Performance Evaluation of TCP/IP vs. OpenFlow in INET Framework Using OMNeT++, and Implementation of Intelligent Computational Model to Provide Autonomous Behaviour

Pakawat Pupatwibul, University of Technology Sydney, Australia
Ameen Banjar, University of Technology Sydney, Australia
Robin Braun, University of Technology Sydney, Australia

The Asian Conference on Technology, Information and Society
Official Conference Proceedings 2014

Abstract
Analysing performance of transmitting data from a source to a certain destination is an interesting task. One of the most reliable networking protocol suites is the Transport Control Protocol and the Internet Protocol (TCP/IP), which will be studied against a new management paradigm called Software Defined Networking (SDN). SDN is an emerging programmable network architecture, where network control plane is decoupled from forwarding plane. SDN forwarding methods are based on flows, which operate in contrast to conventional routing methods, such as TCP/IP routing table and MAC learning table. Moreover, OpenFlow protocol has efficient forwarding methods to push L2-L4 functions which are simplified into a Flow-Table(s) abstraction. This paper discusses the relationship between the processes of forwarding packets in conventional IP routing table vs. OpenFlow-table and evaluates the performance between both implementations using INET framework in OMNeT++. While TCP performs slightly better than OpenFlow with respect to mean round trip time (RTT). The results also proved the correctness of OpenFlow implemented simulation model. Finally, we propose a Distributed Active Information Model (DAIM) within OpenFlow to support an autonomic network management.

Keywords: OpenFlow, Network Performance, TCP/IP, Software- Defined Networks, OMNeT++.

Introduction

In the last few years network technologies have been improved significantly in performance, complexity, functionality and other aspects, because of current needs and necessities of the modern world. The Internet protocol suite, widely known as TCP/IP, is a networking model and the basic communication language or protocols used to connect hosts on the internet. TCP/IP is the best known protocol suits today because of the successful development of the internet, and thus useful to study the behaviours of this protocol further, by making use of simulations.

The Transmission Control Protocol (TCP) and the Internet Protocol (IP), commonly known as the TCP/IP standard, is widely used for network communications. TCP/IP attempts to make efficient use of the underlying network resources, by specifying how data should be transmitted, formatted, addressed, routed and received at the destination (Forouzan, 2002). TCP/IP was developed to support maximum throughput over many kinds of different networks. Although TCP/IP supports many current network services, it is not efficient for the requirements of business needs and end users. This has led to the development of alternative networking architectures, and the introduction of Software-Defined Networking (SDN).

However, OpenFlow is still not widely standardised yet because each year Open Network Foundation (ONF) has introduced newer versions including improved and extra functionalities. The latest version is known as OpenFlow specification 1.4 (ONF, 2012). The evolution of OpenFlow protocol versions are kept experimenting within labs. This chapter compares the performance of TCP/IP with the newly emerging OpenFlow standard for software defined networking.

A variety SDN approaches have been developed, but there is only limited information on the performance of each, and realistic performance comparisons are not widely available. Simulation tools such as the OMNeT++ INET Framework are suitable for the task of designing, building, and testing network architectures, and provide practical feedback when developing real world systems (Varga, 2001). Such simulation tools allow system designers to determine the correctness and efficiency of a design before a system is deployed. Simulators also enable the evaluation of the effects of various network metrics, and provide mechanisms to obtain results that are not experimentally measurable on larger geographically distributed architectures. However, very few performance evaluations of OpenFlow architectures using simulation tools such as OMNeT++ INET Framework have been published (Varga, 2010 & Varga, 2012).

The performance comparison simulates OpenFlow and TCP/IP networks using the OMNeT++ INET Framework discrete events network simulator. By analyzing key network metrics including round-trip-time (RTT) and data transfer rate (DTR), the results indicate that OpenFlow performed slightly better than TCP/IP in this analysis. The results also proved the correctness of OpenFlow implemented simulation model.

