Vol.2015-IOT-30 No.8
2015/7/4

FPGA によるOpenFlow スイッチ遅延計測器の開発

安田 豊[1]　三好 健文[2]　船田 悟史[2]

概要：OpenFlow スイッチでは設定するフローエントリの内容次第で予期せず大きな性能変化が生じる場合がある．たとえばスイッチ ASIC が直接処理できるエントリ内容ではマイクロ秒単位の遅延で転送するが，ASIC で対応できない内容では制御用 CPU のソフトウェアで実現するために遅延がミリ秒に達する．これを見逃すことは運用時の大きな障害につながりかねない．しかし多様なマッチパターンとアクションによって発生する遅延について網羅的に把握することは困難であるし，各サイトで必要な組み合わせに絞って自己検証しようにも高精度な遅延計測装置は一般に高価である．そこで我々は FPGA を用いて 10Gbps 環境において 8ns 単位で遅延計測が可能な機器を開発した．本稿ではその設計と実装について示す．

キーワード：FPGA, OpenFlow，スイッチ，遅延計測

Development of a FPGA-based measurement tool for latency on OpenFlow switch

YASUDA Yutaka[1]　MIYOSHI Takefumi[2]　FUNADA Satoshi[2]

Abstract: On OpenFlow based switch, the forwarding latency has a large differences depends on the flow entry configurations of it. There are some configurations that make a large delay unexpectedly. Typical reason is the capability of the switch ASIC. It can forward the packet in microseconds if the configuration fits the feature of ASIC. If it does not fit, it takes milliseconds. But it is hard to comprehend the latency in all cases. It is possible to measure limited configurations on each site as their limited use cases, but the high precision measurement equipment is expensive, generally. We have developed a FPGA-based latency measurement tool with accuracy of 8ns on 10Gbps connection. This paper shows the design and implementation of it.

Keywords: FPGA, OpenFlow, switch, latency measurement

1. はじめに

OpenFlow スイッチを実用のネットワーク環境に適用する際の問題の一つに，必要なパケット転送性能が出ていることを管理者（オペレータ）が直接的に確認する手段がないことがある．つまり従来的な L2/L3 ネットワーク機器では最大スループットや固定的な転送遅延といった単純な指標だけで性能予測が可能だったが，Rotsos[1] らが指摘したように OpenFlow スイッチでは設定するフローエントリ，すなわちマッチフィールドとアクションの組み合わせによって性能が大きく異なり，その予測が困難である．

そうした性能差が生じる原因は様々であるが，典型的な例の一つはハードウェアスイッチが転送処理をスイッチ ASIC で直接処理する場合と，制御用 CPU で処理する場合での遅延の違いである．ASIC がマイクロ秒単位の遅延で転送するところを，制御用 CPU つまりソフトウェアによって処理する場合はサブミリ秒あるいはそれ以上の遅延が生じてしまう．

また OpenFlow という共通インタフェイスがあるにもかかわらず，全てのフローテーブルの設定について事前にその処理能力を調べてデータベースを構築することは，項目

[1] 京都産業大学
Kyoto Sangyo University
[2] (株) イーツリーズ・ジャパン
e-trees.Japan, Inc.

と組み合わせの多さから困難だと指摘されている [2]．この遅延はチップドライバやスイッチ OS のバージョンによって変化する可能性もある．この問題に対応するには各サイトにおいて自分達が利用するフローエントリに限定した転送能力を計測・検証することが有効であるが，高精度なネットワーク機器の試験装置は一般に高額で各サイトに常備することが難しい．

しかしこれを見逃すことは運用時の大きな障害につながりかねない．そこで我々は FPGA を用いて 10Gbps 環境において 8ns 単位で遅延計測が可能な，ネットワーク管理者が自サイトに設置できる程度に安価な機器を開発した．

以下，本稿ではその設計と実装について示す．まず 2 章で問題の整理を行う．3 章で提案手法について説明し，4 章で実装の詳細について述べる．5 章では開発した機器による計測結果を示して，6 章で論文をまとめる．

