doi:10.21311/001.39.3.27

A Framework of Controller with Flow Table Cache and Performance Analysis in Software Defined Industrial Networks

Dong Li[1, 2]

[1]*University of Chinese Academy of Sciences, Beijing, 100049, China*
[2]*Shenyang Institute of Automation, Chinese Academy of Sciences, Shenyang, Liaoning 110016, China*

Peng Zeng, Ming Yang, Xueting Yu, Haibin Yu

Shenyang Institute of Automation, Chinese Academy of Sciences, Shenyang, Liaoning 110016, China

Abstract

The industrial internet and industrial 4.0 introduce the new requirements of industrial networks to be more flexible, manageable and scalable. Software Defined Network (SDN) separates control plane and data plane, abstracts the complexity of the network, provides faster network services and more flexible network management, and bring great impact to the traditional industrial networks. However, due to the limit of power and cost in industrial network switches, the scale of flow table of these switches is smaller than the scale of switches in datacenter, which leads to a low flow table hit ratio and heavy load in SDN controller. In this paper, we propose a framework of SDN controller with flow table cache and increase the hit ratio. Based on the framework, a model of SDN controller is proposed and the performance of SDN controller can be analyzed. The simulation shows the analysis results from the model is close to the simulation and the controller with flow table cache is more efficient in software defined industrial networks.

Key words: Software Defined Network, Industrial Network, Flow Table.

1. INTRODUCTION

Software Defined Network (SDN) separates control plane and data plane, abstracts the complexity of the network, provides faster network services and more flexible network management, and bring great impact to the traditional network. Software Defined Network (SDN) and OpenFlow technology are paid attention as one of the hot spots by industry. The essence of Software defined networks is not only in the separated control plane and data forwarding plane, the open southbound and northbound interfaces, but also the centralized control and manage capability built on network abstraction layer. Moreover, its powerful capability dependent on the virtualization technology, which is promoted by computing virtualization and storage virtualization from the development and implementation mechanism. Network virtualization, computing virtualization and storage virtualization coordinated each other closely.

In recent years, in response to resource and environment challenges and the requirements of personalized products, Europe and the U.S.A. have proposed the development of new manufacturing mode including the "Industry 4.0" and “industry internet”. Such manufacturing mode includes a shift of basic mode from centralized control to enhanced decentralized control, and the goal is to create a highly flexible, personalized and digital production mode of products and services. In this mode, the traditional industry boundaries will disappear, and will produce a variety of new activities areas and cooperation forms. The process of creating new value chain is changing, and traditional industrial chain division will be reorganized. In the new manufacturing model, the traditional closed and fixed industrial control network system will be broken, and new industrial control network architecture will be built.

As the basis of industrial automatic control system, the industrial control network is developing quickly driven by the information technology since its emergence(Zeng and Yu, 2002). The development of industrial control systems is from simple signal feedback control, computer control technology into the modern network control technologies based on information network, fieldbus and industrial Ethernet technologies(Zeng and Yu, 2004). With the proposes of new architectures of future industries such as intelligent manufacturing, industrial internet and Industry 4.0, the future industrial control networks need to meet the requirements of high-performance and highly intelligent. The new requirements of industrial networks are to be more flexible, manageable and scalable. Industrial control networks based on software-defined can well meet the above requirements and resolve widespread problems in current industrial control network, which we call Software Defined Industrial Networks.

One of the most possible way to build software defined industrial networks is Openflow. The OpenFlow specification defines a protocol between the controller and the switches and a set of operations on the switches.

The operations are defined by flow table. Each switch maintains a number of flow tables, with each table containing a list of flow entries. Each flow entry contains a match field that defines the flow, a counter and a set of instructions. Entries in the match field contain either a specific value against which the corresponding parameter in the incoming packet is compared or a value indicating that the entry is not included in the parameter set of this flows.

A Software Defined Industrial Network is quite different from the commercial one due to the deterministic real-time requirement of delivering messages from the industrial applications. In this way, some fixed flow table must be reserved for real-time data flows. On the other hand, due to the limitations of industrial production conditions, the size, specifications and power consumptions of software defined industrial network switches are strictly restricted. The scale of flow tables of industrial network switches is smaller than the scale of switches in datacenter. Thus, the scale of flow table left for unreal-time applications is rather small, which leads to a low flow table hit ratio and heavy load in SDN controller.

