC はこのパケットとこれ以降に通知処理されたパケットの読込みを完了していないため，(3-5) を行う．同じでなければパケットを解放し，(3-1) に戻る．

(3-5) 解放処理により送信管理表のリスト長が上限を下回ったかを確認する．下回っていたら，パケット通知処理を再開させる．

4. 送信パケット監視機能

4.1 通信監視の基本構成

TwinOS では，先行して起動する OS(以降，先行 OS と略す) と，先行 OS によって起動される OS(以降，共存 OS と略す) がある．TwinOS の実現において必要な改修は，先行 OS に集中させており，擬似 NIC 機能は先行 OS 上に実現している．以降では，TwinOS 上で擬似 NIC 機能を利用して共存 OS が通信を行っている状況で，先行 OS から監視 (ログの取得，パケットの破棄) を行う機能について述べる．

通信監視機能の構成を図 5 に示す．制御部は先行 OS 上の AP として構成し，検査部と操作部は先行 OS 内に構成する．以降に，各処理部について説明する．

(1) 監視機能の制御 (制御部)

(2) 対象パケットの選定 (検査部)

(3) 対象パケットに対する操作 (操作部)

制御部は，パケットの中から監視対象パケットを選定するための規則や監視対象パケットに対して行う操作の規則を決定する．そして，決定した規則を検査部および操作部に通知する．操作の規則でパケットのログを取得する場合には，操作部から渡されたログを読み出し，ログを保存する形式へと変換した後ディスクへ保存する処理を行う．また，制御部では，保存したログを基に解析を行い，ユーザへの通知や判定規則への反映を行う．

検査部は，制御部から通知された規則を基に，共存 OS 内のパケットが監視対象パケットであるか否かを判定する．監視対象パケットであると判定したパケットがある場合，操作部に通知する．

操作部は，検査部が監視対象パケットであると判定したパケットに対して，ログの取得およびパケットの破棄処理を行う．ログを取得する場合は共存 OS 内のパケットの一部を制御部内の領域 (以降，ログ域と略す) に取得データを転送する．パケットを破棄する場合は，受信バッファおよび受信バッファ管理構造体を操作して破棄対象のパケットを共存 OS に渡さないようにする．

制御部，検査部，操作部のログ取得については，対象とするパケットが受信パケットか送信パケットに関係なく同様である．しかし，操作部のパケットの破棄処理については，送信と受信では処理の流れと扱うデータ構造が異なる．

4.2 設　　計

4.2.1 方針と課題

送信パケット監視機能の設計方針を以下に示す．

(方針1)共存 OS を改変しない

(方針2)共存 OS の通信性能への影響を抑える

(方針 1) は，TwinOS 実現の方針である「各 OS の改修量を最小とするとともに，改修量が同等な場合は 1 つの OS に改修を集中させ局所化する」に基づく．また，攻撃者に対する監視処理の隠蔽のため，共存 OS への改変は抑える必要がある．(方針 2) は，先行 OS による監視処理を行った際も共存 OS の通信性能を維持するためである．上記の方針を踏まえて設計するにあたり，以下の課題がある．

(課題1)監視契機

監視処理を先行 OS 上で行うために，共存 OS の

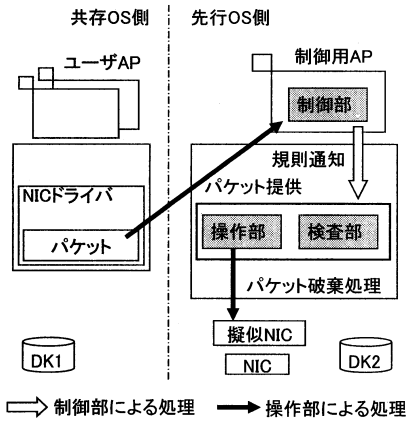

図 5　通信監視機能の構成

NIC ドライバが擬似 NIC を経由して行う送信処理の中で，先行 OS で監視を実行する契機が必要である．

(課題2)監視方式

監視契機を利用して，共存 OS を改変せずにパケットを監視できなければならない．また，共存 OS の通信性能を維持するために，監視対象と判定されたパケットを効率よく処理できなければならない．

4.2.2 対　　処

最初に，監視契機 (課題 1) について述べる．2.2 節で述べた擬似 NIC 機能より，先行 OS がパケットを監視する契機として，I/O 命令発行時 (契機 1) と NIC からの割込み発生時 (契機 2) がある．(契機 2) は，パケットの送信処理で発生する割込みとして，送信管理表からパケットを読み込んだ後に NIC が発生させる複写完了割込みである．この場合，NIC 内バッファに送信パケットを複写してしまった後に発生するため，パケットの保存を行うことはできるものの，パケットの送出を止めることはできない．

