Performance Evaluation of Software-Defined Networking Architectures Using Network Calculus

S. Zerrik, D. El Ouadghiri
IA Laboratory, TAR Team
Faculty of science of Meknes
Meknes, Morocco
{Zerriksafae, dmelouad}@gmail.com

M. Bakhouya
International University of Rabat,
Rabat, Morocco
Mohamed.bakhouya@uir.ac.ma

***Abstract*—Software-defined networking (SDN) are the rising technology in near future to build programmable networks as a way to simplify the evolution of networks. It is a new networking paradigm with the main aim is to decouple forwarding hardware from control decisions. Several SDN architectures have been recently proposed and evaluated. Performance evaluation of these architectures, to study their scalability, was mainly performed using either simulations or experiments. Generally, experimenting with a real system is ex-pensive or impossible. Simulations are extremely slow especially for large systems and provide little insight on how different design parameters affect the actual performance. Therefore, modeling is more effective for evaluating and optimizing SDN architectures prior their implementation. In this paper, we used Network Calculus to study two typical SDN control plane structures, including centralized, hierarchical with 2level. The model describes the behavior of switches and controllers and allows extracting some performance metrics, mainly the delay and backlog.**

Keywords- Software-defined networking, OpenFlow, Scalability of SDN architectures, Analytical performance evaluation, network calculus, backlog, delay.

I. INTRODUCTION

Software-defined networking (SDN) is a new networking paradigm, which aims to decouple forwarding hardware from control decisions. SDN will make networks more flexible, dynamic, and cost-efficient, while greatly simplifying operational complexity. This solution provides several benefits including network and service customizability, configurability, improved opera-ions, and increased performance.

The main components of the SDN architecture are switches, controllers and interfaces as illustrated in Figure 1. Switches are often represented as basic for-warding hardware accessible via an open interface. The controller provides a programmatic interface to the network, where applications can be written to per-form management tasks and offer new functionalities. Thereby, SDN refers to the decoupling of the control plane from the data plane for more agility to handle the large growth rates of network devices and traffic demands. There are three categories of interfaces: southbound, north-bound, and East-West as depicted in Figure 2. The southbound interface defines the control communications between controller platform and data plane devices (including physical and virtual switches). The northbound interface refers to the software interfaces between modules of the controller platform and the SDN applications running on top of the network platform. Those applications can be routing (e.g., topology discovery, traffic engineering), management (e.g., monitoring, maintenance, energy use) and policy (e.g., access control, security). The East-West interface is situated between controllers themselves in order to allow intra domains, inter domains, scalability and interoperability [1].

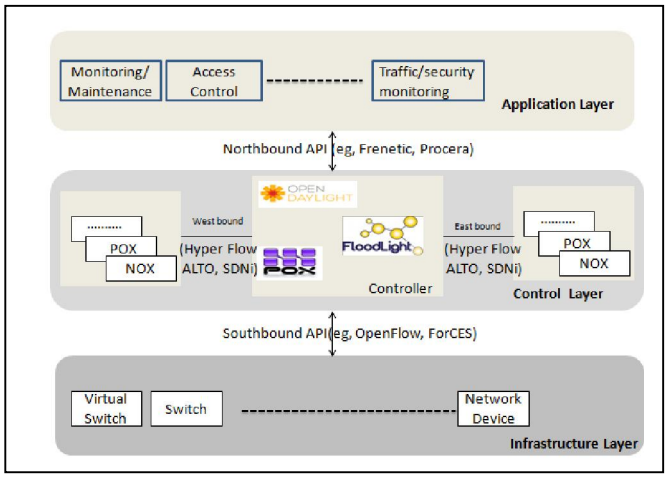


Figure 1. SDN Architecture

In order to allow access to the forwarding plane of network devices, such as switches and routers, a communication protocol, called OpenFlow, is implemented between network infrastructure devices and an SDN controller (southbound API). This protocol uses the concept of flows to identify network traffic based on predefined match rules programmed by the SDN controller [2].

Several architectures have been proposed recently to carry out this solution. Their scalability has been evaluated using either simulations or experiments.

