A Service Curve of Hierarchical Token Bucket Queue Discipline on Soft-Ware Defined Networks Based on Deterministic Network Calculus: An Analysis and Simulation

Shuangyin Ren, Quanyou Feng, Yu Wang, and Wenhua Dou

***Abstract*—Software Defined Networks (SDN) is an emerging architecture for next generation networks and a network paradigm for 5G mobile networks. This paper presents a QoS research of Hierarchical Token Bucket (HTB) queue discipline on SDN paradigm based on Deterministic Network Calculus. In SDN HTB queue discipline, Network Calculus models were designed for packet classification, policing, filtering and FIFO queue scheduling, bandwidth borrowing mechanism is also discussed in Network Calculus model. In SDN topology level, controller-switch feedback model is discussed. These models were employed in a one-controller one-switch SDN network. Delays for different bandwidth and traffic load are obtained from DNC service curve and arriving curve. An SDN network was simulated in Mininet with a remote layer 2 learning switch controller and an OVS switch under specialized ping traffic. Packet delay from Network Calculus is conformant with Mininet simulation result under comparisons of each other.**

***Index Terms*—SDN/Soft-ware defined networks, network calculus, flow control, HTB/Hierarchical token bucket.**

I. Introduction

Software Defined Networks (SDN) has become one of the most popular topics in computer networks in last several years. There are lots version of definitions for SDN, Open Networking Foundation (ONF) proposed that SDN is an "emerging network architecture where network control is decoupled from forwarding and is directly programmable" [1]. In traditional networks, network functions were imbedded into networking elements like hub, switch and router, which causes difficulties for network equipment upgrade, and also constrains innovations. Things changed as SDN deprives control and routing functions from switch/route devices [2]. SDN is an architecture composed by data layer, control layer and application layer. Control layer provides a relatively centralized control to switch/route devices through Openflow protocol, or some other southbound APIs. Data layer forwards traffic flow according to created flow tables and route tables. Application layer fulfils user requires through programming on control layer with its northbound API.

SDN paradigm could run on a diversity of switch/route devices, where different equipment and links have different capability. Queues will form in output interface of these equipment if traffic congested. In real world traffic profile traffic summits in particular time, which is called "network tide", in research and education traffic profile heavy traffic may happen if there are a chunk of data transmitted. It might also happen that flows converges on one switch downlink port. In these scenario traffic flow will queue in output interface, or drop some packets if buffer runs out. There comes a question to provide Quality of Service (QoS), which is how to decide the depth of queue and how many queues are needed. One answer may be that to queue all received packets, this will consume large chunk of silicon memory which overheads more energy, on the other hand more memory may also can't accommodate high speed traffic bursts. If queues are too short, output interface can't accommodate burst neither converged traffic. For TCP packets handshake and confirmation are need, dropped packets will deteriorate network performance. Traffic control and QoS provides guaranteed service to customers by allocating bandwidth and buffer size, or decline a traffic otherwise. B. Hubert [3] and Martin A. Brown [4] provides handbook on network routing and traffic control. N. Beheshti etc. [5] carried out an experiment on effects of buffer size in router performance. Network Calculus [6] is a theoretical tool which provides delay analysis or buffer size analysis for a fixed topology. With service curve model and arriving curve model in Network Calculus, a delay bound to a traffic could get when it goes through a network node. When service curve provided in a traffic path, overall delay bound could also obtain in IntServ (Integrated Service) QoS. A simulation could be set on Mininet to test these analysis. Mininet, as one popular SDN simulation environment, provides well documented tutorials on setting up simulation and modifying route tables [7].