2. 問題の整理

2.1 ASIC と CPU 処理による性能差

フローエントリの内容によって転送遅延が大きく異なる典型的な例として，ハードウェアスイッチが転送処理をスイッチ ASIC で直接処理する場合と，制御用 CPU で処理する場合での遅延の違いが知られている (文献 [1], [3], [4])．設定するマッチパターンおよびアクションを実行する機能がスイッチのハードウェアリソース，つまり利用している ASIC の機能として存在しないと，スイッチはその処理を自身の制御用 CPU に実行させる場合がある．しかしスイッチが制御用に搭載している CPU は一般的なサーバ用プロセッサと比較して低性能であり，フロー単位のパケット処理などには適していない．さらにフローエントリの追加や統計情報の提供など本来のスイッチ制御・管理のためにその処理能力をとられているため，パケットの転送時間は遅く，また不安定とならざるを得ない．結果的にハードウェアで処理した場合はマイクロ秒の遅延で転送するところを，制御用 CPU つまりソフトウェアによって処理する場合はサブミリ秒あるいはそれ以上の遅延が生じてしまう．

またスイッチ ASIC 内部の各パケット処理機能はそれぞれ典型的な L2, L3 スイッチング処理に特化して用意されているため，当該機能がもし ASIC にあったとしても OpenFlow が可能にする柔軟な機能の組み合わせのすべてで利用できるとは限らない．このことについては具体的な例とともに 5.3 に後述する．

2.2 問題となるエントリの発生

OpenFlow をはじめとする SDN スイッチではパケット処理に関する設定操作はオペレータではなくコントローラなどのソフトウェアが自動的に行う．その結果，ある日コントローラが全く新しい種類のエントリを設定したところ，2.1 で示したように対象スイッチがそれをソフトウェアで処理したために転送能力の急激な劣化あるいはパケットの喪失が生じ，かつオペレータがそのことに気づかない，といった事態が生じる可能性がある．

コントローラ・アプリケーションあるいはスイッチのファームウェアをアップデートした場合にも同様の状況，つまりアップデート前は正しく機能していたのに，それらのアップデートによって低速な転送が生じる，といった場合がありうる．

2.3 網羅的検証の困難さ

2.2 に示した問題はコントローラ・アプリケーションとスイッチハードウェアがもつ機能とのミスマッチから生じるものであるが，OpenFlow の規格を両者共にまったく逸脱していないことに注意が必要である．OpenFlow ではパケット操作の機能について定義しているだけで，その処理速度については何も規定していない．スイッチは自らの機能について Features Reply によって表明することが可能であるが，そこには個々の機能を実現可能であるか否かが示されるだけで，処理能力に関する情報は含まれていない．そのためコントローラは能力面で問題のある機能の使用を避けてフローエントリの内容を決定するといった調整ができない．

OpenFlow プロトコルの範囲内で対話的にスイッチの能力情報を得ることができなくても，予めスイッチの能力情報をコントローラに与えておく対処法が考えられる．しかし OpenFlow という共通インタフェイスがあるにもかかわらず，そのスイッチ性能の違いは多様であり，あらかじめ全てのフローテーブルの設定項目について網羅的に調べてデータベースを構築することは困難だと指摘されている [2]．

2.4 OF-PI・新しい設計

OpenFlow 仕様の先進的な実装はソフトウェアベースであり，ハードウェアスイッチへの実装は進みが遅い．大きな要因の一つは頻繁な仕様の拡張であるが，これは研究の進展とともに必要な機能を積極的に取り込んだ結果であり，短絡的に仕様変更間隔を長くすることは研究の足を止めるに等しい．この問題に真正面から取り組む研究が，仕様とスイッチ ASIC の処理機能の結合を弱め，非依存性を高める（OpenFlow2.0 と総称される）McKeown らの提案 [5][6] であり，2014 年 9 月には OF-PI (Protocol Independent Layer) となった [7].

しかし McKeown らのアプローチは多種多様なハードウェアに合わせた数多くの実装を生む．たとえば P4[6] 言語では注目するフィールドそれぞれに最大エントリ数を指定し，それに応じてハードウェアリソースの割り当てが変わってしまうが，そうしたリソースの使い方によって転送遅延が直接的に影響を受けることが以前から示されてい

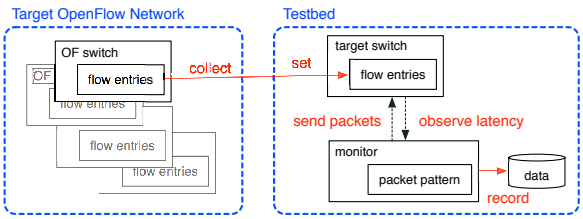

図 1　コンポーネントの配置と計測手順
Fig. 1　Components Layout and Measurement Procedures

る [8].

