Vol.2011-ARC-194 No.8
2011/3/11

アプリケーションゲートウエイにおける処理オフロードの実現

八巻隼人[†1]　原島真悟[†3]
鯉渕道紘[†2]　西宏章[†3]

近年におけるルータの研究では，アプリケーション処理を実行可能な多機能ゲートウエイが開発されつつある．ゲートウエイにおいて，アプリケーション処理の実行は全てのトラフィックデータに対して行われるため，このようなゲートウエイにかかる負荷は大きい．本稿では，従来ホストで行っていた処理をゲートウエイへオフロードする手法の提案を行い，その際のパケット送受信に関してスループットの評価を行った．本評価では，パケット受信において 2.140Gbps のスループットが得られた．

Achievement of offload processing for application gateways

Hayato Yamaki,[†1] Shingo Harashima,[†3]
Michihiro Koibuchi[†2] and Hiroaki Nishi[†3]

Recently, multi-functional gateways are being developed. Those are able to execute application processes. Because the execution of the application processes is done for every traffic data, the load of the gateways is getting heavier. In this paper, we propose a method of offload processing to the gateways, the process had ever been executed by hosts, and we evaluated a throughput of packet transceiver. In this evaluation, we got 2.140Gbps throughput.

†1 慶應義塾大学理工学部システムデザイン工学科
Dept.of System Design, Keio University
†2 国立情報学研究所
National Institute of Informatics
†3 慶應義塾大学大学院理工学研究科
Graduate School of Science and Technology, Keio University

1. はじめに

近年インターネットは飛躍的に発展し，我々にとって，なくてはならない情報伝達手段となっている．総務省の調査によると，平成 22 年度のインターネット利用率（世帯内の誰かがパソコン，携帯電話，携帯情報端末，ゲーム機等から過去 1 年間にインターネットを利用したかどうか）は 92.7%であり，平成 13 年度におけるインターネット利用率 34.0%を 58.7%も上回っている[1)]．この調査結果は日本の 9 割以上の世帯がインターネットを使用しているということを示しており，インターネットが我々の生活に大きく寄与していることが伺える．インターネット利用率の上昇に伴い，そこを飛び交う情報量も年々増加している．総務省の調査によると，平成 22 年度のブロードバンドサービス契約者のダウンロードトラフィック総量[⋆1]は推定で 1.45Tbps となっており，平成 15 年度におけるダウンロードトラフィック総量 0.30Tbps と比べ，およそ 5 倍ものトラフィックデータが流れていることになる[2)]．

このような状況にある現在の情報ネットワークにおいて，ネットワーク中のトラフィックデータの中継機器としての役割を担っているのがルータである．ルータの役割として，主にパケット転送やプロトコル変換が挙げられる．従来のルータに関する研究ではパケット転送のスループットや QoS など，情報をいかに速く正確に伝達するかが目標であった．しかし，近年のトラフィック量の増加に基づくルータの研究により，ルータは進化し，パケット転送以外の機能も併せ持つようになった．パケット転送以外の機能としては，例えばアプリケーションゲートウェイやレイヤ 7 ファイアウォールといった，アプリケーションプロトコルのレベルで通信制御や検査を行う機能がある．このような，アプリケーションレベルで検査を行う機能は，パケットのペイロードを含めて検査しなければならないため，大きな負荷がかかる場合がある．従って，アプリケーションレベルで検査を行う機能がアプリケーションの処理上も，通信上もボトルネックとなることがある．そこで本稿では，このようなアプリケーションゲートウェイにおける，アプリケーションワークロードのオフロード手法を提案し，評価する．