In general, experimenting with a real system is expensive or impossible. Simulations are extremely slow especially for large systems and provide little insight on how different design parameters affect the actual performance. Therefore,

978-1-5090-5146-5/16/$31.00 ©2016 IEEE

modeling is more effective for evaluating and optimizing SDN architectures prior their implementation. Analytical modeling could be used for fast evaluation of performance metrics and quickly analyze the impact of various strategies on the overall system's performance. Time-consuming simulations can only be used later on, when only a few configurations are selected. However, analytical performance evaluation has not been extensively investigated to model SDN architectures and to evaluate main performance metrics. In this paper, we used Network Calculus to describe SDN operations and evaluate some performance metrics, mainly the delay and backlog.

The reminder of this paper is structured as follows. Section 2 present the Network Calculus based modeling, section 3 gives analysis and evaluation, together with obtained results. Section 4 gives some conclusions and perspectives.

II. NETWORK CALCULUS BACKGROUND

In this section, the evaluation results of SDN architectures using Network Calculus framework is presented. We first provide a brief overview of this framework and then its use for modeling and evaluating SDN architectures.

A. Network Calculus Fundamentals

Network Calculus [3] is a modeling framework that allows designers to specify a system as a mathematical model and evaluate main performance bounds such as end-to-end delay. This theory is based on (min; +) algebra for deterministic network performance analysis, especially for worst-case analysis. We consider that any system can be composed of one or several components that exchange traffic in order to accomplish a given task. The traffic pattern of the system can be defined by arrival curves of incoming traffic flows to each component of the system. Let's consider f a data flow characterized by an input function denoted by R(t), which represents the cumulative data units (e.g., packets, bits) of flow arriving at the component C within the time interval [0; t]. Let's consider R*(t) the output function, which represents the cumulative amount of data that leaves the component during the time interval [0; t], R(t) , R*(t).

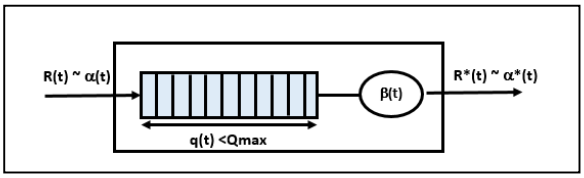


Figure 2. System representation in Network Calculus theory

Furthermore, Network Calculus theory assumes that:

- It exists an arrival curve $\alpha(t)$ that upper bounds R(t) such that $\forall$ s, 0 <s<t, R(t) –R(s) <α(t-s), this means that the amount of traffic that arrives to receive service in any interval [s,t] never exceeds α(t-s).
- It exists a minimum service curve guaranteed to R(t)

The knowledge of the arrival and service curves enables the computation of the delay bound Dmax, which represents the worst-case response time of a message, and the backlog bound Qmax, which is the maximum queue length of the flow

Example of linear arrival curve $\alpha(t) = b + r.t$ that receives a rate-latency service curve $\beta_{R,T}(t) = R.(t-T)^+$, where R>r is the guaranteed bandwidth, T is the maximum latency of the service.

The delay bound D_{max} guaranteed for the data flow with the arrival curve $\alpha(t) = b + r.t$ by the service curve $\beta_{R,T}(t) = R.(t-T)^+$, is computed as follows

$$D_{max} = b/R + T \qquad (1)$$

The backlog bound is expressed as

$$Q_{max} = b + r.T \qquad (2)$$

R is the guaranteed bandwidth, and T is maximum latency of the service, in our analysis, we will use the previous linear arrival curve.

Assume that a flow is constrained by an arrival curve $\alpha(t) = b + r.t$ and FIFO node provides a guaranteed service curve $\beta_{R,T}(t) = R.(t-T)^+$ to the flow. Then, the output bound of the flow is expressed as $\alpha^*(t) = \alpha(t) + r.T$

B. Network calculus model analysis

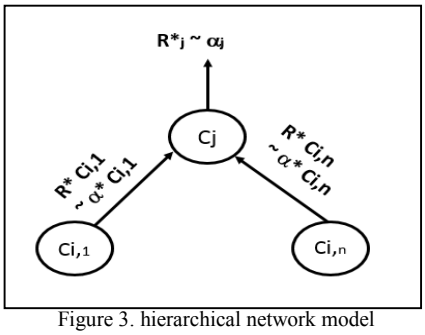


Figure 3. hierarchical network model

As a result, the total input flow of a given parent node j is:

$$\alpha_j(t) = \sum \alpha^*_{ci,n} \qquad (3)$$

Hence, the network flow analysis in SDN based on NC consists in computing iteratively the output flow bound αi using the above equations, the delay is computed node-by-node using equations, end-to-end delay bound is equal to the sum of all hop delay bounds.