こうした次々と新しい機能が追加される状況は，よりフレキシブルなパケット処理パイプラインの導入を誘導すると指摘されており [9]，実際に XPliant[10] などプログラマブルなチップが現れつつある．これらのことから実際の運用環境におけるスイッチの転送能力予測は今後さらに困難になると考えられる．

3. 提案

2 章に述べた問題に対応するために，筆者らはスイッチが期待する性能で機能していることを，各ユーザ・サイトごとに随時検証できる環境を用意することを提案する．本研究ではそれを可能にする高精度で比較的安価な遅延計測装置を開発した．次節以降にその概要を示す．

3.1 ユーザ・サイトにおける性能計測

2.2, 2.3 で述べたように，大きな性能差の発生など障害の原因となり得る事象が起き得ないことをすべてのコントローラ・ソフトウェアとスイッチの双方について，網羅的な検証によって事前に明らかにすることは困難である．そこで本研究では各オペレータが自サイトにおいて安定した実運用環境を維持することに焦点を絞り，検証の対象を実際に使用されているフローエントリとスイッチの組み合わせに限定することとした．

つまり図 **1** に示すような環境において，(1) 実際に運用されているスイッチからフローエントリ情報を抽出し，(2) これをテストベッド上のスイッチに移したうえで，(3) 計測器 (図中の monitor) から検査用のパケットを送信して，転送遅延を計測する．なお計測器は負荷に対する遅延の変化も計測するために，指定した長さと送信間隔の検査用パケットを連続的に送信できなければならない．

3.2 計測精度

計測に際して，どの程度の転送遅延を検出すべきと判断するかはアプリケーションによるが，VoIP やオンライン・マルチメディアゲーム[*1]などではモニタリングすべき遅延とジッターについて 10 マイクロ秒単位の精度が必要だとする指摘もある [11][12] .

本研究では対象とする SDN スイッチの主たる適用先の一つがデータセンターであることと合わせて，計測精度については 10Gbps のインタフェイスにおいて 10 マイクロ秒程度を設計ゴールとして設定した．

3.3 実現技術

計測を行う方法は幾つか考えられる．

(1) 既存の専用計測器を用いる
(2) 一般的な PC を用いてソフトウェアで計測する
(3) FPGA ボードと一般的な PC を用いてハードウェアで計測する

(1) は非常に高精度な結果が信頼性高く得られるが，各サイトで購入・運用することが困難なほど高価である．従来的にもこれらの専用計測器は主としてネットワーク機器の開発企業などが購入・レンタルして運用してきたものであり，ユーザ向けでは無い．

(2) は安価であるが 3.2 で示した領域の精度を得ることは容易でない上に，達成した精度を継続的に維持できていることを検証する必要がある．

そこで本研究では (3) の方法を採る．すなわち遅延測定といった精度や性能を担保する必要がある処理はすべて FPGA に閉じ込めて実装し，ホストとなる PC の OS, CPU 性能などに起因する精度の不安定さを生じさせないようにする．FPGA は PCIe インタフェイスを通じてホスト PC に接続するが，ホストが行うのは設定処理や計測開始・データ回収といった制御処理だけであり，計測処理 (遅延の計算・記録・集計処理) はすべて FPGA 内で行う．そのためホスト PC への性能要求は厳しくなく，インストール作業も開発した計測器システムでは単に FPGA ボードを入れて設定ソフトウェアを走らせるだけでよい．ハードウェアや OS 設定の僅かな違いによって所定の性能を出せず計測結果が信頼できなくなる，といった検証が容易でないトラブルが生じることもない．

本研究で FPGA を利用するのは上記の性能問題だけでなく費用も理由の一つである．FPGA のコストは一般に (2) の方法よりは高くなるが，(1) より遥かに安価で，同等のハードウェアレベルでのタイムスタンプによる高い精度を得られる．(2) の環境でも IEEE1588[13] タイムスタンプ機能をもつ NIC を用いれば精度を高めることは可能であるが，一定間隔での連続パケット送信を含めてシステム全体で期待する精度が継続的に保てていることを検証する必要がある上，そうした NIC は比較的高価である．