In this paper, a framework of software defined industrial network controller with flow table cache is proposed, and a mathematical model is built to analysis the performance. The mathematical model is based on the theoretical basis of queuing theory, use Jackson queuing network as a basis model, and is modified and improved according to the characteristics of the SDN network so that the mathematical model can meet the requirements of SDN network characteristics. The rest of the paper is organized as follows. Section 2 introduces the related work of modeling SDN controller. Section 3 formulates our models to SDN controller and analysis the performance of SDN controller with and without flow table cache. In section 4 we give out the evaluation of our model and section 5 concludes with a summary of main research results.

2. RELATED WORK

The analytical modeling of OpenFlow-based SDN networks has only been attempted in a handful of papers before. Feedback oriented queuing theory has been used to capture the control plane and data plane interaction in the Markovian servers' environment (Jarschel and Oechsner, 2011). This is the first research on SDN network attempting to capture the interaction between the controller and the switch. However, it is not clear how the model can be extended to more than one switch in the data plane.

In previous work (Ciucu and Schmitt, 2012), a network calculus based approach is used to quantify the packet processing capability of the switch in the data plane. However, the feedback between the nodes in the data plane and the controller is not considered. This shortcoming of feedback modeling is addressed (Naous and Erickson, 2008). However, the model is depicted only for a single node in the data plane and the time stopping method employed therein has limited real time application. Secondly the framework is based on deterministic network calculus which does not provide any meaningful bounds (Bianco and Birke, 2010).

At present, almost all the studies of analyzing and evaluating performance of OpenFlow-based SDN networks are carried out by simulations or measurements. The performance of OpenFlow switch implemented on NetFPGA platform is evaluated in the study (Khan and Dave, 2013), but it is limited by a single switch and the implementation platform. OpenFlow switches are deployed on the Linux platform in study (Sherwood and kok-kiong, 2013), and performance evaluation are extended to L2 and L3 routing. A modular and parameterized hardware-based OpenFlow switch is proposed in study (Jackson, 2005), which can be stably deployed on three different platforms—NetFGPA, ML605 and DE4. Properties can be compared against the three different deployments. The results show that OpenFlow-based SDN switch can be deployed on multiple platforms and performance differences between different platforms is unlikely. Although performance evaluation on OpenFlow protocol based SDN switch have be made by similar studies, but depth unified frame of performance evaluation on OpenFlow-based SDN network is not be proposed. Moreover, the performance evaluation in most of the above research is for a single network node, do not take the entire network topology into account. In addition, there are not a good way to evaluate the performance of SDN network controller on the network level.

3. THE FRAMEWORK OF CONTROLLER AND MODELING

In OpenFlow network environment, the communication between the controller and the switch is in accordance with the following rules. When a flow with no specified forwarding instructions comes into a network, the following actions are taken:

1. A packet (or part of the packet) of the flow is sent by the switch to the controller, assuming that the switch is not configured to drop unknown packets.
2. The controller computes the forwarding path and updates the required nodes in the data path by sending entries to be added to the flow tables.
3. All subsequent packets of the flow are forwarded based on pre-calculated forwarding decisions.

In software defined industrial networks, since the scale of flow table in switches is limit, we build a flow table cache in controller to accelerate the computing process. The SDN controller is usually run on a powerful server, the flow table cache can be much larger than the flow table in a switch. When a packet is sent to the

controller, the controller first looks up in the cache. Unless the packet misses the flow table cache, the calculation will be done and a new entry will be added into the flow table cache. That is the framework of SDN controller with flow table cache. Since the looking up is much faster than computing the path, a SDN controller with flow table cache can have a better performance result than one without flow table cache. Then, we try to model the controller and give out the numerical results of the comparing.

For OpenFlow-based SDN, especially industrial SDN, the key of network performance analysis is the communication between the controller and the switches. The modeling of OpenFlow networks will play a crucial role on studying data carrying capacity of industrial control network, the real-time requirements in control network, as well as performance bottlenecks of controllers and switches in industrial control software-defined network.