C. System model

Like in any tree network, the topology contains a special node called root, which identifies the entire network. In addition, in a tree network, some special devices may have the ability to allow the association from other nodes. End devices with no ability to associate other devices are called local controller (**LC**).

For that purpose, we specify the worst-case topology model by the following two parameters:

- *Max-Depth*: represents the maximum depth of the network, which specifies the maximum number of logical hops for a message from switch to reach the **ROOT**.
- *N-LC*: the maximum number of local controller.

D. Traffic model

We assume that the maximum individual data flow that can be sent by each node (LC, SC) is bounded by the arrival curve $\alpha_N(t) = r_N + b_N.t$ where r_N is the maximum burst size of the N flow, and b_N is its average rate.

For our analysis and evaluation, we select centralized and one distributed SDN control plane, it is a hierarchical architecture, as shown in Figure4. In SDN, there are several types of network requests, which the control plane needs to process, such as network view maintenance, network state update and failure recovery. The flow initiation is the main and basic function of SDN controller [6], which is the focus of this study.

III. ANALYSIS AND EVALUATION

In order to analyze and evaluate the performance of architectures, we need to build a model for the entire system, we assume that this analysis for hybrid SDN architecture,

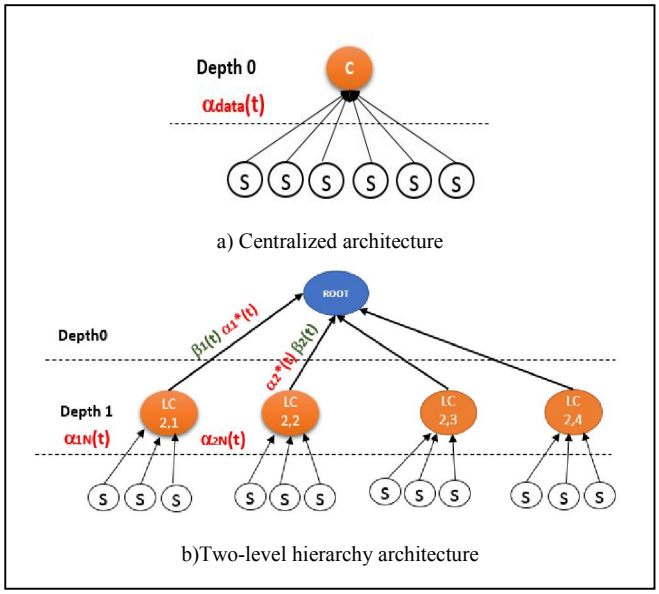


Figure 4. SDN Control plane architecture

A. Analysis of Centralized architecture

In this architecture, there is one controller for the entire network, constrained by the arrival curve α_N, and granted a service curve of the controller is β_N. The arrival curve we have used is a leaky bucket controller which $\alpha_N(t) = r_N + b_N.t$, r_N is the maximum burst size of the N flow, and b_N its average, we assume that average arrival rate of flow requests has Poisson distribution, b_N and r_N are written based on N request received by the controller.

$$\alpha^*_N(t) = r_N + b_N.t \quad (4)$$
$$r_N = N*(N-1)$$
$$b_N = N*(N-1)*\lambda$$
λ factor of Poisson distribution.

B. Analysis of two-level hierarchy architecture

Consider the following queuing system in Figure 5, which explicate more the one in Figure 4 b).

We assume for this architecture, that LC has the view of its local network and manage the data plane directly, and the Root manages each LC.