3.4 システム構成

本研究で開発した計測器の具体的な構成を図 **2** に示す．3 で述べたように，遅延計測処理はホストとなる PC と PCIe で接続された FPGA ボード内で行われる．ホスト

[*1] またニッチではあるがビジネスとしては重要と断った上で HFT (High Frequency Trading) も挙げられている．

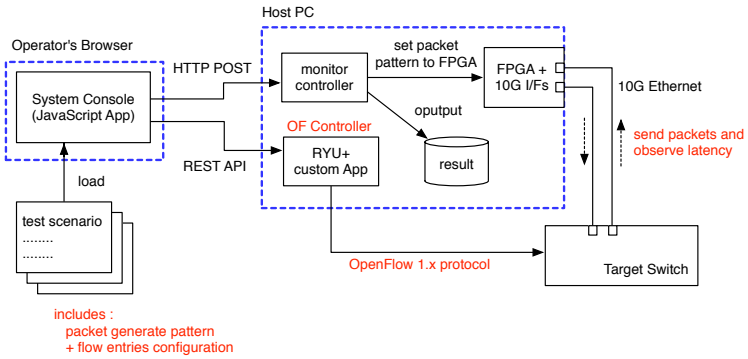

図 2　システム構成
Fig. 2　System Structure

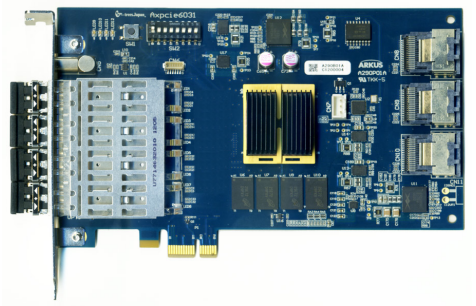

図 3　Axpcie6031

表 1　Axpcie6031 の仕様
Table 1　Axpcie6031 Specification

FPGA	Xilinx XC7K325T-2
オンボードメモリ	DDR3 2GB
イーサネットポート	4 ポート．10GbE 対応 SFP+ケージ
PCIe	Gen2 x1
オンボードオシレータ	156.25MHz
高速シリアル I/O	3 本
その他	DIP スイッチ 8bit，LED 8bit

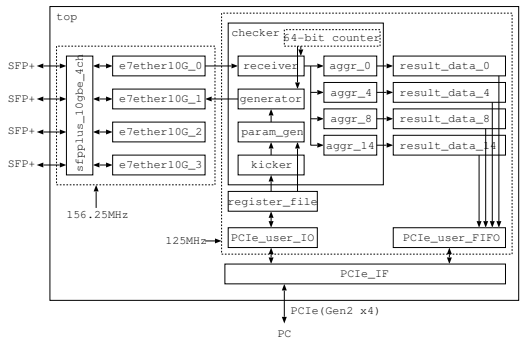

図 4　FPGA ボード内部構成
Fig. 4　Internal Structure of FGPA Board

PC 上ではパラメタ設定や計測開始指示・データ回収を行うが，この制御を行う専用プログラムをモニタ・コントローラと呼ぶ．ホスト PC はまた対象となるスイッチにフローエントリを設定するための OpenFlow コントローラともなる．コントローラには RYU[14] を用いた．システム全体の操作は Web アプリケーションとして作成されたシステム・コンソールの GUI を通して行う．

システム・コンソールの操作と処理の流れは以下のようになる．

(1) 一連の計測パターンが定義されたシナリオファイルを指定し，そこからフローエントリと送出するパケットのパターン情報を取得する
(2) 取得したフローエントリ情報を，RYU の REST API を通して計測対象のスイッチに設定する
(3) 取得した送出パケット情報を，モニタ・コントローラの独自 API を通して FPGA に設定する
(4) FPGA に計測開始を指示し，結果を受けとって画面上に表示する（並行してモニター・コントローラも計測結果をホスト PC に保存する）