Thus, this paper evaluates the performance of TCP and OpenFlow implementation in INET framework 2.0 using OMNeT++. In addition, the paper covers a number of multiple running from the simulation to preserve accuracy of the simulation results. The remainder of this paper is organized as follows. Sect. 2 describes the motivations of the study. In Sec. 3, we introduced the background and overview of TCP/IP and

OpenFlow networks. We present more details of related works on network simulation and emulation tools in Sect. 4. In Sect. 5, the performance of TCP/IP vs. OpenFlow is evaluated in the simulations. Section 6 proposes the DAIM model within OpenFlow to support an autonomic network management. Finally, Sect. 7 concludes this work.

Motivations

OMNeT++ can be applied to different network scenarios, topologies, behaviours and environments or other application fields for different purposes. Then, it can be studied and analysed to see how the system is functioning and performing. For example, applications of networking simulation area include network traffic, data transfer rate, packet counts and round trip time for packets transmission. OMNeT++ will be the first step for Australia when attempting to implement a new network infrastructure such as OpenFlow. OMNeT ++ is easy to simulate geographic distance and help predict how that would affect the behaviours of this new infrastructure, when considering different technologies or products running on different software. Thus, we have used OMNeT++ modules to design, simulate, verify, and analyse the performance of different networks protocols, where in this context we used TCP/IP and OpenFlow.

OpenFlow can offer network administrators the flexibility to manage, configure and optimise network resources, and thus can effectively control the network behaviour in real-time as well as deploying new network applications and services. OpenFlow-Based SDN can present several substantial benefits including centralised management and control of network devices from various vendors, the direct manipulation of new network services without having to configure each node individually, programmability by administrators, enterprises, users, and software vendor, and the ability to provide centralised and automated management which increases network security and reliability.

Currently, a related work of integrating the OpenFlow protocol version 1.2 in the INET framework for OMNeT++ has been developed. The motivation is to test the correctness of their implemented model in overall compared to TCP modules in INET, and focus especially on the performance of controller's placement based on a variety of performance metrics in the investigated network.

Background of TCP/IP and OpenFlow network

This section introduces Transmission Control Protocol/Internet Protocol (TCP/IP) and provides a quick introduction to OpenFlow-Based SDN, discussing what they are all about in an overall context. We begin by defining both TCP/IP and OpenFlow network in the most general terms.

A. Overview of TCP/IP

The Transmission Control Protocol and Internet Protocol (TCP/IP) is a suite of communication protocols widely used to connect hosts on the Internet and on most other network communications as well (Fall et al., 1996). TCP operates at the transport layer, the middle layer in the seven layers of the OSI model (Jeroen et al., 2004). This layer maintains reliable end-to-end communications between network devices. On the other hand, IP is a network layer protocol, which is responsible for packet forwarding including routing across intermediate routers (see Table 1.).

Because TCP/IP was developed earlier than the OSI 7-layer mode, it does not have 7 layers but only 4 layers.

<table>
<tr><th>TCP/IP Protocol Suite</th><th>TCP/IP</th><th>SDN</th></tr>
<tr><td>FTP, SMTP, Telnet, HTTP, ……</td><td>Application</td><td>Application</td></tr>
<tr><td>TCP, UDP</td><td>Transport</td><td rowspan="3">Control Layer</td></tr>
<tr><td>IP, ARP, ICMP</td><td>Internet</td></tr>
<tr><td rowspan="2">Network Interface</td><td rowspan="2">Network Access</td></tr>
<tr><td>Physical</td></tr>
</table>

Table 1: Comparison of OSI, TCP/IP and OpenFlow models

One fundamental feature of the IP protocol is that it only deals with packets, addresses, and directing messages to where they are intended (Stewart et al., 2001). This is the most significant unit of TCP/IP data transmission. TCP allows two devices to complete a connection and exchange streams of data. TCP assures that the data and packets will be delivered to the destination in the same order in which they were sent.