2. 処理オフロードの概要

本節では，提案する処理オフロード手法について述べる．従来のアプリケーションゲート

⋆1 1 日の平均トラフィックの月平均

Vol.2011-ARC-194 No.8
2011/3/11

ウェイには，図 1 に示すように，流れる全てのトラフィックデータに対して処理を行わなければならず，大きな負荷がかかることが知られている．そこで，本研究では搭載されているアプリケーションソフトウエアが処理しなければならないデータ量を減らすことで，アプリケーションワークロードのオフロードを行う．本手法では，アプリケーションワークロードのオフロードを，PCI Express FPGA（Field-Programmable Gate Array）ボードを用いることで行う．FPGA とは，ユーザが望む論理回路を容易に，また何度でも書き換えることが可能なデバイスである．ユーザは HDL 等の言語で回路を設計し，論理合成ツールを用いてデザインの合成，論理シミュレーションを使用してデザインの検証を行う．デザイン合成を完了し，デザインインプリメンテーションを施し，FPGA にビットストリームファイルをダウンロードすることで，実機上で動作を行うことが可能となる．PCI Express FPGA ボードでは，FPGA ボードとゲートウェイ間を PCI Express により接続することができる．PCI Express とは，PCI バスの欠点を補うべく開発され，1 レーンあたり数百 MB/s から数 GB/s の転送速度を持ち，レーンを複数重ねることで更に高転送速度を可能にするインタフェースである．提案手法では，FPGA にアプリケーションのアクセラレータを実装することで，トラフィックデータをより負荷の少ない状態に改変し，アプリケーションへと転送する．

一般に，PCI Express 接続された FPGA ボードとゲートウェイ上で実行されているオペレーティングシステムおよびアプリケーション間でパケット転送を行うには，PCI Express インタフェースに介入するルーチンを内包するソフトウェアを作成する必要がある．これは，PCI Express インタフェースを介したデータ転送を行う場合，PIO アクセスや DMA アクセスといったユーザ空間で扱うことができないリソースを扱わなければならないためであり，結果としてユーザがこのようなアプリケーションを自由に設計・運営することが困難となる．またゲートウェイにおけるアプリケーションは，パケットに対して処理を行うため，イーサネットポートを介する場合が多い．従って，FPGA ボードとゲートウェイを PCI Express 接続しただけでは，FPGA でオフロード処理が行われたデータをアプリケーションへ転送することは難しい．そこで本稿では，仮想イーサネットワークデバイスを用いて上記の問題点を解決する．本仮想イーサネットワークデバイスは，PCI Express インタフェースを内包し，イーサネットのプロトコルを用いて FPGA ボードとゲートウェイ間でのパケット転送を担う．本提案手法では図 2 に示すように，ゲートウェイへと流れてくるパケットに対して，FPGA ボードを介してゲートウエイへと転送することでオフロード処理を行う．転送されたパケットは，FPGA に実装されたアクセラレータによりオフロード

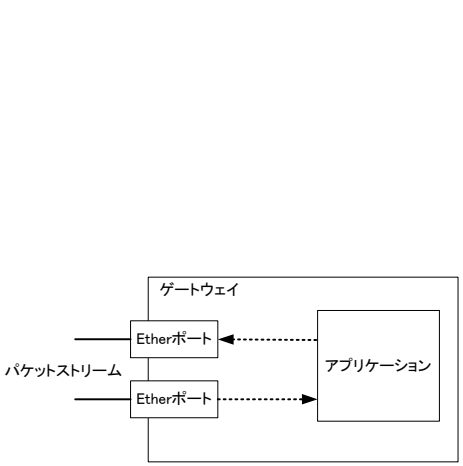

図 1 通常時のアプリケーションワークロード
Fig. 1 Ordinary application work load.