3.5　計測結果の記録

2 章に示したように転送遅延が生じる原因はさまざまである．これを正しく捉え，可視化するには遅延時間の最小/最大/平均値などでは不十分であり，本計測器ではその分布を記録する．具体的にはヒストグラム表示が可能なように遅延の値をビンに分けて数える方法を採るが，2.1, 3.2 で述べたように 10ns 程度の精度から最長で秒単位となる転送遅延を固定的な時間幅で数えると記憶領域の利用効率が著しく低くなる．そこで以下のようにビンの時間幅が異なる四つのカウンタを用意し，非常な低遅延領域での精度の高さと，幅広い遅延時間幅の分布を記録する際の効率の良さを両立させる．

各カウンタの刻み幅と，それぞれがカバーする遅延時間の範囲を以下に示す．

- 8ns 刻み：0〜8,184ns を 1023 分割
- 128ns 刻み：0〜130,944ns を 1023 分割
- 2048ns 刻み：0〜2,095,104ns を 1023 分割
- 131,072ns 刻み：0〜134,086,656ns を 1023 分割

なお各カウンタごとに計測範囲を超えて遅く到着したパケットについても数えており，最終的に全パケットが受信されたことを確認できる．

4.　実装

特に重要な実装箇所である FPGA 部分についてその詳細を示す．

4.1　アーキテクチャ

利用した FPGA ボードは e-trees.Japan 製の Axpcie6031 である (図 **3**)．このボードの仕様を表 **1** に示す．4 ポートの SFP+への入出力は，それぞれ FPGA の高速シリアル I/O ユニット (GTX) に直接接続されている．

FPGA 内部のアーキテクチャを図 **4** に示す．図 4 の `checker` が検査装置としてのカーネル部分である．

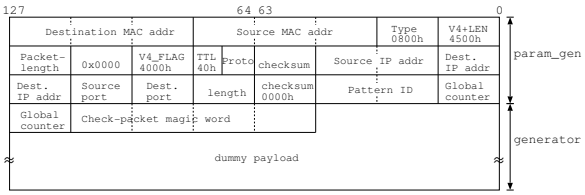

図 **5** 検査パケットのフォーマット
Fig. 5 Probe Packet Format

表 **2** リソース使用量
Table 2 Resource Usage

インスタンス	LUT	FF	RAM36
PCIe 関連ユニット	2678	2754	14
checker	2173	2002	0
e7ether10G	8609	8259	35
sfpplus_10gbe_4ch	8415	8156	0
デバッグ，その他	3069	7301	66
合計	27148	30474	115
資源消費率	18%	7.4%	26%

checker は，パケット生成用のパラメタと測定結果の受け渡しのために，PCIe を介してホストとなる PC と接続する．e7ether10G_0〜e7ether10G_4 および sfpplus_10gbe_4ch が，実際にパケット送受信を行うためのコンポーネント群である．FPGA の高速トランシーバユニットを介して SFP+ポートとデータをやりとりし，ワイヤレートでパケットを送受信できる．checker および PCIe 関連のコンポーネント群は PCIe 由来のクロックである 125MHz で動作し，e7ether10G_0〜e7ether10G_4 と sfpplus_10gbe_4ch は SFP+駆動用の 156.25MHz で動作する．両者間のデータは e7ether10G 内にある FIFO によって同期が取られる．

4.2 計測手法

4.2.1 検査用パケットの生成から送信まで

まず，register_file に格納されたユーザの指定したパラメタを元に，param_gen が，送信するパケットの送信元アドレスや送信先アドレス，ポート番号などのパケットヘッダを生成する．なお param_gen は 32bit の XOR-SHIFT 法による乱数生成器を持っており，アドレスやポート番号について一部あるいは全部がランダムなパケットヘッダの生成が可能である．生成されたパケットヘッダは，kicker によってユーザの指定した間隔で generator に与えられる．generator では，パケット送信時刻に相当する global_counter の値とマジックワードを検査情報としてペイロードに付与し，検査パケットを完成させる．生成された検査パケットは，イーサネットパケットとして SFP+ポートから送信される．例として，VLAN や MPLS の指定がない場合の検査パケットのフォーマットを図 **5** に示す．

