
DOI: 10.1109/ACCESS.2016.2621770

This is the published version.

©2016, IEEE

Reproduced under the terms of the IEEE OAPA copyright policy. Further reuse requires permission from IEEE.

Available from Deakin Research Online:

http://hdl.handle.net/10536/DRO/DU:30091615
A General QoS Aware Flow-Balancing and Resource Management Scheme in Distributed Software-Defined Networks

KESHAV SOOD\textsuperscript{1}, Graduate Student Member, IEEE, SHUI YU\textsuperscript{1}, Senior Member, IEEE, YONG XIANG\textsuperscript{1}, Senior Member, IEEE, AND HONGBING CHENG\textsuperscript{2}

\textsuperscript{1}School of Information Technology, Deakin University, Burwood, VIC 3126, Australia.
\textsuperscript{2}College of Computer Science and Technology, Zhejiang University of Technology, Hangzhou 310032, China.

Corresponding author: H. Cheng (chenghb@zjut.edu.cn)

ABSTRACT Due to the limited service capabilities of centralized controllers, it is difficult to process high volume of flows within reasonable time. This particularly degrades the strict quality of service (QoS) requirements of interactive media applications, which is non-negligible factor. To alleviate this concern, distributed deployments of software-defined network (SDN) controllers are inevitable and have gained a predominant position. However, to maintain application specific QoS requirements, the number of resources used in network directly impacts the capital and operational expenditure. Hence, in distributed SDN architectures, issues such as flow arrival rate, resources required and operational cost have significant mutual dependencies on each other. Therefore, it is essential to research feasible methods to maintain QoS and minimize resources provisioning cost. Motivated from this, we propose a solution in a distributed SDN architectures that provides flow-balancing (with guaranteed QoS) in pro-active operations of SDN controllers, and attempts to optimize the use of instance resources provisioning cost. We validate our solution using the tools of queuing theory. Our studies indicate that with our solution, a network with minimum resources and affordable cost with guaranteed application QoS can be set-up.

INDEX TERMS SDN networks, QoS aware routing, flow-balancing, resources and cost optimization.

I. INTRODUCTION

In Software-defined Networks (SDN), the data and control plane is decoupled and managed by a centralized controller [1]. The software-based central controller significantly allows network engineers to manage network services through the abstraction of high-level of functionality. It further enables network administrators to directly programme the applications and network services, to fasten the network innovation, radically simplifying and automate the management of complex networks at large scale, and so on. Despite of these attractive features, the software-based SDN controllers have an upper limit to process flows at per time, resulting in a significant delay in processing the additional flows and overhead at the control plane. In such a case, it is very critical to maintain application specific Quality of Service (QoS) requirements [2]–[4]. Hence, the distributed deployment of SDN controllers is inevitable and have gained a predominant position.

Researchers observed that decentralized SDN architectures can effectively maintain the specified QoS requirements, and also alleviate the scalability problem [2], [5]–[7]. We observe that in order to maintain QoS, to invest more resources in the network is a general solution, and the discussed issue is not any different from traditional distributed networks. However, with the current SDN trends, scientists identify few concerns:

1) In large scale distributed SDN architectures, the control layer composed of geographically distributed controllers must be able to optimize the number of controllers, locations, and their workload [3]. Furthermore, the control layer must be able to satisfy the performance metrics [8], robust enough to failures, to intelligently distribute flows to reduce data loss, effectively able to optimize the overhead generated by distributed controllers, highly synchronized to consistently update the shared network information [3], [9], and so on.

2) The high load on controller leads to high failure probability and in some cases cause cascading failures of other controllers in distributed networking [3].

3) The inherent key limitations that can further degrade the application specified QoS are, i) the limited...
processing capabilities of logical or Virtual Machine (VM) based controllers and, ii) the service providers specify a limitation of concurrent virtual machines instances that are available to an account, e.g., this limit is 20 in Amazon EC2 [10]. Therefore, it becomes essential to effectively analyze that are current resources allocated to maximum performance and minimize costs and maintaining QoS problems? [11].

On one hand, more resources can significantly reduce the response time of controller to execute a task, and helps to maintain the QoS and eventually gives more revenues to industry. However, on the other hand, facilitating more resources increase the infrastructure and operational cost as well as cost of power consumption, and the cost of energy. This can counterweight the revenue [12]. Thus, maintaining QoS is crucial since it dominates the operational cost.