Like the OSI model, functionalities of TCP/IP has been organised into four abstraction layers. (1): Network Access layer contains the network interface card which provides access to the physical devices. (2): Internet layer establishes network to network communications and therefore connects internetworking. (3): Transport layer handles the end-to-end (host-to-host) communication. (4): Application layer offers the users with the interface to communication and gives a way for applications to have access to networked services.

TCP/IP is a set of protocols developed to allow cooperating computers to share resources across a network and also to ensure network's robustness by recovering automatically from any failure of nodes on the network. Furthermore, it can allow large scaled networks to be contracted with minimal requirements of central management.

B. Overview of OpenFlow-based SDN

Software-defined networking (SDN) is a relatively advanced method for implementing communication networks (McKeown et al., 2008). SDN separates the decision maker, called the control plane, which decides where packets are sent, from the underlying infrastructure, called the data plane, which forwards packets to the decided destination. This is a migration of control can formerly and tightly bound in individual network devices enabling the underlying infrastructure to be abstracted for applications and network services, which can treat the network as a logical or virtual entity. A newly emerging standard for SDN is the OpenFlow standard, which includes a standardized protocol for communications between the control plane and the data plane (ONF White Paper, 2012).

OpenFlow was initially introduced by Stanford University in 2008, as the first standardised communication interface defined between the control plane and the data plane of the SDN architecture (ONF, 2012). OpenFlow is an open standard that enables researchers to run experimental protocols in the networks without having to expose vendors' internal implementations of their network devices. In classical switches and routers, the fast packet forwarding (data plane) and the high-level routing decision (control plane) happen in the same network element. In OpenFlow, it separates these two functions. The data plane portion still resides in the switches, whereas the routing decisions are moved to a different device called the controller, typically a standard server. Figure 1 shows the communication between controller and OpenFlow switch trough a Transport Layer Security (TLS) and its predecessor, Secure Sockets Layer (SSL) channel using the OpenFlow protocol (OpenFlow Switch Consortium, 2009).

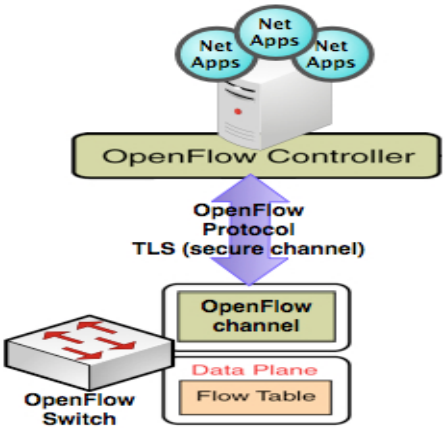


Figure 1: OpenFlow-Based SDN structure

Related works

This paper focuses on evaluating and analyzing networks performance and results, in this regards we use OMNeT++ network simulator. OMNeT++ has many advantages as a network simulator it's been used widely in research academia fields because; it is able to simulate manufacturing, crowd management, airports and weather forecasting. OMNeT++ is scalable as its all simulated modules are implemented as C++ modules and they are linked together as a single process. Moreover, OMNeT++ can modify parameters such as link bandwidth and delay also it is possible to modify configuration of network size, mobility pattern or speed for performance results corrections (Varga & Hornig, 2008). When time is concerned in OMNeT++ the performance results need to be repeated for correction and accuracy. OMNeT++ supports OpenFlow network as an extension of INET framework including spanning tree protocol (STP).