It is important to have an analytical model to analysis performance of software definition industrial control network. It can capture the feedback interaction between the controller and the switch, is able to model any amount of traffic going from switch to controller (and vice versa), can be easily extended to more than one switch in the data plane, and can reflect the features of applying SDN in industrial control network. In this study, Jackson network model based on M / M / c queue in queuing theory is used. The SDN and industrial control network-specific properties are added into modeling, which have theoretical and practical value. The mathematical model can evaluate the performance of OpenFlow-based industrial software definition network, and then calculate the average amount of time that data packets in network and the traffic load of the controller.

We assume that the overall traffic process at the switch and at the controller follows Poison process. The data transferring time between the switch and the controller is included into the controller service time. Also, we assume that the buffer size for switches and controllers is infinite.

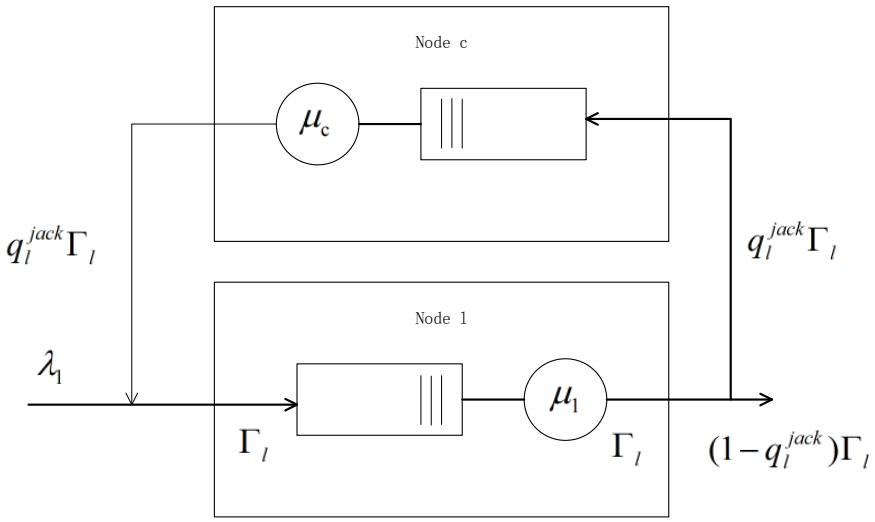

Figure 1. TheJackson Network Model

We use the Jackson network model as the basic mathematical model for OF-based industrial SDN network, as shown in figure 1, which is classic open queuing network based on M/M/1 queue.

In queuing theory, a queuing model is a mathematical approximate simulation of the actual queuing system. Queuing network system is composed of several a single queuing system. Customers from different systems or in different types access and go through the network in different line or ways, and are serviced at service station (node). The different services are provided by each service station, which can be a real service or only through it. For example, a customer access to the bank, some people go directly to the cashier to withdraw money or save money, some people need to start with the receptionist service, and then the cashier. If the amount of money is huge, people need manager's service. So, receptionist, cashier, manager and automatic storage machine are nodes, and the entire banking system is a queuing network. If we simulate the Three Gorges ship lock system, the locks of different levels opened and closed, and the different routes the vessels going through is a queuing network system. Many factories have to face device maintenance, accessories supply and order, production line management, etc. which can constitute queuing network system using queuing network model to simulate the system implementation. In the queuing network system, customers can enter the system from one queue to another, such transfers can be random, subject to a probability distribution, or be fixed. The SDN network packets are seen as a customer waiting to be served, the switch and the controller are considered as service points. When data packet from the data source enters the network, it will wait at the switch for forwarding "service". If the switch cannot match the flow table, the packet will enter controller service point, wait in queue for the "services" that controller calculates routes and flow table. Thus, the communication

between switches and controllers of SDN network to can be established with the mathematical model of queuing network modeling, as well as analysis and performance evaluation.

A queuing network system is defined as the Jackson network, then it is satisfied:

1, All visitors beyond the system, regardless of which service station first visiting, all submit to Poisson distribution;

2, whichever service station, all service times are exponentially distributed;

3, All service stations, the reception number of customers can be unlimited;

4, When a customer service is completed, the probability of transfer to another service is nothing to do with the service process that the customer has accepted and the service station where other services in.