From this discussion, we can notice that, in order to guarantee a pressed response time, more resources should be invested, which in turn increase provisioning cost (and vice-versa). Hence, factors such as response time, resources required, and associated operational cost have a mutual and significant dependencies on each other. Therefore, it is essential to research feasible methods to maintain QoS and optimize resources provisioning cost without affecting enterprise profit. Here, we emphasize that although each of this aspect (QoS, resource management, and cost minimization) has been addressed in existing research [4], [6], [13]–[17], there is almost no work that addresses all of them simultaneously.

To fill this gap, we propose a solution that provides flow-balancing of SDN controllers that addresses QoS requirements and attempts to optimize the use of instance resources allocated and operational costs to the controller. One can argue that the industry-grade controllers are possibly to be deployed either in conjunction with load balancers or in configurations that provide load balancing and high availability inherently. This eventually helps to maintain QoS. However, we researched that any of the existing flow-balancing configurations would not immediately solve all aspects of the problem being approached. There has been almost similar research done in other areas such as economical modeling and resources optimization. However, research in SDN environment is at an early stage and network performance is examined by real-world experiments without mathematical models.

In this paper, our distributed decision based flow-balancing scheme is different from the existing works in two ways; i) firstly, we address QoS, resource management, and cost factors simultaneously and ii) secondly, none of the existing works deals with per flow-based QoS management and diversion of flows. The existing solutions, [8], [13]–[19], determined the load on the controller based on the PACKET-IN events. Whenever an average number of flow-requests (PACKET-IN events) exceeds the threshold of total flow-request rate of controller, then the controller balanced its load on other controllers in the network. The controller diverts all additional incoming events onto peers, uniformly.

We emphasize that flow-based traffic diversion is possible using flow-based SDN network monitoring tools [3]. Furthermore, as opposed to the related works, we do not find the optimal solution in dynamic SDN environments, rather we attempted to investigate the feasibility and significant benefits of the proposed flow-balancing solution.

This solution (benefits brought by our consideration, i.e., integrated study of QoS, resources, and cost) will helps to alleviate the control layer issues example, optimal number of controllers, their workload and placement. Furthermore, our solution will especially helps the rapidly growing small scale SDN enterprise which are always pragmatic with Information Technology resources allocation and prudent with spending of their financial resources.

Our contributions in this paper are:
1) Firstly, we study the inter-dependencies of the issues such as application specific QoS requirements, resources and operational cost minimization, simultaneously. None of the existing research in distributed SDN networking have integrally studied these factors. This is an early work in this area.
2) Secondly, we propose a QoS aware distributed decision flow-balancing scheme, in order to guarantee the specified QoS performance metrics, and helps in minimizing the resources and operational cost.

The rest of the article is organized as follows. Section II, highlights SDN background and Section III, presents related works. In Section IV and Section V, we describe our proposed method and appropriate system modeling using queuing theory based M/M/1 and M/M/m theoretical tools [20], respectively. Section VI provides performance evaluation of the proposed method. Further discussion in this area is presented in Section VII. Finally in Section VIII, we conclude the paper.

II. SDN BACKGROUND

In the 1960’s a researcher, Paul Baran, working at the Rand Corporation in the United States, proposed that the voice signal from a telephone may be transmitted autonomously through a network in the form of packet data [21]. Further, to increase the network packet forwarding intelligence, Policy Based Routing (PBR) methods were proposed. At that time, a new term “Flow” took birth, it defines particular set of traffic between two end points that receive the same forwarding treatment. PBR defines a set of criteria (commonly known as match-action criteria in SDN that determines whether an incoming packet corresponds to a particular flow or not. This flow forwarding approach provided a base for SDN technology. In this regard we can incorporate PBR at the ground level of SDN [21].

There is a steady progression of solutions and ideas around advancing networking technology prior to SDN. The early efforts include technologies such as Multi-protocol Label Switching (MPLS) (1990), used to separate control software and establish semi-static forwarding paths for flows in traditional routers, Devolved Control of ATM Network (DCAN) (1997), used to separate control and forward plane
in ATM Switches, and Open Signaling (1997) that began with the ATM Switches. Besides, Forward and Control Element separation (ForCES) (2003), 4D named after four plane decisions (2004), and Ethane (2006) are all known as precursors of SDN [21]. Although all these solutions adequately and automatically reconfigure the edge network, the static and manually configured core of the network remains the same. Now, Software-defined Networking (2008) [22] make it possible. Furthermore, the changing traffic patterns, IT consumerization, rapid development of cloud services, and mega dataset that requires massive parallel processing, have also fuelled the innovation in SDN.