In this article, we will work on analysis modules based on Deterministic Network Calculus (DNC) for SDN packet delay. These models will be created and this module will be taken into packet delay analysis in a fixed topology. Based on DNC, a queue analysis module will also be created for Openflow enabled Open Virtual Switch (OVS). This article will be organized in this way, section I gives an introduction of research background and motivation. In section II will give a basic introduction of DNC and SDN DNC model will be created. Section III will provide a service curve for a fixed topology SDN network and will provide a delay analysis for a ping traffic. An arrival curve for a ping traffic is also provided in this section. Section IV runs a simulation on Mininet with OVS and a remote controller. Then comes a comparison between simulation results and DNC results.

Manuscript received January 28, 2017; revised April 17, 2017. This research is partially supported by the grants from National Natural Science Foundation of China (Grant No. 61402502).

Shuangyin Ren, Quanyou Feng, Yu Wang and Wenhua Dou are with the Computer School of National University of Defense Technology, Changsha, China (e-mail: renshuangyin@nudt.edu.cn, fengquanyou@foxmail.com, ambitionwang@gmail.com, douwh@vip.sina.com).

doi: 10.18178/jacn.2017.5.1.232

Simulation results show that DNC models are conformant with Mininet simulations. Conclusions come in Section V.

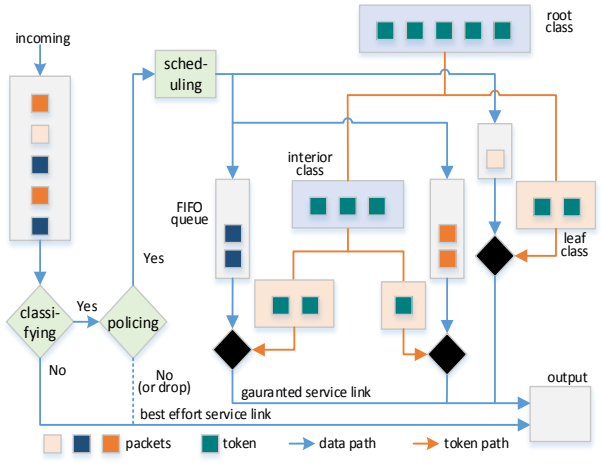

Fig. 1. Flow chat of HTB QoS.

II. DNC Model of Hierarchical Token Bucket Queue Discipline

Network calculus is a mathematical tool for network analysis based on minimum plus algebra [8], maximum plus algebra [9] and also matrix [10]. Network calculus is classified into deterministic network calculus (DNC) and stochastic network calculus (SNC). DNC analyses traffic flow delay bound and network node's buffer size for a guaranteed performance, while on the other hand, SNC [11] provides a delay bound or buffer size with which a percentage possibility service guarantee would be provided to a traffic. This paper takes maximum plus algebra DNC. Two key issues in DNC are how to characterize arriving curve and service curve. Arriving curve is utilized to characterize traffic flow's arriving behavior while service curve is utilized to characterize network elements' ability to provide service for traffic flow. Horizontal distance limit between arriving curve and service curve are data packet delay in this network element. Vertical distance limit between them are buffer size need to provide a guaranteed service. Cruz proposed this theory in two papers [8], [12]. Le Boundec etc. complement deterministic network calculus in book [6].

A. Traffic Control Scenario in HTB

In Hierarchical Token Bucket (HTB) traffic control scenario, classification works on arrival packet based a specific packet portion. There are different type of classifications based on QoS type as IntServ/DiffServ. In IntServ, it will read 5-tuple portion, which is <*src_ip*, *src_port*, *protocol*, *dst_ip*, *dst_port*>, then do classification accordingly. In Differentiated Service (DiffServ), it may just read on part of its header. Unclassified packets will be sent hardware dequeue in best effort service manner. Policing function will drop packets those are out of profile. Two goals of scheduling are packet delay control and link sharing. Scheduling disciplines include first come first serve, priority scheduling, weighted fair queue and earliest deadline first. In HTB, nodes utilize buckets and tokens for bandwidth sharing. HTB system includes root class, interior class and leaf class as shown in Fig. 1. Classes are descripted by assured rate (AR), ceil rate (*CR*), priority (*P*), quantum (*Q*) and level (*L*). Packets could be dequeued to output port only if there are available tokens in that corresponding leaf class. For each leaf class, if data rate exceeds AR and less than CR, while there are excess bandwidth from parent class, this class could 'borrow' bandwidth from its parent. Quotation of borrow means this class do not need to return bandwidth until this resource is not needed, or in the other way, this borrowed resource will not be return even though borrower needs this bandwidth.

