SDN 環境の性能測定に関する一提案

山本 成一[a),b)] 西形 渉[c),a)] 上田 耕佑[a)] 河合 栄治[a),d)]

A study of performance measurement on SDN

Seiichi YAMAMOTO[a),b)], Wataru NISHIGATA[c),a)], Kosuke UEDA[a)], and Eiji KAWAI[a),d)]

あらまし SDN 環境では、ソフトウェアを利用してネットワークの要件定義を実現する。このため、その性能は利用するソフトウェアに依存する。実環境での性能を知るには、ソフトウェアに関する検証の他、実環境で発生する制約も考慮する必要がある。本稿では、実利用に注目した、 SDN 環境の性能評価方法を検討する。

1. SDN 環境の評価

従来のネットワークは、各ベンダが提供するネットワークスイッチやルータで構成されている。これらの機器は IP ルーティングやイーサネットスイッチング等の標準化された機構で動作し、その挙動はネットワーク利用前の個別設定で規程される。一方、SDN(Software Defined Network) 環境では、コントロールプレーンとデータプレーンが分離され、コントローラが中央集権的にネットワーク内の各トラフィックの挙動を制御する。この際にコントローラの挙動は特定のアルゴリズムに制限されず、ネットワーク管理者が作成するソフトウェアにより自由な定義が可能である。

現在、広く利用されている SDN 環境である OpenFlow [1] では、そのコントロール環境（処理系）として ryu [2],trema [3] 等が知られている。これらの処理系は利用者にデータプレーンを抽象化したネットワーク操作環境 (API) を提供している。よって、ネットワーク管理者は普及したプログラミング言語の利用や、多くのサンプルを組み合わせにより、制御アルゴリズムを容易に変更できる。

一般的にソフトウェアの品質や性能は、利用するアルゴリズムが大きく影響する。SDN 環境も同様であるため、適正なアルゴリズムの利用が望まれる。また、SDN を実環境で利用するにあたっては、物理的制約事項との関連も忘れてはならない。

本稿では、コントローラソフトウェアの挙動に注目し、SDN 環境の性能を定量的に求める手法を提案する。

a) 国立研究開発法人情報通信研究機構
National Institute of Information and Communications Technology
b) 東京大学 生産技術研究所
Institute of Industrial science, The University of Tokyo
c) イクシアコミュニケーションズ株式会社
Ixia Communications K.K.
d) 奈良先端科学技術大学院大学
Nara Institute of Science and Technology

2. 提 案 手 法

SDN コントローラの性能測定について先行する手法を示す。

cbench [4] は、OpenFlow スイッチをエミュレーションするソフトウェア実装である。 cbench は測定対象となるコントローラに対し、packet_in メッセージを送付する。コントローラは flow_mod メッセージで応答する。コントロール処理系のベンチマーク測定として、flow_mod を計測することで、スループットやレイテンシを算出する。このようにして、異なるコントロール処理系の性能を定量的に比較する。

OFCBenchmark [5] は OFCBenchmark クライアント (OC) と、 OFCBenchmark コントロールセンター (OCC) で構成されるサーバクライアント型のソフトウェア実装である。OCC からの指示を受け、 OC は接続する仮想スイッチにトラフィックを生成する。これにより、広域な OpenFlow ネットワークにて OC が連携してトラフィック送受信を行い、性能測定を行っている。cbench および OFCBenchmark はソフトウェア実装であるため、実機を用意せずとも評価環境を構築できるというメリットがある。

Ixia 社の IxNetwork [6] は計測用のソフトウェア実装である。IxNetwork は同社のハードウェア計測器を利用して、OpenFlow スイッチのエミュレーション環境を構築できる。IxNetwork は cbench で計測する flow_mod に加え、 packet_out の計測も可能である。また、ハードウェアで動作するため、測定精度が高いというメリットがある。

上記の 3 方式は特定のアルゴリズムで制御される機構を測定対象としており、任意のアルゴリズムを測定対象とすることができない。また、実環境のデータプレーンを利用しないため、実利用から離れた環境となり、実環境の制約事項を見逃す可能性が大きくなるというデメリットがある。

本稿では、1) コントローラで任意のアルゴリズムを利用し、2) 実環境に近い測定を行う事を要求事項とす

る。よって、コントロールプレーンメッセージの計測する手法は採用せず、特定のネットワークフローを生成し、それが意図どおり流れたことを確認する、データプレーントラフィックを観測する手法を提案する。

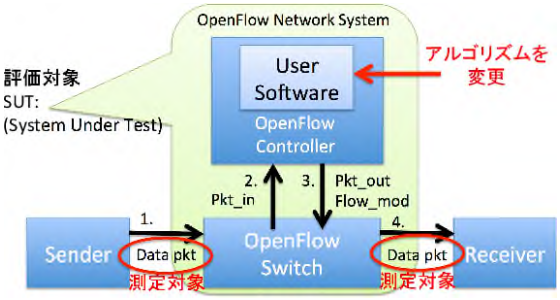

図 1 提案システム概要

3. 評 価

OpenFlow ネットワークシステムでのトラフィック転送にかかる時間をレイテンシとし、一定のフロー数に対するレイテンシを測定した。各フローの初回処理 (packet_in) で一定の遅延を挿入させた。この遅延の変更を、制御アルゴリズムの変更とした。(注1)