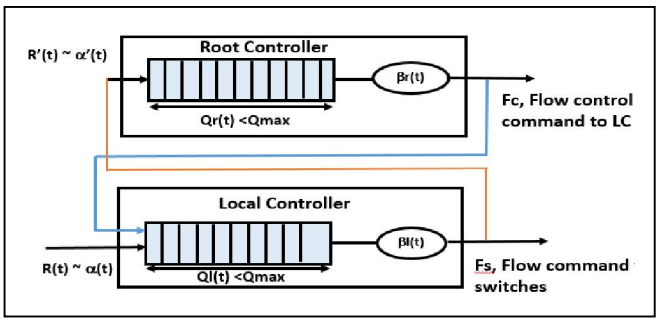


Figure 5. Queuing system model

The output stream of the local SDN controller is divided in two parts. One named global flow is forwarded to the root controller that represent the events, for which the local SDN controller is not able to make a decision. The other, named local flow represents the events, for which an existing action is available in the local controller.

We are assuming that there are m_{LC} local controllers and the average local controller distance is d, and among the N^2–N flows, there are (N^2/m_{LC} –N) flows resolved in LC, and (N^2-N^2/m_{LC}) forwarded to the Root.

Global flow, are processed by the Root controller first, and then will be split into multiple local requests that will be processed by corresponding local controllers.

1) Local controller LC

We assume that there are m_{LC} local controller, let α the overall arrival process of the Local controller, $R(t)$ is upper constrained which

$$\alpha i(t) = ri_N + bi_N.t \quad (5)$$
$$ri_N = (N^2 - N)/ m_{LC}$$
$$bi_N = [(N2 - N) / m_{LC}]*\lambda$$

Fc is upper constrained which

$$\alpha_c(t) = rc_N + bc_N.t \quad (6)$$
$$rc_N = [(N^2 - N^2/m_{LC})/ m_{LC}]*d$$
$$bc_N = [[(N^2 - N^2/m_{LC})/ m_{LC}]*d] * \lambda.$$

According to the multiplexing rule [7], if constrained flows are merged, the output process is also constrained or:

$$A_i \sim (\sigma_i, \rho_i) \rightarrow \sum A_i \sim (\sum \sigma_i, \sum \rho_i)$$

The arrival process of the LC is upper constrained α which

The output stream of local controller Fs Flow command to switch is $\alpha s^*(t)$ upper constrained,

$$\alpha(t) = \alpha i(t) + \alpha_c(t) = r + b.t \quad (7)$$
$$r = [(N^2 - N^2/m_{LC})*(d+1) + (N^2/m_{LC} - N)]/m_{LC}$$
$$b = [[(N^2 - N^2/m_{LC})*(d+1) + (N^2/m_{LC} - N)]/ m_{LC}]*\lambda$$

$\alpha s^*(t)=rs+bs.t+bs.T$ (according to multiplexing rule)
$r_s= [(N^2/m_{LC} –N)+ (N^2- N^2/m_{LC})*d]/ m_{LC}$ (8)
$b_s=[[(N^2/m_{LC} –N)+ (N^2- N^2/m_{LC})*d]/ m_{LC}]*\lambda$

The output stream of local controller forwarded to Root Controller is $\alpha^*_c(t)$ upper constrained,

$\alpha_c^*(t) =r_c+b_c.t +b_c.T$ (9)
$r_c= [N^2- N^2/m_{LC}]/ m_{LC}$
$b_c=([N^2- N^2/m_{LC}]/ m_{LC})*\lambda$

2) Root controller

The total input of Root controller, is the sum of the output flows of its child nodes LC.

$$\alpha_{Root}(t) = m_{LC}.\alpha^*_c(t) \quad (10)$$

α^*_{Root} the output stream of Root Controller

$$\alpha^*_{Root}(t) = \alpha_{Root}(t)+ b_{root}.T \quad (11)$$

C. Computaion of Delay bound and Backlog

1) Centralized architecture

We propose to compute the maximum delay bound of flows, which is the delay, bound of a data flow sent by a controller to data plane.

We calculate delay and backlog of controller using the formulas (1), (2) and equation (4).

$$D_{max}=\frac{N\times(N-1)}{R}+T$$

$$Q_{max}=N\times(N-1)\times(1+\lambda.T)$$

2) Tow-level hierarchy architecture

We calculate the maximum delay bound of local and global flows, which is the delay, bound of a data flow sent by a controller to data plane.

a) Depth =1

Delay bound of local flow, and backlog of LC are calculted using the formulas (1), (2) and equation (8), and are expressed as follow

$$Dl_{max}=\frac{rlocalflow}{Rflow}+T$$
$$Ql_{max}= r_{localflow}+ b_{localflow}.T$$