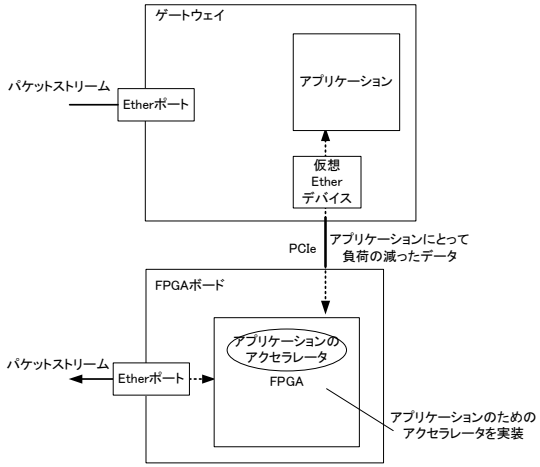

図 2 提案手法によるオフロード処理の概要
Fig. 2 Description of offload processing in proposal technique.

処理がなされ，仮想イーサネットワークデバイスを介してゲートウェイへと送れられる．そしてアプリケーションプログラムは，この仮想イーサネットワークデバイスを介して，オフロード処理されたパケットを受け取り，通常の処理を行うことで，ワークロードを減らすことが可能となる．この時，特にアプリケーションプログラムには修正を施す必要がない．

3. 仮想イーサネットワークデバイス

本節では，本研究で実装を行う仮想イーサネットワークデバイスである，vether(virtual ether) の解説を行う．vether は Linux のネットワークドライバとして設計を行う[3]．Linux のネットワークドライバは，net_device 構造体としてデバイスを作成し，Linux カーネルに登録する．登録されたデバイスは，ifconfig により IP アドレスの割り当てが可能で，これにより vether を eth0 や lo のようなイーサネットポートとして扱うことが可能となる．

次に，vether のパケット転送処理に関して詳細を述べる．Linux ネットワークデバイスでは，net_device 構造体を作成する際，デバイスが操作する関数をメソッドとして net_device 構造体に登録する．vether では，ifconfig 時にインタフェースをオープンする open メソッド，インタフェースの停止を行う stop メソッド，パケット送信を行う hard_start_xmit メソッ

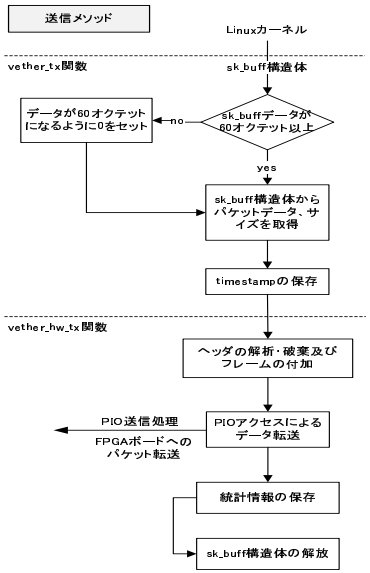

図 3 vether の送信メソッド
Fig. 3 Method of packet transmitting.

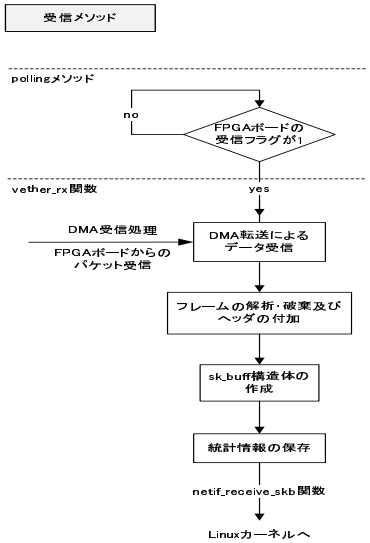

図 4 vether の受信メソッド
Fig. 4 Method of packet receiving.