We emphasize that one should not believe that SDN is just a dynamic forwarding update technology. Forwarding updates occur in modern non-SDN networks too, and in terms of industry value, forwarding updates are at the low end of the scale. In reality, the decoupling of planes and the programming flexibility allow the SDN elements, especially the switches, to behave as firewalls, load balancers, routers, and so on. Thus, the key features of SDN that make it novel from the existing network trends are: control plane and data plane decoupling, a centralized control entity and view of the network, an Open interface among control and data plane devices, and the programmability of the network by external applications [5].

III. RELATED WORK

The QoS concern was first introduced in SDN by Heller et al. [8]. Authors considered the controller to switch distance aiming to reduce end-to-end delay and to determine optimal number of controllers. Authors have not considered the workload of the controllers, thus their solution is not adaptive to dynamic traffic behavior. Further, the resource utilization concept was also not considered. Egilmez and Tekalp et al. [6] proposed distributed QoS architecture aiming to compute constrained shortest path (CSP). They evaluated the architecture primarily on video quality jitters and analyzed the communication costs, and has not considered flow-balancing and resource minimization factors. In [16] the authors proposed heuristic algorithm to dynamically adjust the controller load, based on average flow request. However, they have not considered multiple factors into account, which we have integrally considered in this work. Furthermore, few researchers proposed per-flow based QoS monitoring and routing at data plane. However, this is not possible at routers edge because of its limited capabilities [1], [3], [24].

Gu et al. [25], with objective of cost optimization, proposed flow-balancing and controller placement algorithm. They proposed to place an identical VM that must maintain the SLA between two data centers. The approach is highly useful, but, is very complex to adopt in real-world environment. Because, data center’s traffic dynamics change very frequently. Thus, it is very challenging to have an intelligent algorithm capable to quickly parse and respond the topology graph (that also changes quickly) and scale to thousand of service, in a very short time. In [17], authors proposed that controller’s response time must meet delay bounds, but this work is not directly aligned with our work.

Furthermore, the existing solutions proposed that the central controller determines the number of controllers required and their placement. Also, these solutions are topology dependent and as the network diameter grows, they become dictate to the underlying systems how the forwarding plane should behave. It acts as strategic control point to manage flow control to the switches/routers via southbound Application Programming Interfaces (APIs) and the applications and business logic via northbound APIs.

**FIGURE 1. A high level view of SDN architecture and work flow.**

Fig. 1 shows a high level view of SDN architecture and work flow. The SDN architectures decouples network control and forwarding functions, enabling network control to become directly programmable and the underlying infrastructure to be abstracted from applications and network services [23]. A switch evaluates every incoming flow independently, finds a matching flow against it, and performs the associated action. If no match is found, the switch forwards packet to controller for getting instructions on how to deal with the packet. Based on the defined intelligent policies, the controller takes decision and updates/populates the switch with the new flow table entries. Please note, however, that only the first packet belonging to a particular flow goes to the controller, while subsequent packets that belong to the same flow get queued at the switch.

Typically, the controller updates switches with the new flow entries as new flow patterns are received. SDN controller, the brain of the network, offer a centralized view of the overall network, and enable network administrators to
non-scalable [4], [19]. Furthermore, they are only applicable at static traffic or load conditions. Besides, Onix [13] proposed a novel strategy to scale the network. But, the database that collects the network state information operates asynchronously and no QoS guarantees are given. HyperFlow [14] and DIFANE [15] introduces new functionalities, in order to balance flows, in switches to reduce controller load. Researchers argue that adding new functionalities at data plane breaks the general concept of SDN [19].

Based on our extensive study, we notice that none of the existing research work has considered flow-balancing and minimizing resources and associated cost together. Further, we observed that the community needs a solution that must be topology independent and adaptive to traffic dynamics. The number of resources can be minimized such that the response time of controller can meet a given delay bound. Thus, we treat this problem having multiple constraints. We emphasize again that our solution has jointly considered multiple factors into account and attempted to reduce the resources requirement and minimize the cost and maintain the application specific QoS constraints.

### IV. DISTRIBUTED DECISION BASED FLOW-BALANCING SCHEME

In this section, we first discuss our proposed distributed decision scheme following which we highlight the deployment framework of the proposed modules in SDN controller.

At small scale enterprise, a single controller in SDN environment may be sufficient to handle influx of data or flows. However, at data centers and large scale enterprise networks, etc. the load on the controller increases beyond its limited capability of processing flows per time. Therefore, with the increase in network diameter, controller load increases too, and then the number of controllers required increase as well. We emphasize that the allocation of controllers should be dynamically performed in order to maintain the QoS requirements, manage the limited available resources (or controllers), and minimize the operational costs. Hence, in order to maintain the QoS requirements, it becomes necessary to map the flow to the controller which can process it in a pressed and required time.