B. Classifying and Policing Model

When you submit your final version, after your paper has been accepted, prepare it in two-column format, including figures and tables.

Arrival packets are classified into different classes. Suppose for a fluid model classifying will add a delay τ_c and its service curve is $\beta_c=\delta_{\tau\_c}$, where τ_c varies based on computing power. In packetized model, it offers a service curve as,

$$\beta_c(t)=[\delta_{\tau c}-l_{\max}]^+ \tag{1}$$

where, l_{max} is the maximum length of arriving packets.

If a traffic flow takes $\alpha(t)$ as arriving curve, its arriving curve is $\alpha(t)+l_{max}1_{\{t>0\}}$ after this classifying.

Traffic is policed after classification. Policing function makes sure that a flow do not exceed guaranteed service. Excess traffic may be dropped or sent to best effort path. Policing devices always buffered flows and leaks in guaranteed rate. A packetized shaper is a shaper that forms its output packets has a data rate *r*. Output flow of policing device is,

$$\alpha_p(t)=kv_{T,\tau}=k\left\lfloor {(t-\tau\_d)}/{T}\right\rfloor^+ \tag{2}$$

where, *k* is rate of data flow, *T* is interval and τ_d is packet delay and $\lfloor x\rfloor^+$ is the floor of *x*.

In HTB scenario, traffic was served at *AR* from beginning, if traffic flow bit rate exceeds *AR*, policing criteria changes. For a packetized model, the output flow is,

$$\alpha_p(t)=\begin{cases} k\left\lfloor\frac{t-\tau\_d}{T}\right\rfloor^+ & \text{if } \frac{k}{T}t\le\frac{R}{k}t+B \\ k\left\lfloor\frac{R}{k}\right\rfloor^+(t-\tau\_d) & \text{if } \frac{k}{T}t>\frac{R}{k}t+B \end{cases} \tag{3.1}$$

$$\alpha_p(t)=\alpha(t)\otimes\beta_p(t) \tag{3.2}$$

where *B* is queue length which buffers traffic burst. $\otimes$ is minimum plus convolution,

$$(f\otimes g)(t)=\inf_{0\le s\le t}\{f(t-s)+g(s)\}$$

Equ. (3.1) and Equ. (3.2) shows that HTB policing holds different criteria for different traffic load. When network load runs in a light load, it is a lossless system, while if it runs in a heavy load, it's a loss system and policing drops packets exceed *CR*.

C. FIFO Scheduling and Token Bucket Model

For a basic First Input First Output (FIFO) queuing, a leaky bucket model could be utilized to describe its traffic flow behavior. Service curve of fluid leaky bucket [13] model is,

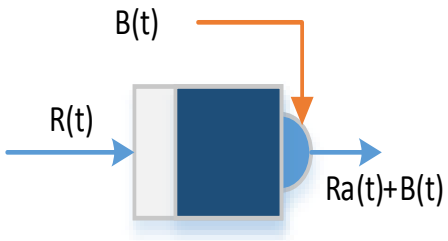

Fig. 2. Leaky bucket model with bandwidth borrowing mechanism in HTB.

$$\beta_b(t) = \begin{cases} \gamma t + \mathrm{b} & \text{if } \mathrm{t} > 0 \\ 0 & otherwise \end{cases} \quad (4)$$