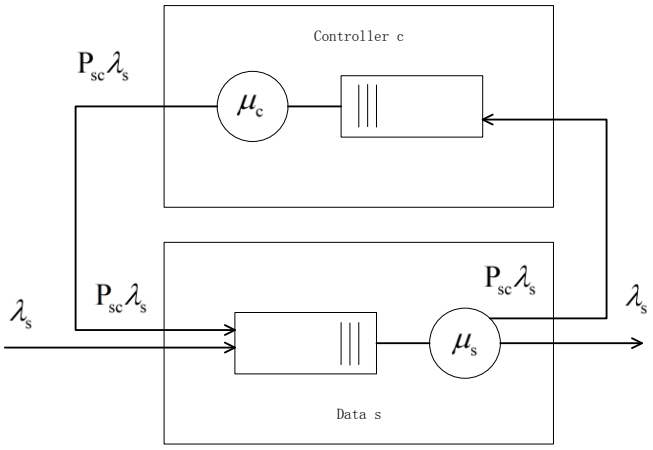

Figure 2. Typical SDN Model

3.1 Software Defined Industrial Network Model

Typical Jackson queuing network system as shown in Figure 1, the service rate of node $c$and node l connected through a feedback path follows exponential distribution, and the average value are μ_c and μ_l ,respectively . The external traffic arrival to the node l is denoted as$\lambda_l$packets per unit time. Let Γ_l be the number of input packets of node l. Define $\Gamma_c = q_l^{jack}\,\Gamma_l$, which indicates the number of packet goes to the node c.$q_l^{jac\,k}$is the probability that the packet goes to the node c. Assuming that no packet is lost in the system, the buffer of the system is infinite, the balance equation for the system can be written as:

$$\Gamma_l = \lambda_l + q_l^{jack}\,\Gamma_l \tag{1}$$

In Figure 2, we define $P_{sc}$is the probability that a packet forwarded from the node$c$ i to the node s.The external traffic arrival to the node s is denoted as$\lambda_s$packets per unit time. Let $\Gamma_s$and $\Gamma_c$be the number of input packets of node s and***Controller***. Thus, the following equation can be derived:

$$\Gamma_s = \lambda_s + P_{sc}\lambda_s \tag{2}$$

$$\Gamma_c = P_{sc}\lambda_s \tag{3}$$

The significance of a mathematical model is that, researchers can assess network performance better through the network mathematical model, as well as better network planning and construction in practice. Through mathematical model of industrial SDN network, it is possible to obtain two basic performance metrics of networks - the average residence time of the data packet and a data packet residence time distribution in the network. Let T_s^c represent the communicating time between node s and the controller, which shows the time interval of the data packet transmitting from the data source to the data destination. It is possible to obtain:

$$T_s^c = \begin{cases} T_s & 1 - P_{sc} \\ T_s + T_s + T_s^{(2)} & P_{sc} \end{cases} \tag{4}$$

In the formula above, $T_s^{(2)}$ represents the time that a packet enters the controller c at the second time. Therefore, the average packet sojourn time can be obtained by the above formula.

$$E[T_s^c] = E[T_s](1 - P_{sc}) + (E[T_s] + E[T_c] + E[T_s^{(2)}])P_{sc} \tag{5}$$

Assuming N_i and T_i represent the packets number and packet sojourn time in node i respectively, then according to the classic M/M/1 queue properties in queuing theory, we have:

$$E[T_i] = \frac{1}{\mu_i - \Gamma_i} \tag{6}$$

$$E[N_i] = \frac{\rho_i}{1-\rho_i} \tag{7}$$

In the formula above,ρ_i represents the load of node i. Thus, formula (5) can be rewritten as:

$$E[T_s^c] = \frac{1}{\mu_s - \Gamma_s}(1 + P_{sc}) + \frac{1}{\mu_c - \Gamma_c} P_{sc} \tag{8}$$

According to the above formula,$\Gamma_s = (1 + P_{sc})\lambda_s$,$\Gamma_c = P_{sc}\lambda_s$can be obtained. Therefore, the above equation (7) is converted to:

$$E[T_s^c] = \frac{1}{\lambda_s}\frac{\rho_s}{1-\rho_s} + \frac{1}{\lambda_s}\frac{\rho_c}{1-\rho_c} \tag{9}$$

It is assumed the sojourn time of packet in switches and controllers is independent, then for node i ,Laplace transform$W_i^c(s) = E[e^{-sT_i^c}]$ may be written as:

$$W_i^c(s) = (1 - P_{ic})\frac{a_i}{a_i + s} + P_{ic}(\frac{a_i}{a_i + s})^2(\frac{a_c}{a_c + s}) \tag{10}$$

In the above formula,$a_i = \mu_i - \Gamma_i$,$a_c = \mu_c - \Gamma_c$.