Which $r_{localflow}= rs+ bs.T$, $b_{localflow} =b_s$

b) Depth=0

Delay bound of global flow, and backlog of the Root, are calculted using the formulas (1), (2) and equations (9), (10) (11), and are expressed as follow

$$Dg_{max}=\frac{rglobalflow}{Rflow}+T$$
$$Qg_{max}= r_{globalflow}+ b_{globalflow}.T$$

$r_{globalflow} = r_{root} + b_{root}.T$, $r_{root}=m_{LC}*r_c$ $b_{root}=m_{LC}*b_c$

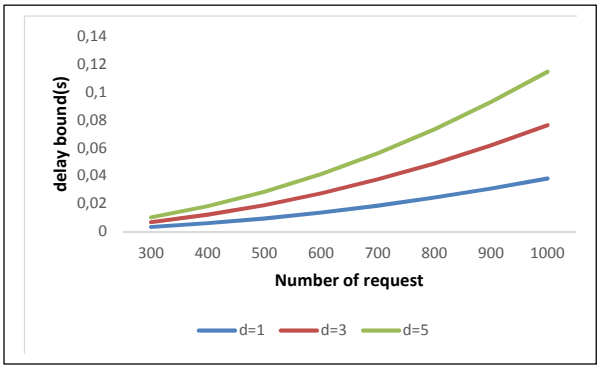
$b_{globalflow}= b_{root}$

End-to-end delay bound is equal to the sum of all hop delay bounds, which

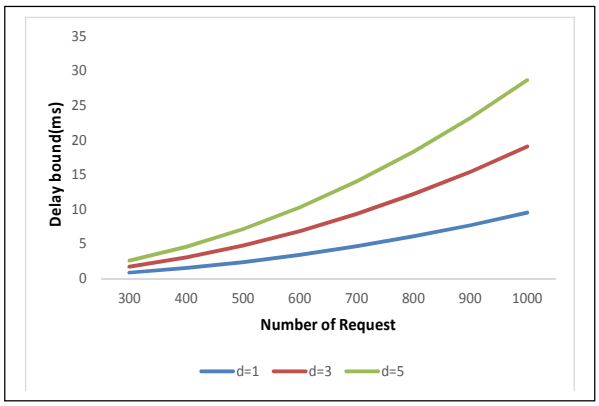
$D_{max}= Dl_{max+} Dg_{max.}$

D. Numerical evaluation

We evaluate and compare the delay bound in centralized and hierarchical architecture with two level. N ranges from 500 to 1000, λ=0.001 per/s [5], R=1.45Mreq/s [6], T=Sf/R, Sf is the flit size, for this analysis Sf= 8 bytes.



a) m_{LC}=6



b) m_{LC}=12

Figure7. delay bound of local flow with different values of d

In figure 7 a) and b), we compare delay bound of local flows with different values of d, we can see that delay bound becomes worse as d increases, even if number of controller increase, we remark that d affect the delay.

In figure 8 a) and b), we compare delay bound of both centralized architecture and global flows of hierarchical architecture, we remark that centralized architecture present the worst case, but when d=5, the delay bound of centralized and global flows are very close.

We note that delay bound decreases with hierarchical architecture, then we can address the problem of scalability by designing hierarchical structure, but we remark as d increases, the delay increase, in other words as the number of controller to resolve a query increases, it affect the delay,

then, during the design, we must take into consideration factor d.