In HTB scheduling, only leaf classes hold packet queue. In a leaf class, if data rate $R(t)$ exceeds assured data rate $R_a(t)$ and less than ceil data rate $R_c(t)$, then this leaf class would borrow bandwidth from its parent class. Parent class could also borrow bandwidth from its parent too. If a parent class has more than one child class and all of them run out of bandwidth, parent would distribute its resource based on child class's priority and quantum. The distribution method is,

$$R(t) = \min(R_c(t), R_a(t) + B(t)) \quad (5.1)$$

$$B(t) = \begin{cases} \frac{QR_{pexcess}(t)}{\sum_{i \in D(p)} Q_i} & \text{if } \min_{i \in D(p)} P_i \leq \mathrm{P} \\ 0 & otherwise \end{cases} \quad (5.2)$$

$$R_{p_{excess}}(t) = R - \sum_{i \in D(p)} AR_i \quad (5.3)$$

where, P is priority of a class.

HTB FIFO could be modelled as a leaky bucket model with a time varying leaky rate, as shown in Fig. 2. Service curve of HTB scheduling model is,

$$\beta_s(t) = \begin{cases} \mathrm{R(t)}t + \mathrm{b} & \text{if } \mathrm{t} > 0 \\ 0 & otherwise \end{cases} \quad (6)$$

where $R(t)$ is real time data rate got from Equ. (5.1) and Equ. (5.2)

D. Service Curve of HTB Node

For a guaranteed service flow, a HTB node provides a service curve as Equ. (7) from Nodes Concatenation Theorem. Service curve of a concatenation system is the minimum plus convolution of its modules service curves in series.

$$\beta_{switch}(\mathrm{t}) = \beta_c \otimes \beta_p \otimes \beta_s \quad (7)$$

In this discussion, all service curve is discussed in packet lossless scenario except policing. Affection of best service data flow to output hardware is not discussed.

III. Delay Analysis in SDN Paradigm

SDN OVS switch architecture as shown in Fig. 3, network routing function is decoupled and moved to SDN controller. When a packet flows in from input interface, switch reads its MAC address to search for its matched output interface. If an output interface matched, packets will be switched directly into an output queue of that interface. However if failed matching, switch saves this received packet and sends packet head to its controller for next action, or send its whole packet to controller for forwarding. If controller failed to get its next hop, packets will be dropped.

A. Service Curve of SDN Networks

In SDN networks, the control cycle service curve is,

$$\beta_{ctl} = \beta_{2\tau\_link} \otimes \beta_{ctl\_0} \quad (8)$$

where β_{ctl_0} is service curve of controller, $\beta_{2\tau\_link}$ is service curve of link between controller and switch while it propagation and transmission delay is $\tau_{\_link}$.

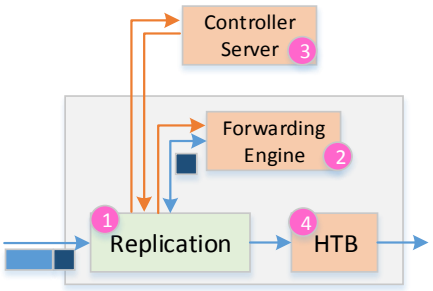

Fig. 3. SDN OVS module for network calculus analysis.

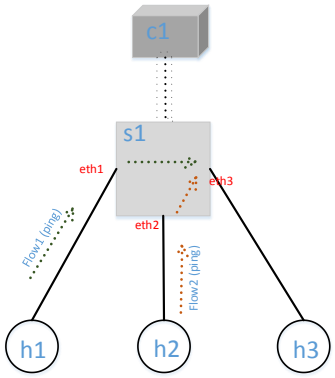

Fig. 4. Topology of SDN network.

Pseudo Code Pseudo code of controller engine

Input: sr_addr, $dest\_addr$, $< addr, switch\_port >$
Return: $outputAction$

```
update control info
if transparent == NO and
      etherType == LLDP then
    return DropAction
end if
if dest_addr == multicast then
    return FloodAction
end if
if dest_addr ∉< addr, switch_port > then
    return FloodAction
end if
if dest_addr == sr_addr then
    return DropAction
end if
send openflow packet to output port
update < addr, switch_port >
```

Fig. 5. Pseudo code of controller engine.