ド，ハードウェアヘッダの付加を行う hard_header メソッド，インタフェース固有の ioctl コマンドを実行する do_ioctl メソッド，統計情報の提供を行う get_stats メソッド，インタフェースの設定を変更する set_config メソッド，送信タイムアウトの処理を行う tx_timeout メソッド等を登録しており，vether にパケットが到着すると，これらの関数が自動的に呼び出され，適切な処理が行われる．なお，vether はゲートウェイから FPGA へとパケット転送を行う送信メソッドと，FPGA からゲートウェイへとパケット転送を行う受信メソッドを個々に実装している．図 3 や図 4 に示すように，送信メソッドでは PIO アクセスにより，受信メソッドでは DMA アクセスによりパケット転送が行われる．ルーティングによりパケットが vether デバイスへと送られると，vether の送信メソッドが呼び出される．送信メソッドでは，パケットサイズの確認や待ち列の制御，パケットヘッダの解析といった転送パケットの制御，PIO アクセスによる FPGA へのパケット転送処理，送信パケット情報の更新等が行われる．また，FPGA 内ではパケット受信フラグが実装されており，FPGA に実装されたアクセラレータによりパケットが処理されると，この受信フラグが立ち，vether の受信メソッドが polling により呼び出される．受信メソッドでは，送信メソッド同様の転送パケットの制御，DMA アクセスによるゲートウェイへのパケット転送処理，受信パケット情報の更新等が行われる．DMA アクセスによる受信処理では，データ転送を行う前に DMA セットアップを行う必要がある．また，データ転送完了時には FPGA 側の割り込み処理により，DMA の完了が通知される．

表 1 遅延の測定結果
Table 1 Result of delay.

測定範囲		遅延 [ms]
送信メソッド（PIO 送信処理を除く）		6.524×10^{-1}
受信メソッド（DMA 受信処理を除く）		6.018×10^{-1}
PIO 送信処理		1.836
DMA 受信処理	セットアップ時まで	2.833
	転送完了時まで	3.440
	データ転送期間	5.418×10^{-3}

4. 実験と評価

本研究では，vether のパケット送受信の動作確認および遅延の測定を実験により行った．実験は，Linux Cent OS 5.5，Intel Xeon X5450 CPU @3.00GHz，4GByte Memory 環境により行った．また遅延の測定は，Agilent Technologies 社製の 16823A ロジック・アナライザを使用し，FPGA 内のデータ信号線を観測することにより行った．パケットの送信では，ping により 64Byte のパケットを vether を介して FPGA へと転送した．この際，vether の PIO 送信処理にかかる遅延と送信メソッド全体（PIO 送信処理を除く）にかかる遅延の測定を行った．パケットの受信では，FPGA 内に受信フラグを立てることで，FPGA 内のデータを vether を介して PC へと転送した．この際，vether の DMA 受信処理において，セットアップ完了までにかかる遅延と DMA 転送完了までにかかる遅延，そしてデータ転送期間の遅延の測定および受信メソッド全体（DMA 受信処理を除く）にかかる遅延の測定を行った．また，本実験において，DMA 転送によるデータ転送のデータサイズは 1,556Byte である．本実験の遅延の測定結果を表 1 に示す．

表 1 の結果より，PIO 送信処理における遅延の発生は 1.836ms である．従って，64Byte のパケット転送に 1.836ms かかるということであり，この値から PIO 送信処理におけるスループットは 272.3Kbps となる．受信処理における遅延の測定結果では，DMA 受信処

理全体で 3.440ms の遅延が発生しており，その中でもセットアップ完了までに 2.833ms の遅延が発生している．これより，DMA 受信処理の遅延の大半がセットアップにかかっていると言える．また，データ転送期間の測定結果より，受信処理のスループットを求めると 2.140Gbps となる．実際にはセットアップや転送完了後に割り込みが発生する時間 (表 1 におけるセットアップ時まで，転送完了時までの遅延結果) を考慮すると，スループットは 19.56Mbps まで低下する．これに対し，スループットを改善するいくつかの方法が考えられる．以下にスループット改善策をあげる．

(1) 現在の DMA データ転送サイズである 1,556Byte を増やす．
例えば，1 回の DMA データ転送サイズを 1MByte にしても，セットアップや転送完了割り込みにかかる遅延は変わらないため，スループットの向上が期待できる．