遅延挿入なしの場合と、最小の遅延 (30msec) を挿入した場合での、フロー数に対するレイテンシ変化、およびレイテンシ差分の変化を図 2 に示す。レイテンシ差

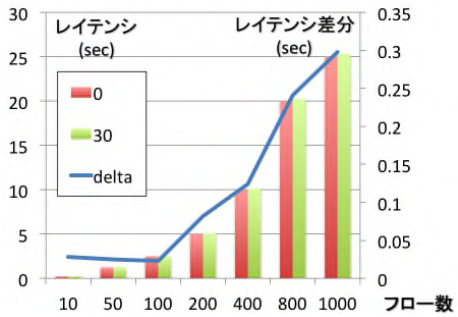

図 2 遅延挿入によるでレイテンシ増加の傾向

分は、フロー数が 100 までの間は挿入分相当 (0.03sec) である。フロー数が 100 を超えるとレイテンシ差分が単調増加となることから、対象システムは一定量の遅延処理がボトルネックとなることが判明した。

次に、挿入遅延時間を変更した場合のレイテンシ変化を図 3 に示す。これより、評価対象システムは、フロー

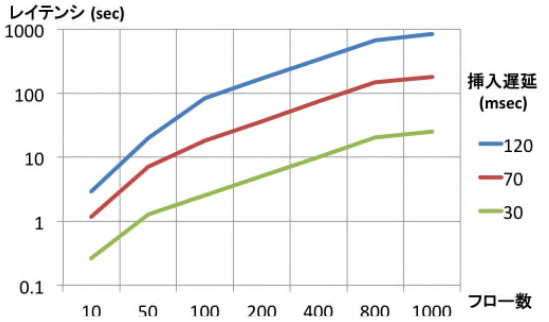

図 3 挿入遅延時間を変更した場合のレイテンシ変化

数 (F) とレイテンシ (L) は、挿入遅延 (D) と定数 (K) を用いて、

$$\log L = D \log F + K_1 \quad (1)$$

$$L = e^{K_1} F^D \quad (2)$$

$$L = K_2 F^D \quad (3)$$

で近似できる特性を有することが判明した。レイテンシがフロー数の冪関数となる理由は、ボトルネックで発生する遅延が累積し、逐次処理されるフローを遅延させている為と考えられる。

4. ま と め

本稿では SDN 環境の性能測定として、データプレーントラフィックを観測する手法を提案し、基礎実験として特定環境下での特性を評価した。今後は複雑な構成の対応や、ハードウェア測定器を活用した精度向上を検討する予定である。

参 考 文 献

[1] McKeown, Nick and Anderson, Tom and Balakrishnan, Hari and Parulkar, Guru and Peterson, Larry and Rexford, Jennifer and Shenker, Scott and Turner, Jonathan. OpenFlow: Enabling Innovation in Campus Networks. *SIGCOMM Comput. Commun. Rev.*, 38(2):69–74, March 2008.

[2] Ryu SDN Framework Community. Ryu SDN Framework. `http://osrg.github.io/ryu/`. Accessed: 2015-09-01.

[3] Yasuhito Takamiya, et al. Trema, Full-Stack Openflow Framework in Ruby and C. `http://trema.github.io/trema/`. Accessed: 2015-09-01.

[4] Amin Tootoonchian, Sergey Gorbunov, Yashar Ganjali, Martin Casado, and Rob Sherwood. On controller performance in software-defined networks. In *USENIX Workshop on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services (Hot-ICE)*, volume 54, 2012.

[5] Michael Jarschel, Frank Lehrieder, Zsolt Magyari, and Rastin Pries. A flexible openflow-controller benchmark. In *Software Defined Networking (EWSDN), 2012 European Workshop on*, pages 48–53. IEEE, 2012.

[6] Ixia. IxNetwork™OpenFlow Solution. `https://www.ixiacom.com/sites/default/files/resources/datasheet/ixnetwork_openflow.pdf`. Accessed: 2015-09-01.

(注1)：評価環境を以下に示す。OpenFlow コントローラ、OpenFlow スイッチ、送信ホスト、受信ホストをそれぞれ OFC,OFS,Sender,Receiver とする。

- OFC サーバ,Sender,Receiver: Aopen XC mini(Intel Mobile CPU Core i5-560M, Mem 8GM, Ubuntu Linux 14.04)
- OFC: ryu 3.23.2 (python 2.7.6)
- OFS: NEC PF5240 (OS-F3PA Ver. V5.1.1.0)

OFC の制御アルゴリズムとして、ryu にサンプルで付属するスイッチ実装 (simple_switch.py) を用いた。遅延挿入は python の"time.sleep()"関数を利用した。Receiver では"ip address" コマンドで macvlan を利用し、異なる mac アドレスと IP アドレスの組を 1000 個用意した。Sender はタイムアウト 10 秒とした fping でデータプレーントラフィックを送信した。このトラフィックでは、宛先ホスト数を変更する事で、フロー数を変化させた。Sender で実行する fping の実行時間を"time" コマンドで測定し、レイテンシとした。