When two queues work with a priority, higher priority flow with arriving curve as α_1 go through higher priority queue, while lower priority flow with arriving curve as α_2 goes through lower priority queue. Flow α_1 will get a service curve β_1, while service curve for α_2 is,

$$\beta_2 = [\beta_1 - \alpha_1]^+ \quad (9)$$

where, $[x]^+$ is max[0, x].

Fig. 4 is a simple topology of SDN network. For a packet without flow table created and a highest priority, service curve is,

$$\beta = \beta_{\tau\_link} \otimes \beta_{ctl} \otimes \beta_{switch} \otimes \beta_{\tau\_link} \quad (10)$$

As for a lower priority, service curve is,

$$\beta = \beta_{\tau\_link} \otimes \beta_{ctl} \otimes [\beta_{switch} - \alpha_i]^+ \otimes \beta_{\tau\_link} \quad (11)$$

where, α_i is arriving curves with a higher priority, τ_{link} is transmission delay between links of hosts and switch. After a flow table created, service curves are,

$$\beta = \beta_{\tau\_link} \otimes \beta_{switch} \otimes \beta_{\tau\_link} \quad (12)$$

$$\beta = \beta_{\tau\_link} \otimes [\beta_{switch} - \alpha_i]^+ \otimes \beta_{\tau\_link} \quad (13)$$

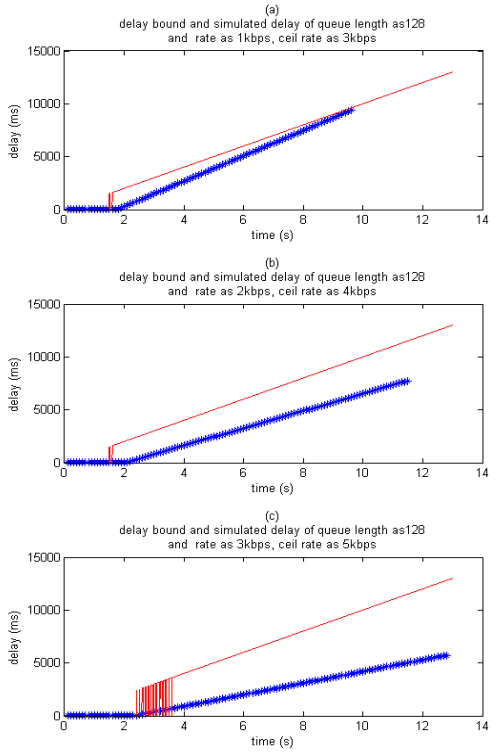

Fig. 6. Network calculus delay upper bound and simulated SDN delay.

TABLE I: PARAMETERS TO DNC MODELS FROM SDN SIMULATION

Parameter	Value	Parameter	Value
k	*480bit*	Q_i	1
B	*7*	AR	*1,2,3kbps*
T	*0.1s*	CR	*3,4,5kbps*
τ_c	*0ms*	τ_link	*0ms*
τ_d	*0ms*		

B. Traffic Arriving Curve and Packet Delay

If this SDN network takes a constant bit rate traffic as an input, its arriving curve is,

$$\alpha(t) = k\left\lfloor {}^{(t-\tau\_d)}/_{T} \right\rfloor^+ \quad (14)$$

Vertical distance of arrival curve and system service curve is buffer size needed for a lossless transport, while horizontal distance is time delay.

$$v(\alpha(t), \beta(t))(t) = \inf\{d | d > 0, \beta(t + d) \geq \alpha(t)\} \quad (15)$$

$$h\big(\alpha(t), \beta(t)\big)(t) = \inf\{b | b > 0, b = \alpha(t) - \beta(t)\} \quad (16)$$

IV. MININET EXPERIMENT DESIGN

A SDN network is realized on Mininet as shown in Fig. 4. Switch realized is based on Open vSwitch (OVS) and a controller is realized as shown in Fig. 5.