(2) DMA 転送時の TLP パケットのペイロードを増やす．
DMA 転送では，転送パケットを更にデータサイズの小さい TLP パケットに分割してから送るため，分割された分だけ転送回数が増えることになる．そこで，TLP パケットのペイロードサイズを増やすことで，転送回数は少なくなり，データ転送にかかる遅延も減少する．従って，スループットの向上が期待できる．

(3) PCI Express のレーン数を増やす．
本研究で使用している PCI Express は 4 レーンを持っており，更にレーン数を重ねることで PCI Express のスループット自体の向上が期待できる．

(4) ソフトウェアドライバを改善する．
ソフトウェアドライバのチューニングを行い，セットアップにかかる時間を減らすといった方法も考えられる．

vether の送信メソッドにおいて，実験の測定結果より，PIO 送信処理では実ネットワークにおける使用に対してスループットが十分ではないことがわかった．そこで，送信メソッドにおいて，DMA アクセスによる送信処理の検討を行う．本実験では，1,556Byte のパケット送信を行い，遅延の測定を行った．現状における DMA 送信処理では，DMA 転送期間の遅延結果は 2.777×10^{-2}ms となり，この値からスループットを求めると 432Mbps となる．

また，実際のオフロードにおける処理コスト低減の可能性について，文字列検索を行うアプリケーションを想定して検討を行う．一般的な，決定性有限オートマトンを用いて文字列検索を行うソフトウエアアプリケーションでは，1.35Gbps のスループットが得られることが文献よりわかっている[4]．また，同文献より，このアプリケーションを SIMD を用いて高速化したソフトウエアにおいて，スループットは 5.11Gbps であることがわかっている．これに対し，文字列検索のハードウエア実装では，22.1Gbps のスループットが得られている[5]．このことから，高速化を行ったソフトウエアを利用する場合と比しても，ハードウエアでは 4.32 倍の速度で実行可能であることが示されている．PCI Express の転送遅延やスループットを考慮しても，複雑な文字列検索を有する場合は，提案したオフロード機構によって有利な状況も想定できると言える．

5. 結論と今後の課題

本稿では，アプリケーションゲートウェイへの親和性の高い処理オフロード手法として，PCI Express FPGA ボードと仮想イーサネットワークデバイスを用いたオフロード手法の提案を行い，仮想イーサネットワークデバイスにおけるパケット送受信処理の評価を行った．仮想イーサネットワークデバイスの DMA 転送処理において，100Mbps を十分サポートできるスループットを得た．今後の取り組みとしては，現在検討を行なっている DMA アクセスでの送信処理の実装を行い，ネットワークの実トレースを利用し，アプリケーションの実装を行った上での評価を予定している．

謝辞 この研究は，独立行政法人情報通信研究機構の高度通信・放送研究開発委託研究「ネットワーク仮想化を活用したデータサービスアプリケーション基盤技術に関する研究開発」並びに「新世代ネットワーク技術戦略の実現に向けた萌芽的研究」および文部科学省科学技術研究費補助金基盤研究 C「安心・安全な情報提供を可能とするインターネット基盤の構築に関する研究」の一環としてなされたものである．

参 考 文 献

1) 総務省 通信利用動向調査,
http://www.soumu.go.jp/johotsusintokei/statistics/statistics05.html.
2) 総務省 我が国のインターネットにおけるトラヒックの集計・試算,
http://www.soumu.go.jp/menu_news/s-news/01kiban04_01000001.html.
3) Jonathan Corbet and Alessandro Rubini. *Linux Device Drivers*. O'Reilly Media, 3rd. edition, (2005).
4) Daniele Paolo Scarpazza, Oreste Villa and Fabrizio Petrini : Peak-Performance DFA-based String Matching on the Cell Processor, IEEE International Parallel and Distributed Processing Symposium (2007).
5) Alan Kennedy, Xiaojun Wang, Zhen Liu and Bin Liu : Ultra-High Throughput String Matching for Deep Packet Inspection, Design, Automation & test in Europe Conference & Exhibition (DATE) (2010).