On the other hand, OMNeT++ has many limitations, one of those limitations relate to OpenFlow protocol is that, it is older than the latest version of OpenFlow of this writing is version 1.4 but this paper uses OpenFlow switch specification 1.2. In addition, OMNeT++ uses C++ modules with its simulation engine code as well as all devices and objects as user-level executable program exactly as ns-3 network simulator (NS-3, 2012). Meanwhile, NOX OpenFlow controller operations is a user-level program but, OMNeT++ and NOX cannot be compiled and linked together to

form a single executable program (Gude et al., 2008). For example, it is compulsory to create C++ module from scratch to build OpenFlow switch or controller, even ns-3 needs to build new modules for the same reason. Therefore, the re-implemented modules may not be the same as the behavior of real device/object with real applications, because the re-implemented module is a much-simplified abstraction of the real devices/objects. As OMNeT++ is supporting many functions such as STP, where STP is not supported by ns-3 as well as no TCP connection between simulated hosts, so in real model when use a TCP connection the results are conceders the packet losses and congestion.

There are many other test-beds for observing the network communication such as EstiNet which combines between network simulator and emulator (Wang et al., 2013). However, emulator has limitations as it is only designed for real-time network functional testing. Whereas the simulation is for arbitrary scenarios, the feature that emulator cannot do, so it does not scale very well. The simulated module of OMNeT++ does not connect to real OpenFlow controller as an external entity for measuring the effects of the OpenFlow protocol where Mininet and EstiNet can do. The network emulator separates namespaces such as Mininet, which lead to reducing the overhead of the system rather being as one simulation process. Mininet is emulated hosts as virtualization approach. Mininet emulated hosts can run real application and can exchange information (Lantz et al., 2010). For example, one of these hosts can be a controller because of the controller is as a real application, also can be an OpenFlow switch or just a normal host which can link to other emulated devices using Ethernet pair (linux kernel).

Network emulator has unpredictable results and different experimental results on each run because of the emulated hosts run based on CPU speed, current system activities, system load, memory bandwidth, number of emulated hosts and multiplexing over the CPU. For example, Mininet schedule packets promptly by the operating system, then it is not guarantee that all software OpenFlow switches will forward packets at the same rate of emulated hosts.

Mininet needs to run up a shell process to emulate each hosts and run up a user-space or kernel-space (OpenVswitch) to simulate each OpenFlow switch. Therefore Mininet is less scalable than EstiNet, ns-3 and OMNeT++. Mininet can only be used to study the behavior of the emulated hosts but cannot be used to study time of network/application performance. Mininet GUI can be used for observation purposes such as observing the packet playback of a simulation run and user needs to write Python code to set up and configure the simulation case. However, OMNeT++ has a GUI, which can be used for result observation and users need to write C++ codes to setup and configure the simulations. Overall, it is better to use OMNeT++ even though it takes more time and effort to create simulations, however once modules are created, it's much easier to create new ones.

Performance comparison results

This section provides a performance comparison of TCP/IP vs. OpenFlow modules in INET Framework 2.0 using OMNeT++ network simulation. The network simulator makes it possible to evaluate how a network performs under circumstances with different versions of the protocol stack, and enables analysis of the effects of different variables including channel speed and delay.

A. Measurement methodology

The OMNeT++ INET Framework 2.0 network simulator is a C++ discrete event simulator. An advantage of this simulator is that it simplifies the integration of new modules, and allows existing modules to be customized. The INET Framework is a network simulation package that contains models for wired and wireless networking protocols, including UDP, TCP, SCTP, IP, IPv6, and Ethernet. The INET Framework has recently implemented an extension to enable OpenFlow to be modeled. The OpenFlow extension is still in early development, and is currently based on Switch Specification Version 1.2 (Klein & Jarschel, 2013).

The OMNeT++ network simulator is used here to simulate the operation of OpenFlow and TCP/IP while logging performance metrics including Data Transmission Rate (DTR) and the mean round-trip-time (RTT) for nodes in the simulated networks. We analyse the DTR and RTT for the nodes within TCP/IP and OpenFlow networks, using a similar network topology for each. Each network includes a number of hosts, two switches and a destination server (see Figure 2). The OpenFlow network also includes an additional device called the controller (standard server), which is directly connected via separate links to the OpenFlow switches. Hence the OpenFlow switches can perform Layer 2, 3, and 4 routing, as compared to the Layer 2 MAC learning table used by TCP/IP. The simulations are logged, and the logs are subsequently analysed to enable the performance of the OpenFlow and TCP/IP networks to be compared (Banjar et al., 2014a).