In this simulation, VMs are provisioned as host, switch and controller. OVS is an open source software implementation of switch that could be run on virtualized environments like VMs or hardware environments on some switches. OVS integrated openflow APIs to communicate with controllers. These APIs enables remote controller to update OVS address table, set firewalls etc. While "OVS and Mininet are fighting over traffic control queuing disciplines" [14], queue settings are still can't realized from controller. In this proposal, address tables are controlled by remote SDN controller, queue settings are done from OVS commands.

Parameters taken for Equ.15 and Equ.16 are shown in Table I. These parameters are set/acquired in SDN simulation. SDN packet delay is shown in Fig. 5(a,b,c) in red solid line, simulation delay is shown in blue stars, which show that SDN DNC analysis is conformant with Mininet simulations.

V. CONCLUSION

In this paper, SDN DNC models for HTB QoS is created to curve HTB classification, policing, filtering, scheduling and FIFO queuing, models for SDN controller feedback cycle and switching are also created. In scheduling model, bandwidth borrowing mechanism is also discussed. All these models are tested with a constant bit rate stream in a one-controller one-switch topology on Mininet. Comparisons between simulation results and DNC results show that DNC models are conformant with Mininet simulations.

REFERENCES

[1] Fundation, Open Networking, "Software-defined networking: The new norm for networks," *ONF White Paper*, 2012.

[2] W. Xia, Y. Wen, C. H. Foh, D. Niyato, and H. Xie, "A survey on software — Defined networking," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 1, pp. 27-51. 2015.

[3] B. Hubert, *Linux Advanced Routing & Traffic Control HOWTO. Setembro de*, 2002.

[4] M. A. Brown, "Traffic control howto," *Guide to IP Layer Network*, 2006.

[5] N. Beheshti, J. Naous, Y. Ganjali, and N. McKeown, "Experimenting with buffer sizes in routers," in *Proc. the 3rd ACM/IEEE Symposium on Architecture for Networking and Communications Systems ACM*, 2007, pp. 41-42,

[6] J. Y. Boudec and P. Thiran, "Network calculus: A theory of deterministic queuing systems for the internet," *Springer Science & Business Media*, 2001.
[7] M. Team, Mininet: An instant virtual network on your laptop. [Online]. Available: http://mininet.org/download/
[8] R. L. Cruz, "A calculus for network delay. I. Network elements in isolation," *IEEE Transactions on Information Theory*, vol. 37, no. 1, pp. 114-131, 1991.
[9] H. Li, "A performance analysis approach based on stochastic network calculus," Ph.D. dissertation, National University of Defence Technology, Changsha, 2011.
[10] F. Baohua, Z. Heying, and D. Wenhua, "A matrix interpretation of network calculus," in *Proc. 2009 Future Information Networks ICFIN 2009. First International Conf.*, 2009, pp. 45-49.
[11] Y. Jiang, "A basic stochastic network calculus," *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 4, pp. 123-134, 2006.
[12] R. L. Cruz, "A calculus of delay Part II: Network analysis," *Inform. Theory*, vol. 37, no. 1, pp. 132-141, 1991.
[13] G. Niestegge, "The 'leaky bucket' policing method in the ATM (Asynchronous Transfer Mode) network," *International Journal of Digital & Analog Communication Systems*, vol. 3, no. 2, pp. 187-197, 1990.
[14] T. Doxygen. Mininet python API reference manual. [Online]. Available: http://mininet.org/api/classmininet_1_1node_1_1OVSSwitch.html

Shuangyin Ren received the bachelor degree of science of electrical engineering from EECS school, Peking University at June 2010. He received the master degree of engineering of computer science and engineering from Computer School, National University of Defense Technology. Currently, he is a Ph.D student in Computer School, National University of Defense Technology.

Ren's research area is network calculus, wireless sensor networks and software defined networks.