Thus, formula(10) can futher be written as:

$$W_i^c(s) = b_i^{(1)}\frac{a_i}{a_i + s} + b_i^{(2)}(\frac{a_i}{a_i + s})^2 + d_i\frac{a_c}{a_c + s} \tag{11}$$

According to Laplace inverse transform, the probability density function $w_i^c(t)$ and probability distribution function $\widetilde{W}_i^c(t)$ of the time spent in the network by a packet in the node I can be obtained.

$$w_i^c(t) = b_i^{(1)}a_i e^{-a_i t} + b_i^{(2)}a_i(a_i t)e^{-a_i t} + d_i a_c e^{-a_c t} \tag{12}$$

$$\widetilde{W}_i^c(t) = P(T_s^c > t) = (b_i^{(1)} + b_i^{(2)})e^{-a_i t} + b_i^{(2)}(a_i t)e^{-a_i t} + d_i e^{-a_c t} \tag{13}$$

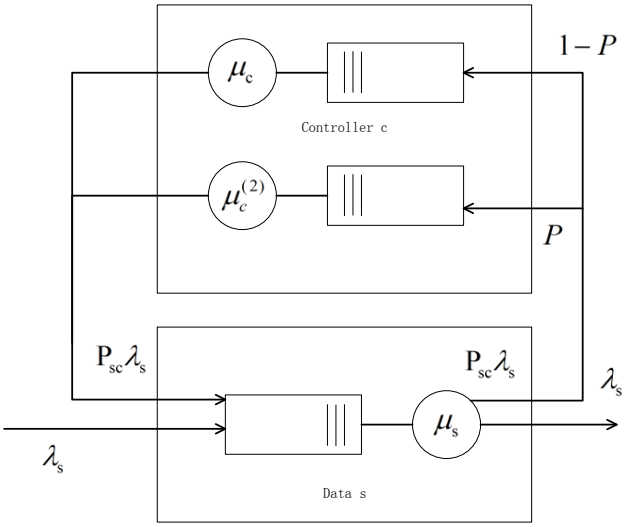

Figure 3. SDN Model with flow table cache in controller

3.2 SDN model with flow table cache in controller

T_s^crepresents the communicating time between node s and the controller c, which showing the time interval of the data packet transmitting from the data source to the data destination. So, it is possible to obtain:

$$T_s^c = \begin{cases} T_s, & 1 - P_{sc} \\ T_s + T_c^{(2)} + T_s^{(2)}, & P_{sc}(1-P) \\ T_s + T_c^{(2)} + T_s^{(2)}, & P_{sc}P \end{cases} \tag{14}$$

In the above formula, $T_s^{(2)}$represents the time that a packet enters the node s at the second time, $T_c^{(2)}$ represents the time that a packet matches flowtable cache in controller. Therefore, the average packet sojourn time can be obtained by the above formula.

$$E[T_s^c] = E[T_s](1 - P_{sc}) + \left(E[T_s] + E[T_c] + E\left[T_s^{(2)}\right]\right)P_{sc}(1 - P) + \left(E[T_s] + E\left[T_c^{(2)}\right] + E\left[T_s^{(2)}\right]\right)P_{sc}P \quad (15)$$

Assuming N_i and T_i represent the packets number and packet sojourn time in node i respectively, then according to the classic M/M/1 queue properties in queuing theory, we have:

$$E[T_i] = \frac{1}{\mu_i - \Gamma_i} \quad (16)$$

$$E[N_i] = \frac{\rho_i}{1 - \rho_i} \quad (17)$$

In the formula above, ρ_i represents the load of node i. Thus, formula (15) can be rewritten as:

$$E[T_s^c] = \frac{1}{\mu_s - \Gamma_s}(1 + P_{sc}) + \frac{1}{\mu_c - \Gamma_c}P_{sc}(1 - P) + \frac{1}{\mu_c^{(2)} - \Gamma_c^{(2)}}P_{sc}P \quad (18)$$