このため，(契機 1) に該当する図 3 の I/O 命令 (A)〜(D) を監視契機とする．

次に，監視方式 (課題 2) として，監視契機を利用して実現可能な 3 つの方式を図 6〜8 に示し，以下に説明する．

パケット情報書換方式 (方式 1)：　この方式は，送信管理表が持つパケットの管理情報 (物理アドレス，サイズ) を書き換える．具体的には，破棄対象パケットの送信管理表が持つパケット管理情報を書き換え，パケット内のデータが送出されないようにする．図 6 では，送信パケット P1 を破棄対象としている．

リスト構造書換方式 (方式 2)：　この方式は，NIC ドライバの持つ送信管理表のリスト構造を書き換える．

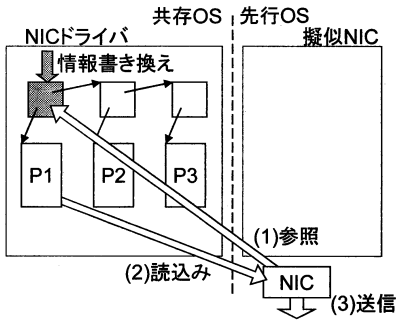

図 6　パケット情報操作方式

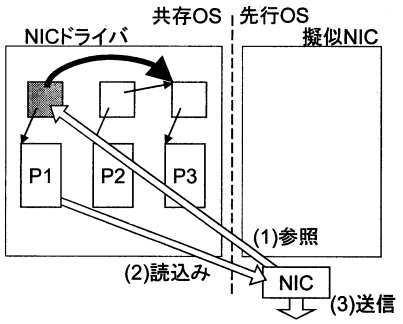

図 7　リスト構造操作方式

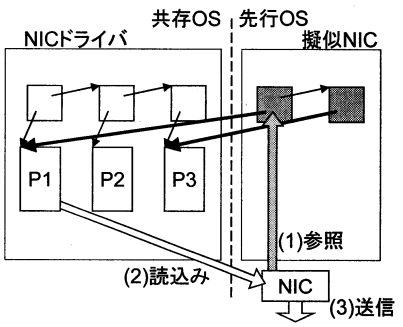

図 8　送信管理表複写方式

具体的には，通常パケットの送信管理表のみでリストを再構成して NIC に通知することで，破棄対象パケットの読込みを抑制する．図 7 では，送信パケット P2 を破棄対象としている．

送信管理表複写方式 (方式 3)：　この方式は，共存 OS の送信管理表とは別の送信管理表を先行 OS 上に作成する．具体的には，共存 OS の送信管理表から，通常パケットを持つ送信管理表を選択して先行 OS 上に複写する．そして，先行 OS の送信管理表でリストを構成して NIC に通知することで，破棄対象パケッ

表 1 監視方式の比較

	共存 OS への改変	通信への影響	処理の単純さ
パケット情報書換方式	パケット情報の書き換え	大 (破棄対象パケットに対する読込み)	単純
リスト構造書換方式	リスト構造の書き換え	小 (リスト書き換えによる処理時間増加)	やや複雑
送信管理表複写方式	なし	中 (送信管理表複写による処理時間増加)	複雑

トの読込みを抑制する．図 8 では，送信パケット P2 を破棄対象としている．

各方式を以下の観点から比較し，課題を表 1 に示す．

(観点 1)共存 OS への改変

(観点 2)共存 OS の通信性能への影響

(観点 3)処理の単純さ

3 つの方式の中で，(方式 1) は最も短い処理時間でパケットの監視処理を行うことができる．これは，破棄対象の送信管理表が持つパケットの管理情報を書き換えるという単純な処理だけで破棄処理を実現できるからである．しかし，他の方式と違い，NIC による破棄対象パケットに対する読込みを抑制しない．このため，破棄対象パケットが大量に発生した場合に通信性能が低下する．

(方式 2) は，(方式 1) と違い無駄なパケットの送信が行われることがなく，(方式 3) と違い新しく管理が必要なデータ構造を持たない．したがって，3 つの方式の中で最も高い通信性能を得ることができる．