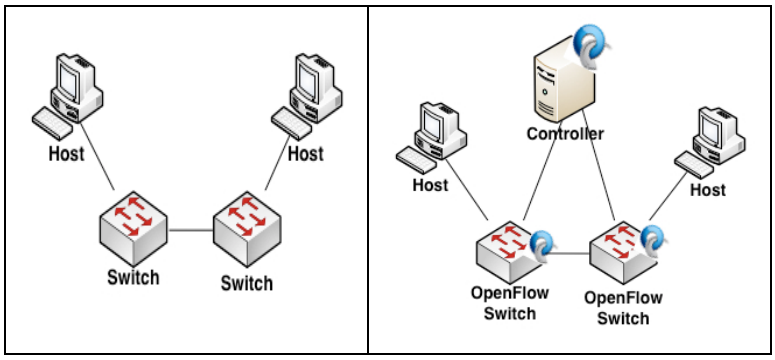


Figure 2: Layout of simulated networks

Performance is compared by varying the traffic and link delays by the same amounts in both the OpenFlow and the TCP/IP simulation channels. In this experiment, we measure the DTR and the mean values of measured RTT between the time-triggered nodes, both for the standard TCP/IP in INET, and OpenFlow. Sequential ping requests are generated, where each includes a sequence number, and replies are expected to arrive in the same order. After the simulation has run for 300 seconds, we measure the RTT for each ping and reply. This results in 48 measurements each for OpenFlow and TCP/IP, for each 300s simulation period. Each simulation was run ten times (OpenFlow and TCP/IP five times each) to reduce simulation artifacts.

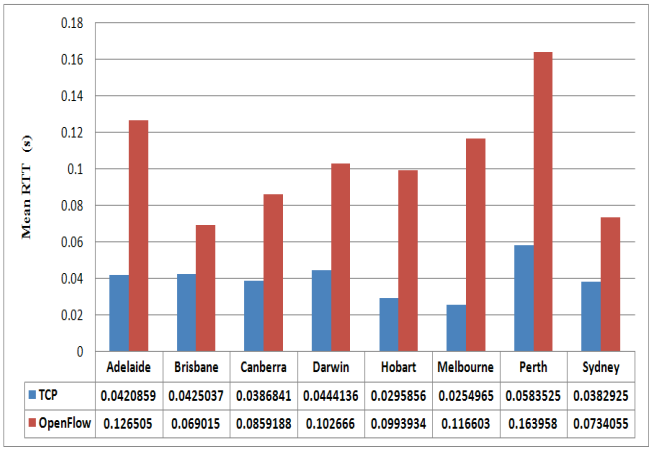
A large number of samples are recorded, and the means and standard deviations of DTR and RTT are computed. RTT is approximated using following equations (Sünnen D., 2011):

$$Tr = \frac{packet\ size}{link\ bandwidth} = \frac{1500 * 8bit}{100 * 10^6 bit/s} = 0.12\text{ms} \quad (1)$$

$$RTT = \alpha * RTT + (1 - \alpha) * Tr \quad (2)$$

The packet size is known to be 1,500 bytes long. The link speed limited to 100 Mbit/s in both the TCP/IP and the OpenFlow networks. ***Tr*** is the transmission time between the segments sent and the acknowledgement arrival, and α is a smoothing factor, which equals the value (7/8) ≈ 0.875.

B. Result evaluation



	Adelaide	Brisbane	Canberra	Darwin	Hobart	Melbourne	Perth	Sydney
TCP	0.0420859	0.0425037	0.0386841	0.0444136	0.0295856	0.0254965	0.0583525	0.0382925
OpenFlow	0.126505	0.069015	0.0859188	0.102666	0.0993934	0.116603	0.163958	0.0734055

Figure 3: Mean RTT TCP vs. OpenFlow

Comparing the OpenFlow simulation with TCP/IP suite allows us to draw conclusions on performance evaluation. It is assessed by varying the same amount of additional traffic and link delay in both OpenFlow and TCP/IP simulation channels. In this experiment, we measure the mean values of measured RTT between the time-triggered nodes, both for the standard TCP and the OpenFlow.