4.2.2 検査用パケットの受信から遅延値の算出まで

イーサネットパケットを受信すると receiver でヘッダとペイロードを解析し，マジックワードが所定位置に含まれていることを確認する．対象パケットであれば送信時刻を取得して global_counter の今の値，すなわち現在時刻との差を遅延時間として求める．求めた遅延時間は，aggr_0 から aggr_14 の 4 つのユニットでビンに分類され，それぞれ FPGA 内部のブロックメモリによる保存領域である result_data_0 から result_data_14 に保存される．

4.3 リソース使用量

実装したシステムの主要なリソースの使用量を表 **2** に示す．合成には Xilinx Vivado 2014.1 を利用した．LUT，レジスタおよび RAM36 の使用割合は，実装に用いた XC7K325T-2 のリソースの 18%，7.4%および 26%に相当する．将来的な機能拡張に十分な空きリソースがあり，また，同シリーズの最小規模の FPGA にも実装可能な回路規模であり，安価な検査装置を実現するという目的に適していると考えられる．

5. 評価実験

5.1 ループバックテスト

まず計測器そのものが安定して遅延計測が行えることを確認するために，計測器の送受信ポートを直結した状態で計測した．計測にはデータペイロード 60 バイトの UDP パケット[*2]を用いている．その結果，送信したパケットの 90% が 800ns 以上 808ns 未満，残りはすべて 808ns 以上 816ns 未満の転送遅延で安定して受信され，パケットの喪失がないことを確認した．これ以降の全ての計測も同じ検査パケットによって行っており，そこで示す転送遅延にはこのループバック状態での遅延が含まれていることに注意されたい．

5.2 スイッチ性能の比較

最も単純なフローエントリ設定，つまり「ポート N に届いたパケットはすべてポート M に出力する」場合の転送遅延について，二種類のハードウェアスイッチで計測し，その比較を行った．Switch A, B は異なるメーカーで，使用しているスイッチ ASIC も違うものである．Switch A のポート構成は SFP+ が 4 つ，1000BASE-T が 48 ポートであり，Switch B は SFP+ が 4 つ，10G-T インタフェイスが 48 ポートある．

図 **6** に Switch A，図 **7** に Switch B の転送遅延の分布を示す．Switch A では 2.94 μ〜3.00 μ秒の範囲で山なりに転送遅延が分布しているのに対して，Switch B は 2.76 μ秒前後の極めて狭い範囲に遅延が集中していることがわ