(方式 3) は，先行 OS 上に作成した送信管理表の管理を必要とし，監視処理が複雑である．また，通常パケットを送信処理する度に送信管理表の複写を行うので，共存 OS の通信性能に悪影響を与える．しかし，他の方式と違い，共存 OS が持つ送信管理表と送信パケットを書き換える必要がない．この点は，(方針 1) として重要な項目であるため，監視処理は (方式 3) を採用する．

5. 評　　価

5.1 評 価 方 法

以上の送信パケット監視機能を TwinOS に実装し，監視機能の実装による共存 OS の通信性能への影響について，以下の評価を行った．

(評価 1)通信処理時間

(評価 2)他の演算処理との相互影響

送信監視機能を実装した TwinOS(以降，監視実装と略す) を，以下の条件で動作させ，通信処理時間を測定する．

(1)　先行 OS 上で監視 AP を動作させない

(2)　パケットのログ取得処理を行わない

(3)　パケットの破棄判定処理では，すべてのパケットの送信を許可する

比較対象として，改造を行っていない Linux(以降，オリジナルと略す) と擬似 NIC 機能のみ実装した TwinOS(以降，擬似 NIC 実装と略す) を用いる．測定は，TCP 通信を行うクライアントサーバプログラムによって行う．送信側を測定対象とし，受信側として他の計算機を用意する．送信は send システムコールを用い，測定中は send に与えるデータサイズを一定とする．受信は recv システムコールを用い，バッファサイズは send に与えるデータサイズと同じとする．そして，2 つの計算機間の通信が確立してから，データを送受信し終えるまでの時間を測定する．

また，通信時における他プロセスの演算処理性能を測定する．擬似 NIC 実装と監視実装において，通信処理時間の測定に用いたクライアントサーバプログラムと同時に，演算処理としてループ処理を行うプログラムを走行させる．そして，通信の開始から終了までの間に実行されるループ処理の繰り返し回数を測定し，単位時間当たりの繰り返し回数から，送信監視機能の実装による他の演算処理への影響を評価する．

5.2 測 定 環 境

送信側の測定環境は，CPU:Pentium4 3.0GHz, OS: Linux Kernel 2.4.7, NIC:3com 3c905-TX である．受信側には Red Hat Linux 7.2 を用い，CPU:Pentium4 3.0GHz, OS : Linux Kernel 2.4.7, NIC : I-O DATA ETX-PCI で測定した．伝送路は 100Mbps であり，測定に使用した計算機の MTU(Maximum Transmission Unit) は 1500byte である．計算機は全てシングルユーザモードで起動させ，専用の LAN を構築して測定を行った．また，通信処理時間の測定には，CPU のクロックカウント値を出力する rdtsc 命令を用いた．

5.3 通信処理時間

send に与えるデータサイズを 1〜1024byte の間で変化させ，1Gbyte 分のデータを送受信する時間を測定した．各データサイズで 10 回測定した平均の通信処理時間を図 9 に示す．

図 9 より，オリジナルと擬似 NIC 実装を比較すると，データサイズが 16byte 以上の場合では両者の処理時間はほぼ等しい．これは，データサイズが大きくなると，CPU の処理速度が NIC の処理速度を上回る

表 2　演算処理の繰り返しの回数 (単位：繰り返し回数/3.0*10^9 クロックサイクル)

OS ＼ データサイズ	32byte	64byte	128byte	1024byte
擬似 NIC 実装	55945492	71036048	78895868	86240923
監視実装	55570483	70883741	78603882	85987343

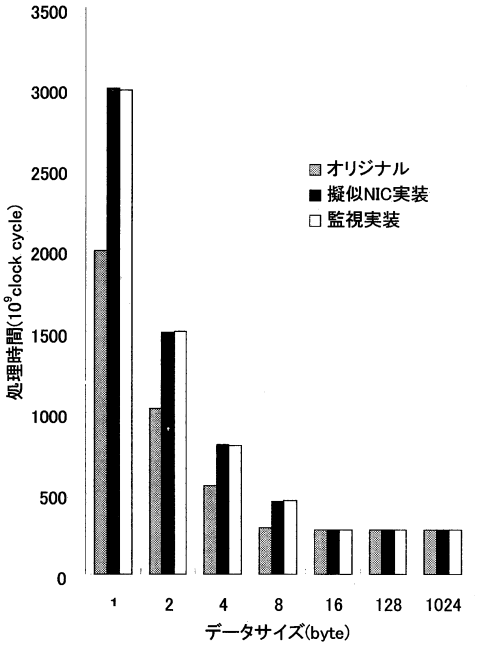

図 9　通信処理時間

ためである．一方，データサイズが 8byte 以下の場合，擬似 NIC 実装の処理時間はオリジナルの約 1.5 倍となっている．これは，先行 OS と共存 OS 間での OS 切り替え，擬似 NIC 機能による I/O 命令と割込みの横取りが原因である．しかし，実際の通信において，この処理時間の差は問題にならない．なぜならば，通常スループットを必要とする通信を行う場合，パケットはサイズをできるだけ MTU に近くなるように大きくして送信されるからである．