The results in Figure 3 shows the mean RTT from measured scalar values for TCP and OpenFlow among different domains, and it is obvious that the performance of TCP clients in every domain has lower RTT values than OpenFlow when using the ping application. Another outcome of Figure 3 is that OpenFlow performances are affected by the placement of the central controller. For example, the last two domains, Perth and Sydney, where the controller was placed closer to Sydney then the result for this domain is better than Perth with 0.0905 seconds lower rate of the measured mean RTT.

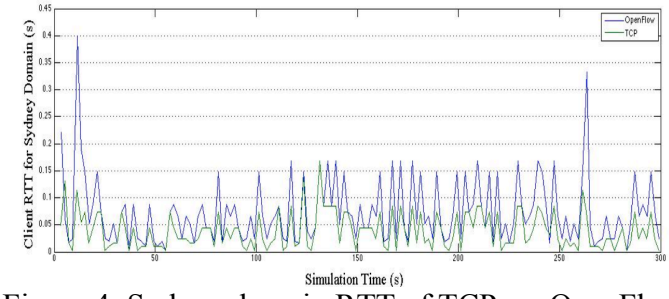


Figure 4: Sydney domain RTT of TCP vs. OpenFlow

Figure 4 and 5 show traces of the round trip times for both TCP and OpenFlow that was used to test the various algorithms. Moreover, they show the comparison in more details to prove the outcome of Sydney and Perth. Where OpenFlow Sydney domain RTT starts at 0.22 seconds, and the OpenFlow Perth domain RTT starts at 0.57 seconds. Thus, these differences are caused by the placement of the central controller. It is also evident that there are sudden spikes at the beginning and at the end of OpenFlow caused by connection establishment and termination. However, the performance of OpenFlow and TCP are slightly similar during 100 to 250 seconds of simulation run time. Because TCP takes less connection set-up time than OpenFlow, it overall performs with lower RTT values, which indicates how well the client can ping to others. This may confirm the assumption that TCP performs slightly better than OpenFlow with respect to mean round trip time (RTT), and performs faster with the same circuitry and not incurring major performance losses.

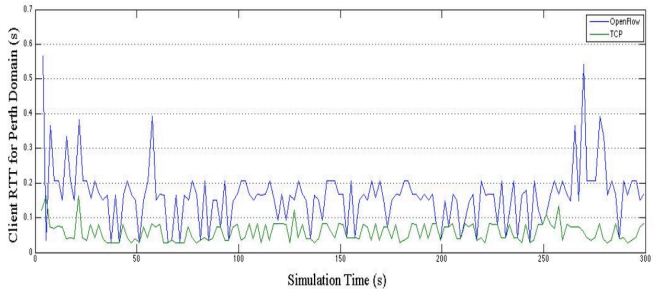


Figure 5: Sydney domain RTT of TCP vs. OpenFlow

Proposed DAIM model and its implementation

DAIM is a sustainable information model, which collects, maintains, updates and synchronizes all the related information. Moreover, the decision making ability within each device locally, on the basis of collected information, allows it to autonomically adapt according to the ever changing circumstances (Banjar et al., 2014b). The DAIM model structure is proposed with the hope that it addresses the limitation of previous network protocols such as Simple Network Management Protocol (SNMP), Common Information Model (CIM) and Policy-Based Network Management.

Proposed DAIM model will address the limitations of current approaches and future distributed network systems, creating an autonomic computing management strategy. The DAIM model approach will also satisfy the requirements of autonomic functionality for distributed network components like self-learning, self-adaptation and self-CHOP (configuration, healing, optimization and security). Each component can be adaptable according to any changed conditions of the dynamic environment without human intervention.