[*2] Ethernet パケット長としては 102 バイトとなる．

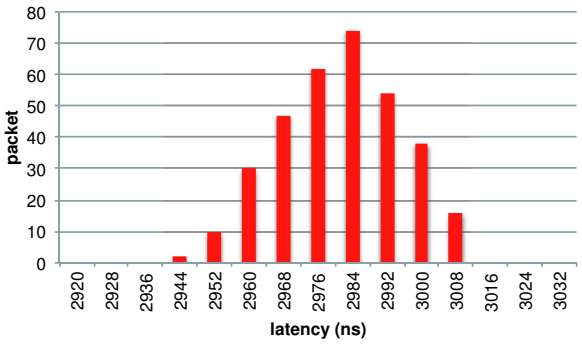

図 6　Switch A の転送遅延分布
Fig. 6　Distribution of the latency on switch A

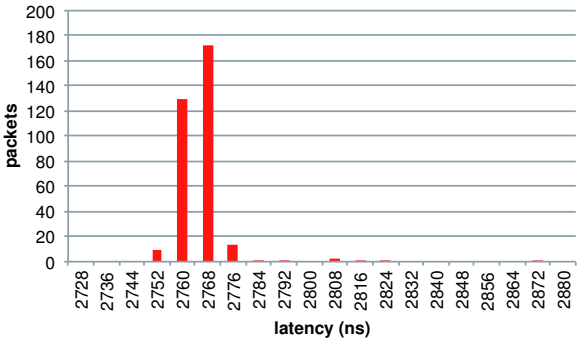

図 7　Switch B の転送遅延分布
Fig. 7　Distribution of the latency on switch B

かる[*3]．3.2 で提示した 8ns の精度によって Switch B の優れた性能を確認することができた．

5.3　フローエントリによる遅延の相違

1G Ethernet 48 ポートからなるハードウェアスイッチ C を用いて 2.1 に示したフローエントリの内容による性能差を確認した．

まず「ポート N に来たパケットはすべてポート M に出力する」簡単なフローエントリを指定した状態では，9 μ 秒前後の安定した転送遅延となることを確認した．しかしマッチパターンに特定の IP ソースアドレスを指定し，アクションに IP 宛先アドレスの書き換えを行うフローエントリ (表 **3** の #1) を設定した場合，96%のパケットの遅延は 350 μ〜470 μ 秒と広い範囲に，かつ二つの山型で分布するようになった．残りはグラフの右枠外に離散的に分布しており，2.9%のパケットの遅延は 1 ミリ秒を超え，最も遅いものは 10 ミリ秒に達した．この大きな遅延は指定したフローエントリが ASIC ではなく制御用 CPU のソフトウェアで処理されたことを示唆する．

次にこの遅延が IP SRC のマッチ処理に起因するものか，IP DST の書き換え処理に起因するものか調べるために，それぞれ一方だけを有効にした組み合わせ (同#2, #3)

[*3] 2.80 μ 秒前後, 2.87 μ 秒にそれぞれ僅かな分布があるが，全体の 2%程度に過ぎない上に，それらは Switch A の最良ケースよりもまだ低遅延である．

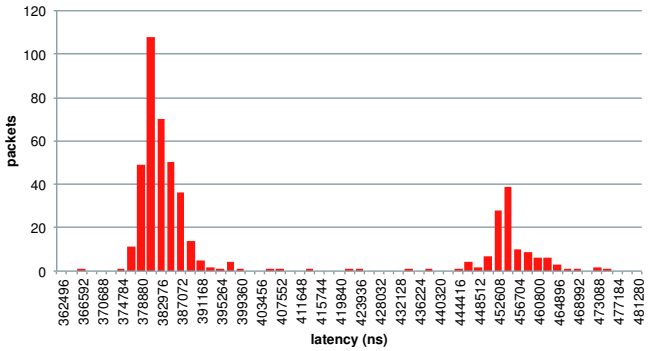

図 8　Switch C の転送遅延分布
Fig. 8　Distribution of the latency on switch C

表 3　フローエントリ内容による遅延
Table 3　Latency dependence on it's flow entry

	IP SRC Match	IP DST Mod	ToS Mod	latency
#1	✓	✓		slow
#2		✓		quick
#3	✓			quick
#4	✓		✓	quick

を試したが，その両方とも 9 μ 秒前後で高速に転送が行われた．つまり#1 で指定されたマッチ条件とアクション指示の二つの処理は，それぞれ個別には ASIC 内で実行可能だが，しかし両者を同時に機能させることができない[*4]と考えられる．また #4 に示すように IP SRC のマッチ処理と ToS の書き換え処理の組み合わせは高速に処理可能であることが確認できた．つまりマッチ処理とアクション処理のなかでも同時に ASIC で処理させることが可能な組み合わせも存在する．しかしそうした同時に機能させられない組み合わせが何であるかを事前に推定するのは容易ではなく，この結果は本研究の提案するユーザサイトにおける運用状況に限定した実測定の有効性を支持するものと考える．

さらに#1 に示したソフトウェア処理と推定されるフローエントリ設定では，パケット送出量が 16Kpps を超えると大量にパケット喪失することを観測した．今回の実験で用いた短い検査パケットでは僅か 12.8Mbps 程度に過ぎず，2.2 で述べたような状況下で突然にパケットのロストが生じる原因となり得ることが確認できた．

6.　おわりに

本研究ではスイッチが期待する性能で機能していることを，各ユーザ・サイトごとに随時検証することを提案し，それを可能にする最初の試作として高い精度で遅延計測が可能で，比較的安価な計測器を開発した．開発した計測器は FPGA を利用するもので，計測精度に関わる処理を

[*4] ASIC 内のパイプライン処理として両機能を結合できないと推測される．

すべて FPGA 内で実行することによって 10Gbps 環境で安定して 8ns 単位の精度での計測が可能である．利用した FPGA ボードは一般的なものであり，SDN スイッチを導入する程度に規模のあるネットワークの管理者が個々に導入可能な価格帯であると考える．