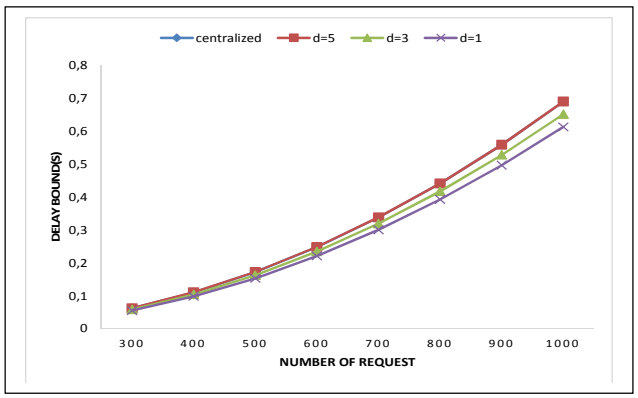


Figure8. delay bound of local flow with different values of d

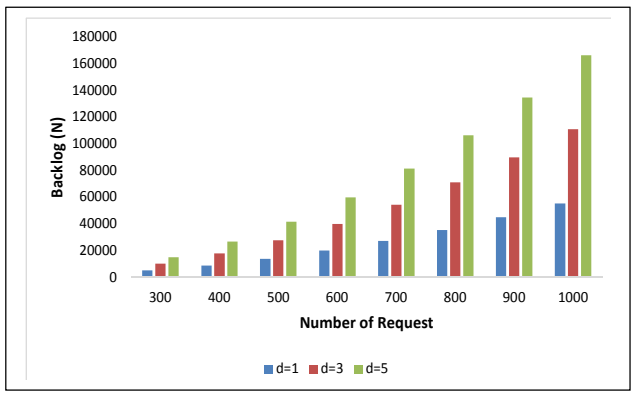


Figure 9. Bakclog of local controller with different values of d

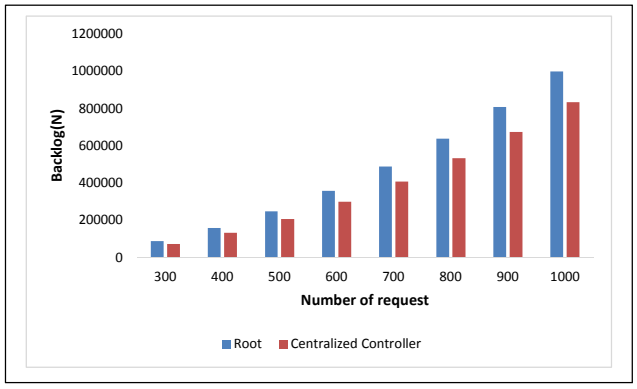


Figure 10. Backlog of Root and centralize controller

Figure 9) shows the backlog of local controllers of 2-level hierarchy, with different values of d, d=1 present the best performance, in figure 10) Centralized and Root controller presents the worst case.

IV. CONCLUSION AND PERSPECTIVES

In this paper, we provided an introduction of Software-Defined Networking (SDN) and OpenFlow. The scalability is one of most important issues since the introduction of SDN, designing a scalable SDN control plane becomes a critical problem, the flow initiation in controller introduces a flow setup delay and may limit scalability. We build a mathematical model using network calculus to quantify the delay bound of flow initiation and backlog of controller. We analyze the delay in different structures in which hosts have the similar traffic pattern. Two typical structures of control plane, namely centralized, hierarchical structures are studied and analyzed with detailed calculation, and some useful conclusions have been proposed to compare these structures.

REFERENCES

[1] Myung-ki shin, ki-Hyuk Nam, Hyoung-Jun Kim “Software Defined Networking (SDN): A Reference Architecture and Open APIs”, 2012 IEEE International Conference on ICT Convergence (ICTC), pp. 360-361.

[2] Kapil Bakshi “Considerations for Software Defined Networking (SDN): Approaches and Use Cases”, IEEE Aerospace Conference, 2013, pp. 1-9.

[3] J.-Y. L. Boudec, P. Thiran, Book, LNCS 2050 (2001) “Network calculus: A theory of deterministic queuing systems for the internet”.

[4] Soheil Hassas Yaganeh, Amin Tootoonchian, Yashar Ganjali “On the Scalability of Software-Defined Networking”, IEEE Communications Magazine Feb 2013.

[5] Jie Hu, Chuang Lin, Xiangyang Li, Jiwei Huang. (2014). “Scalability of Control Planes for Software Defined Networks:Modeling and Evaluation”. 2014 IEEE 22nd International Symposium of Quality of Service (IWQoS), 2014.

[6] Tootoonchian, A., Gorbunov, S., Ganjali, Y., Casado, M., & Sherwood, R. “On controller performance in software-defined networks”. Proceeding Hot-ICE’12 Proceedings of the 2nd USENIX Conference on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services,2012

[7] Siamak A., Philipp W. and Ramin Y. “Performance Evaluation of a Scalable Software-Defined Networking Deployment”, 2013 Second European Workshop on Software Defined Networks, IEEE Xplore 2013, pp.68-74.