According to the above formula, $\Gamma_s = (1 + P_{sc})\lambda_s$, $\Gamma_c = P_{sc}\lambda_s$ can be obtained. Therefore, the above equation (18) is converted to:

$$E[T_s^c] = \frac{1}{\lambda_s}\frac{\rho_s}{1 - \rho_s} + \frac{1 - P}{\lambda_s}\frac{\rho_c}{1 - \rho_c} + \frac{P}{\lambda_s}\frac{\rho_c^{(2)}}{1 - \rho_c^{(2)}} \quad (19)$$

It is assumed the sojourn time of packet in switches and controllers is independent, then for node i ,Laplace transform $W_i^c(s) = E[e^{-sT_i^c}]$ may be written as:

$$W_i^c(s) = (1 - P_{ic})\frac{a_i}{a_i + s} + P_{ic}(1 - P)\left(\frac{a_i}{a_i + s}\right)^2\left(\frac{a_c}{a_c + s}\right) + P_{ic}P\left(\frac{a_i}{a_i + s}\right)^2\left(\frac{a_c^{(2)}}{a_c^{(2)} + s}\right) \quad (20)$$

In the above formula, $a_i = \mu_i - \Gamma_i$, $a_c = \mu_c - \Gamma_c$, $a_c^{(2)} = \mu_c^{(2)} - \Gamma_c$.
So formula(20) can futher be written as:

$$W_i^c(s) = b_i^{(1)}\frac{a_i}{a_i + s} + b_i^{(2)}\left(\frac{a_i}{a_i + s}\right)^2 + d_i\frac{a_c}{a_c + s} + d_i^{(2)}\left(\frac{a_c^{(2)}}{a_c^{(2)} + s}\right) \quad (21)$$

According to Laplace inverse transform, the probability density function $w_i^c(t)$ and probability distribution function $\widetilde{W}_i^c(t)$ of the time spent in the network by a packet in the node i can be obtained.

$$w_i^c(t) = b_i^{(1)}a_i e^{-a_i t} + b_i^{(2)}a_i(a_i t)e^{-a_i t} + d_i a_c e^{-a_c t} + d_i^{(2)}a_c^{(2)}e^{-a_c^{(2)}t} \quad (22)$$

$$\widetilde{W}_i^c(t) = P(T_s^c > t) = (b_i^{(1)} + b_i^{(2)})e^{-a_i t} + b_i^{(2)}(a_i t)e^{-a_i t^2} + d_i e^{-a_c t} + d_i^{(2)}e^{-a_c^{(2)}t} \quad (23)$$

4. NUMERICAL RESULTS

In order to verify our model of software defined industrial network controller, we developed a simulation model to mimic the queuing behavior. We assume that at the arrival to the node s, packets are queued in the data node before being processed. The processing time of data node is considered to be exponentially distributed with a mean value of 9.8μs, which represents the processing capability. According to the test results of typical SDN switch, the value of 9.8μs is the processing time taken by the switch for forwarding packets of size 1500 bytes. At the controller, the controller is assumed as NOX. In the simulation process, the processing time of controller is considered to be exponentially distributed with a mean value of 240μs, which represents the controller processing capability.

In figure 5, the conditions of different probabilities that the switch cannot match the flow table, controller load of typical Industries SDN control network and industrial SDN control network with flow table cache in controller are compared. The curve typical SDN and the curve flow table cache in controller is obtained by using above mathematical model to calculate. And the curve simulation is obtained from simulating the Industrial SDN control network with flow table cache in controller.