OpenFlow をはじめとする SDN 技術は研究・開発の途上にあり，とくに OpenFlow はその仕様や機能の拡張が続けられるなかで日々挑戦的な実運用環境への応用が進められている．この仕様の策定と実運用環境への適用は SDN 技術を成熟させていくために必要な両輪であり，並進させなければならない．そして今後，OpenFlow2.0 的なアプローチを含めてスイッチハードウェアはさらにプログラマブルになっていくと思われるが，それらは非常に多種の，最小単位ではユーザごとに性能が桁違いに異なる状況を生じさせてしまう．それを把握するための仕組み無しに実利用環境にそうした技術を適用すると，運用においてトラブルが生じ，結果的にそれが OpenFlow あるいは SDN 技術の普及を妨げることにつながりかねない．本研究はそうした問題が大きくなる前に OpenFlow スイッチの性能検証手法を確立し，ネットワーク管理・運用者を支援するツールを通して安定的な実運用環境の維持を可能にすることで SDN 技術の発展に寄与しようとするものである．

今後の課題の一つに測定精度の向上がある．図 4 に示すように，実装した検査装置では，送信されるパケットは 156.25MHz と 125MHz の二つのクロックドメインをまたぐ．各クロックドメイン内部でのパケット生成/解析処理はサイクルレベルで確定的な振舞いをする．しかし，クロックドメインをまたぐための FIFO による同期は，動作時の状態によってレイテンシが異なる場合がある．このことにより，パケットを生成してから送信するまでのレイテンシにゆらぎが発生する．今後精度向上のために，156.25MHz の単一クロックによるシステム構築を検討する．また現段階では運用環境における自動的・継続的なフローエントリ情報の取得については未完成であるため，それらを含めたシステムとして完成させるべく開発を継続する．

謝辞　開発途上の計測器の動作確認を多く手伝ってくれた京都産業大学コンピュータ理工学部（当時）の土屋葵氏に感謝する．

参考文献

[1] C. Rotsos N. Sarrar, S. Uhlig, R. Sherwood, and A. W. Moore : OFLOPS: an open framework for openflow switch evaluation, Proceedings of the 13th international conference on Passive and Active Measurement, Mar. 2012.

[2] M. Kuzniar, P. Peresini, and D. Kostic : What You Need to Know About SDN Flow Tables, Proceedings of the 16th international conference on Passive and Active Network Measurement, Mar. 2015.

[3] A. Curtis, J. Mogul, J. Tourrilhes, P. Yalagandula, P. Sharma, and S. Banerjee : DevoFlow: Scaling Flow Management for High-Performance Networks. Proceedings of the ACM SIGCOMM 2011 conference, Aug. 2011.

[4] D. Y. Huang, K. Yocum, and A. C. Snoeren. : High-fidelity switch models for software-defined network emulation. Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking, Aug. 2013.

[5] N. McKeown : How to tell your plumbing what to do: Protocol Independent Forwarding, Open Networking Foundation Workday (talk), Sep. 2014.

[6] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese, and D. Walker : P4: Programming Protocol-Independent Packet Processors, ACM SIGCOMM Computer Communications Review. Jul. 2014.

[7] OF-PI: A Protocol Independent Layer, Version 1.1, Open Networking Foundation, September 5, 2014.

[8] O. E. Ferkouss, R. Ben Ali, Y. Lemieux, and C. Omar : Performance model for mapping processing tasks to OpenFlow switch resources, 2012 IEEE International Conference on Communications, Jun. 2012.

[9] M. Yu, A. Wundsam, and M. Raju : NOSIX: A Lightweight Portability Layer for the SDN OS, ACM SIGCOMM Computer Communication Review, Apr. 2014.

[10] XPliant Ethernet Product Family, Cavium, http://www.cavium.com/XPliant-Ethernet-Switch-Product-Family.html

[11] R. R. Kompella, K. Levchenko, A. C. Snoeren, and G. Varghese. : Every microsecond counts: Tracking fine-grain latencies with a lossy difference aggregator. Proceedings of the ACM SIGCOMM 2009 conference on Data communication, Aug. 2009.

[12] M. Lee, N. Duffield, and R. R. Kompella. Not all microseconds are equal: Fine-grained per-flow measurements with reference latency interpolation. Proceedings of the ACM SIGCOMM 2010 conference, Aug. 2010.

[13] IEEE. Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems, 2002. IEEE/ANSI 1588 Standard.

[14] RYU SDN Framework, http://osrg.github.io/ryu/