We are proposing that by creating a DAIM model on the networks we could give effect to what we are calling a Reactive Interpreter Network. So it would be a truly distributed computing environment, where these DAIM agents reside in the network elements, which would be OpenFlow switches (Pupatwibul et al., 2013). The actual values in the OpenFlow tables, reside in the OpenFlow switches, and would then be the properties of DAIM agents. These agents would then have to do the work of modifying or adapting their values so as to implement the requirements of the

network. So the whole DAIM model stretches across all these network elements and then could be thought of as reactive distributed interpreter that is interpreting the system requirements to enable the infrastructure to provide for the business needs.

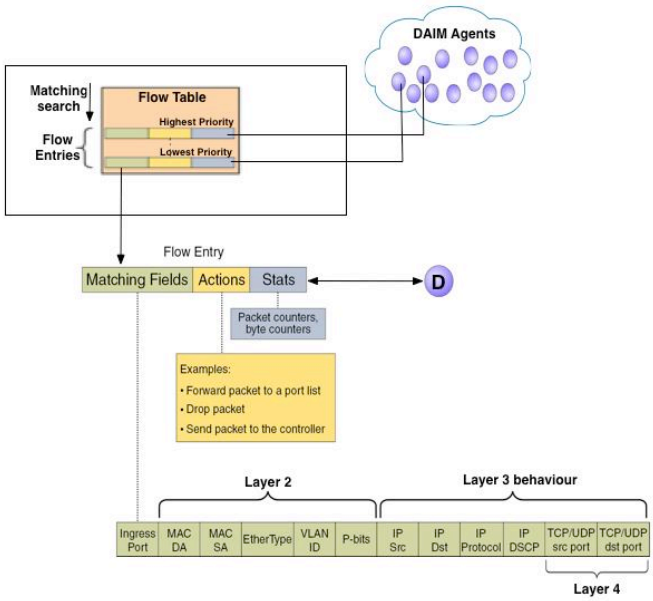


Figure 6: DAIM model integrated in OpenFlow switch

The DAIM cloud has a multi-agent operating system such as SPADE (Smart Python multi-Agent Development Environment) that can create, change, and terminate the intelligent DAIM agents. These agents have the responsibility to maintain their own values, and they can adapt and modify their own value according to the collected information. DAIM agents can make their own local decisions based on the system requirements (see Figure 6).

When the DAIM cloud receives an unmatched packet, it creates DAIM agents which can access and control network elements such as system requirement database, and other switches to determine the forwarding rules. The DAIM agents should be able to check this flow against system requirements and other policies to see whether it should be allowed, and if allowed the DAIM agent needs to compute a path for this flow, and install flow entries on every switch along the chosen path. Finally, the packet itself will be forwarded. The DAIM agents can provide a distributed environment where the network information is the property (values) of software agents residing in virtual machines that are distributed throughout the network elements. Therefore, the DAIM agents have the ingredients to implement autonomic behaviours.

Conclusion

We have introduced TCP/IP and OpenFlow networks including the background, behaviour and architecture. We gave a thorough overview of the related works regarding simulation and emulation tools, which present advantages for researchers to use OMNeT++ over other tools. We also presented Round-Trip-Time as network metric and topologies of our study, followed by evaluating the performance of TCP/IP protocol suite in contrast with OpenFlow protocol. It has been evident according to the measurement outcomes that TCP/IP performs more effective than OpenFlow with lower RTT values and can send streaming UDP packets at a higher rate (Mbps).

Lastly, the proposed DAIM and its implementation have been introduced with the hope to resolve the scalability issues and develop autonomic behaviours in OpenFlow.

As for future studies, we aim to implement the DAIM cloud and extend OpenFlow structure based on intelligent agents to exchange information and install forwarding flow tables, which can be used in other distributed computing environment. Therefore, it could be applied to many different environments such as large data centers and road traffic systems.