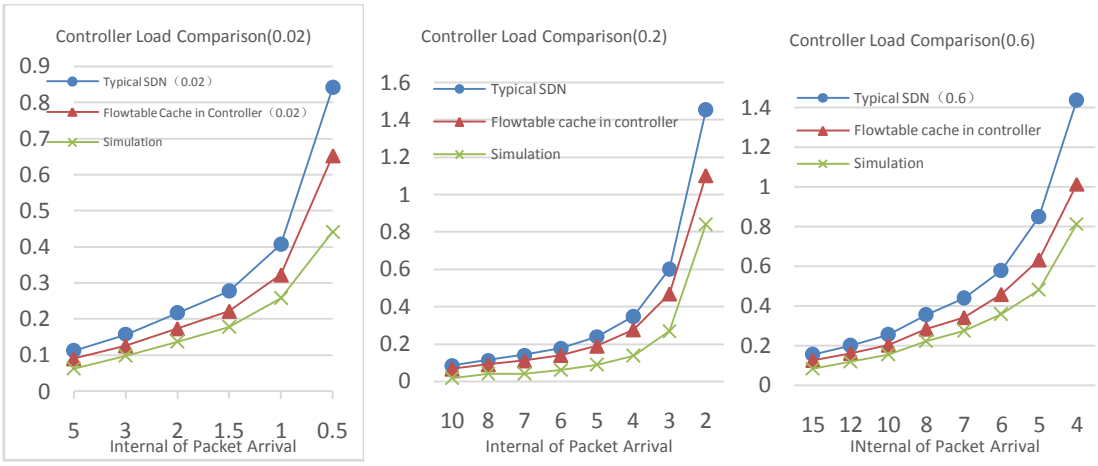

Figure 5. Controller Load Comparison in Different Conditions of Probability

It can been seen that as the packet arrival rate increases, the controller load increases as well. In the other hand, the results of mathematical model calculated have the same general trend with the results of simulation, therefore, the mathematical model proposed herein, can reflect the performance of industrial SDN control network. In figure.5,it can been seen that, in the conditions of different probabilities that the switch cannot match the flow table, the performance of industrial SDN control network with flow table cache in controller is better than that of typical industrial SDN control network.

In the case of increasing the length of flow table cache in controller, the probability that packets can be matched to the flow table cache in controller is increased too. On the other hand, as the length of the flow table cache in controller is increased, the time complexity of look-up and matching table increase. Therefore, we want to find an optimal configuration of the length of the flow table cache in controller. Assuming that doubling or tripling the length of the flow table cache in controller, the time for looking-up table increases to 2 or 3 times corresponding. In the figure below, there are three curves which represent the length of flow table cache in controller.

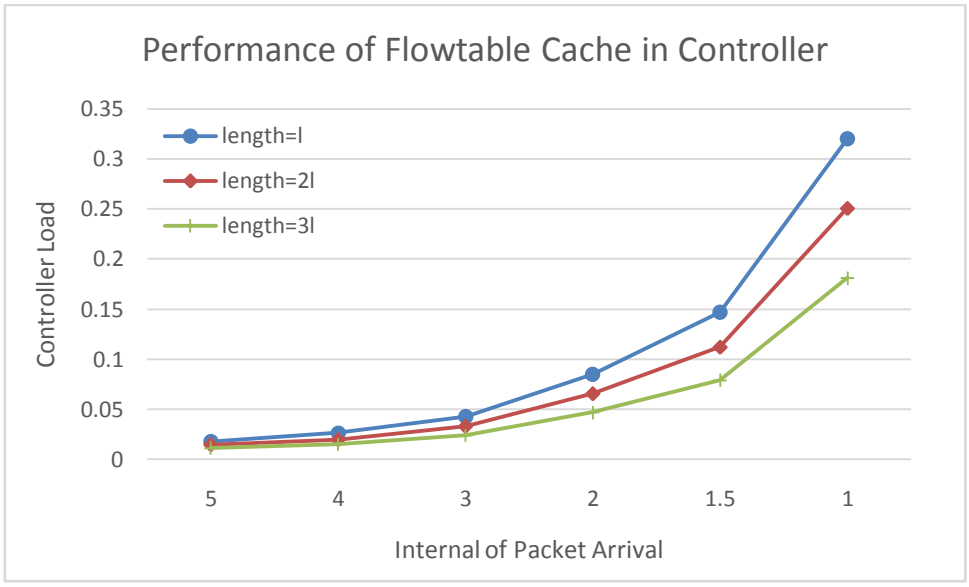

Figure 6. Performance of different length of Flow table Cache in Controller

According to the results shown in Figure 6, It can be seen that, The greater the probability of packets able to match the flow table cache in controller, which means the longer flow table cache in controller, the better the network performance could be. Compared to the influence on controller processing performance by time spent on matching flow table, the increase probability of matching flow table makes more influence on enhancing the performance of the whole network and reducing controller load.