擬似 NIC 実装と監視実装を比較すると，両者の処理時間はほぼ等しく，0.3～0.5%のわずかな差を認めた．この差は，測定値の揺らぎの範囲に収まるものであったため，監視処理による通信性能への影響はないとわかる．

5.4　他の演算処理との相互影響

send に与えるデータサイズを 32～1024byte の間で変化させて 128Mbyte 分のデータ通信を行い，同時にループ処理を動作させた．そして，プロセッサの 3.0G クロックあたり (つまり約 1 秒) の繰り返し回数を求め，表 2 に示す．

データサイズを増加させると，いずれの場合も単位時間当たりの繰り返し回数が増加する．これは，データサイズが大きくなると，1Gbyte のデータ送信で処理される send の数が減少するからである．この結果，NIC がボトルネックとなる通信環境では，プロセッサのアイドル時間が増加する．そして，増加したアイドル時間は，ループ処理に割り当てられるため，単位時間当たりの繰り返し回数が増加する．

両者の測定結果を比較すると，監視実装の繰り返し回数は，いずれのデータサイズでも擬似 NIC 実装より，0.21～0.67%の低下が見られた．これは，送信処理における，共存 OS から先行 OS への送信管理表の複写処理とその管理による影響が大きい．

6. おわりに

本稿では，TwinOS 上で共存 OS が行う通信を先行 OS から監視する機能として，送信パケット監視機能について述べた．送信パケット監視機能では，TwinOS の利用法の 1 つである擬似 NIC 機能を利用した．そして，3 つの監視方式を示し，その内，先行 OS 上に共存 OS の送信管理表を複写して NIC に通知する方法を採用した．本方式は，共存 OS の送信管理表を書き換えることなくパケットの監視を行えるためである．

また，評価として通常の Linux，擬似 NIC 機能のみを実装した TwinOS，および監視機能を実装した TwinOS の通信処理時間を測定し比較した．擬似 NIC 機能を実装した TwinOS は，Linux と比べ，十分な通信性能を得られた．監視機能を実装した TwinOS は，擬似 NIC 機能のみの場合と比べたところ，両者の通信性能に差は認められなかった．また，送信処理と平行して動作する他プロセスの演算処理量に対する，監視処理の影響を評価した．結果，監視機能を実装した TwinOS は，擬似 NIC 機能のみの場合と比べ，0.21～0.67%の演算処理量の低下が見られた．

残された課題として，実際にパケットのログ取得，破棄処理を行った場合の性能測定がある．

参考文献

1) “情報セキュリティ白書 2007 年版，” http://www.ipa.go.jp/security/vuln/documents/2006/

ISwhitepaper2007.pdf (2007).
2) “警察庁技術対策課:Dos/DDoS 対策について,” http://www.cyberpolice.go.jp/server/rdenv/pdf/DDoSInspection.pdf (2003).
3) “警察庁技術対策課:Dos/DDoS 対策について (検証),” http://www.cyberpolice.go.jp/server/rdenv/pdf/DDoSInspection 2.pdf (2004).
4) Martin Roesch, “Snort-Lightweight Intrusion Detection for Networks,” Proc. of LISA '99: 13th Systems Administration Conference, pp.229-238, Nov. 1999
5) Carl A. Waldspurger, “Memory Resource Management in VMware ESX Server,” Proc. of the 5th Symposium on Operating Systems Design and Implementation, pp.181-194, Dec. 2002
6) 田渕 正樹, 伊藤 健一, 乃村 能成, 谷口 秀夫, “二つの Linux を共存走行させる機能の設計と評価,” 電子情報通信学会論文誌 (D-I), vol.J88-D-I, no.2, pp.251-262, Feb. 2005.
7) 乃村 能成, 山本 裕馬, 谷口 秀夫, 桝本 圭, 伊藤 健一, “PCI デバイスを監視する統合的な処理機構の提案,” 情報処理学会論文誌, vol.47, no.SIG 12(ACS 15), pp.430-439, Sep. 2006.