Acknowledgement

This work is sponsored by the Centre for Real-Time Information Networks (CRIN) in the Faculty of Engineering & Information Technology at the University of Technology, Sydney (UTS).

References

Banjar, A., Pupatwibul, P., Braun, R., Moulton, B. 2014a, "Analysing the performance of the OpenFlow standard for software-defined networking using the OMNeT++ network simulator," Computer Aided System Engineering (APCASE), 2014 Asia-Pacific Conference, pp.31-37.

Banjar, A., Pupatwibul, P. and Braun, R. 2014b "DAIM: a Mechanism to Distribute Control Functions within OpenFlow Switches." Journal of Networks9.01, pp. 1-9.

Fall K. and Floyd S., 1996 "Simulation-based comparisons of Tahoe, Reno and SACK TCP," ACM SIGCOMM Computer Communication Review, vol. 26, pp. 5-21.

Forouzan B. A., 2002 “TCP/IP protocol suite: McGraw-Hill”, Inc.

Gude N., Koponen T., Pettit J., Pfaff B., Casado M., McKeown N., and Shenker S., 2008 “NOX: towards an operating system for networks”. SIGCOMM Comput. Commun. Rev., 38(3):105–110.

Jeroen I., Heijenk G. and de Boer P. T., 2004 "TCP/IP modelling in OMNeT+.", Univers ity of Twente, The Netherlands.

Klein D. and Jarschel M., 2013 "An OpenFlow Extension for the OMNeT++ INET Framework," OMNeT++ 2013, Cannes France.

Lantz B., et al., 2010 "A network in a laptop: rapid prototyping for software-defined networks," in Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks, p.19.

McKeown N., Anderson T., Balakrishnan H., Parulkar G., Peterson L. Rexford, J., Shenker S. and Turner J., 2008 “OpenFlow: Enabling Innovation in Campus Networks” SIGCOMM Computer Communication Rev., Vol. 38, pp. 69-74.

NS-3 version 3.16. OpenFlow switch support. http://www.nsnam.org/docs/release/3.16/models/html/openflow-switch.html, Dec 2012.

ONF White Paper. Software-Defined Networking: The New Norm for Networks. April 2012.

ONF, 2012, “Open network foundation, openflow switch specification version 1.3.0 (wire protocol 0x04) viewed 20-08-2012 www.opennetworking.org/images/stories/downloads/specification/openflow-spec-v1.3.0.pdf.

OpenFlow Switch Consortium. "OpenFlow Switch Specification Version 1.0.0." (2009).

Pupatwibul, P., Banjar, A. and Braun, R. 2013 "Using DAIM as a reactive interpreter for openflow networks to enable autonomic functionality." Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM. ACM, pp. 523-524.

Stewart R. and Metz C., 2001 "SCTP: new transport protocol for TCP/IP," Internet Computing, IEEE, vol. 5, pp. 64-69.

Sünnen D., 2011 “Performance Evaluation of OpenFlow Switches,” Semester Thesis at the Department of Information Technology and Electrical Engineering.

Varga A. and Hornig R., 2008 "An overview of the OMNeT++ simulation environment," in Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops, p. 60.

Varga A., 2001 "The OMNeT++ discrete event simulation system," in Proceedings of the European Simulation Multiconference (ESM’2001), p. 185.

Varga A., 2010, OMNeT++ user manual. “OMNeT++ Discrete Event Simulation System”. Available at: http://www. omnetpp. org/doc/manual/usman.html.

Varga A., 2012 “INET Framework for the OMNeT++ Discrete Event Simulator”. http://github.com/inet-framework/inet.

Wang, S.-Y., Chou C.-L., and Yang Chun-Ming, 2013 “EstiNet openflow network simulator and emulator”, in Communications Magazine, IEEE. p. 110-117.

Contact email: 10926297@uts.edu.au, 11311103@uts.edu.au