5.CONCLUSIONS

A framework of SDN controller with flow table cache and the mathematical model based on Jackson queuing network are proposed in this paper. The mathematical model is not only for the general industrial SDN control network, but also can be applied on industrial SDN network controller with flow table cache in controller to solve the industrial limitation on the switch. According to the mathematical model proposed, the two metrics for evaluation the entire network can be calculated: packet sojourn time in network and its time distribution probability density function. Through mathematical model simulation, the results show that, under the same conditions, the industrial SDN network controller with flow table cache in controller is more excellent in packet sojourn time, and the network performance has been raised. With the increase of network packet throughput, the effect made by industry SDN network controller with flow table cache in controller on the performance of the network is more obvious.

Based on the proposed mathematical model, future research directions can be expanded to the following aspects: First, extend the current single control plane node to multiple nodes, even some topological scenarios; Second, based on distributed controller architecture, modeling analysis SDN network with a hierarchical structure; Third, aim for some special needs or equipment in industrial SDN control network, make the mathematical model more refined and perfect.

Acknowledgements

The authors acknowledge the financial support of the Strategic Priority Research Program of the Chinese Academy of Sciences under Grant No.XDA06020500.

REFERENCES

Azodolmolky, Siamak, Reza Nejabati, Maryam Pazouki, Philipp Wieder, Ramin Yahyapour, and Dimitra Simeonidou (2013) "An analytical model for software defined networking: A network calculus-based approach", *Proc. of Global Communications Conference (GLOBECOM),* pp. 1397-1402.

Bianco, Andrea, Robert Birke, Luca Giraudo, and Manuel Palacin (2010) "Open flow switching: Data plane performance", *Proc. of2010 IEEE International Conference on Communications*, pp. 1-5.

Bozakov, Zdravko, and Amr Rizk(2013) "Taming SDN controllers in heterogeneous hardware environments",*Proc. of Software Defined Networks (EWSDN),* pp. 50-55.

Ciucu, Florin, and Jens Schmitt (2012) "Perspectives on network calculus: no free lunch, but still good value",*Proceedings of the ACM SIGCOMM 2012 conference on Applications, technologies, architectures, and protocols for computer communication*, pp. 311-322.

Jackson, Matthew O.(2005) "A survey of network formation models: stability and efficiency",*Group Formation in Economics: Networks, Clubs, and Coalitions*, pp.11-49.

Jarschel, Michael, Simon Oechsner, Daniel Schlosser, Rastin Pries, Sebastian Goll, and Phuoc Tran-Gia (2011) "Modeling and performance evaluation of an OpenFlow architecture",*Proceedings of the 23rd international teletraffic congress*, pp. 1-7.

Khan, Ajmal, and Neisarg Dave (2013) "Enabling hardware exploration in software-defined networking: A flexible, portable openflow switch", *Proc. of2013 IEEE 21st Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM),*pp. 145-148.

Kirkpatrick, Keith (2013) "Software-defined networking",*Communications of the ACM 56*, no. 9 (2013): 16-19.

Naous, Jad, David Erickson, G. Adam Covington, Guido Appenzeller, and Nick McKeown(2008) "Implementing an OpenFlow switch on the NetFPGA platform",*Proceedings of the 4th ACM/IEEE Symposium on Architectures for Networking and Communications Systems*, pp. 1-9. ACM, 2008.

Sherwood, Rob, and Y. A. P. KOK-KIONG.(2010) "Cbench: an open-flow controller benchmarker", 2013-05-13]. http://www.openflow.org/wk/index.php/Oflops (2010).

Zeng, P., H-B. Yu, H. Wang, and T-R. Wang.(2002) "Research on Field Level Wireless Communications Protocols." *Information and Control*, 31(5), pp.396-400.

Zeng, Peng, Hai-bin Yu, Ying Liang, Zhi-jun Shang, and Zhong-feng Wang(2004) "On the Architecture and Application Supporting Technology of Distributed Wireless Sensor Network",*Information and Control*, 33(3), pp.307-313.