We propose that whenever a flow request arrives at the controller, then the controller decides where the request should be served, i.e., locally or on another controller which can satisfy the application specific QoS requirement $\Delta T$. If it is decided to be served on another controller, the request will be forwarded to the controller having least response time (or say least flow-setup time), $T(t)$, where it will be made to wait in queue and then be served. In other words, we map the application to the controller which can satisfy the applications specific QoS requirements.

With our propose scheme, application can be processed in a very short time which guarantees the QoS requirement of application/flow. We observe that the response time is bounded, i.e., $T(t) \leq \Delta T$. Here, $T(t)$ is the response time of the controller to process a flow for a given time point, and $\Delta T$ is the QoS requirement of the application. We investigate that reducing the mean time a flow spent in the system can minimize the resources and associated cost. Thus, the resources investment problem becomes an optimization issue, i.e.,

$$
\begin{align*}
\min f(R, C),
\text{s.t., } T(t) & \leq \Delta T
\end{align*}
$$

(1)

$f$ is a non-linear function of number of controllers $R$ and associated cost $C$. Here, $R$ and $C$ are inter-related. The solution of the above equation should be that the number of controllers required is unique and optimum. However, as the load changes, obtaining the optimal number of $R$ is not possible. Thus, using distributed individual optimization problem, we define

$$
\begin{align*}
\min R(t),
\text{s.t., } T(t) & \leq \Delta T
\end{align*}
$$

(2)

The logical addition and deletion of the controllers can be obtain using above equation. Now, we focus on minimizing the operational cost of the application which is associated to user. By reducing the response time of the controller, we can significantly reduce the instance resources provisioning cost. As the application is processes in short time period, the user has to pay lesser, this reduces the resource provisioning operational cost. For example, in cloud based or virtual SDN environments the logical adding and deletion of controllers is essential to maximize resource utilization and minimize the required resources, and minimize the operational costs. Again, using distributed individual optimization problem, we define

$$
\begin{align*}
\min (C),
\text{s.t., } T(t) & \leq \Delta T
\end{align*}
$$

(3)

Thus, we investigate a linear economic relationship, i.e., flow arrival rate, resources and cost obeys a linear investment relationship. Moreover, for a given flow arrival rate, a network requires less resources and eventually less cost. Therefore, flow-balancing concept reduces controller’s response time, guarantees application’s QoS, minimize number of resources and associated operational cost.

Now we discuss our deployment framework or proposed modules in SDN controller as shown in Fig. 2. This is a distributed decision scheme in which each controller built with i) state collection and ii) flow-diversion control functions. The inter-SDN communication module provides an interface with the underlaying switches and peer controllers to communicate. The inter-SDN module collects the information of service rate of peer controllers. It also collects the information such as routing and state changes messages etc. All these informations forwarded to state collection control function which updates the forwarding information base (FIB), and again update the inter-SDN control module with update information (to prove the inter-controller communication is out of scope of this paper).
Now, state collection function also exchange this information with flow-diversion control function, which ranks the current service rate of all peers and the self service rate in ascending order (please note, here the service rate we refer is the response time of the controller), which also lead to optimal costs. Whenever a flow arrives at controller, the flow-diversion control function captures the QoS requirement of the application, and then decides whether to serve the application locally or offload it onto the controller having least response rate. Concretely, in order to meet the QoS requirement, the flow arrival rate to each controller should be less than its service rate to keep the system in stable state.

V. SYSTEM MODELING

In this section, we discuss the system modeling. The response time analysis is presented in order to approximate the resources demand for our method.

A. RESPONSE TIME ANALYSIS

Our methodology aligns with few existing works [10], [26], [29] and we assume that the controller in SDN network is modeled as $M/M/1$ discipline, which can be easily extended to $M/M/m$ model to study the performance of SDN networks where controllers deployed in a hierarchical architecture. Researchers mentioned that to date, only the $M/M/m$ model offers a closed form results as the distributions possess attractive properties including additive and memoryless [20], [26]. We, therefore, follow this mainstream tool for our analysis on the proposed strategy.

We assume that the incoming packets obey Poisson distribution, which is justified given that the two processes are on a different time scales [20]. Furthermore, we also assume that the service rate of each individual controller follows an exponential distribution, which is common in queueing analysis and also is in-line with the existing researches [27], [28].

For performance evaluation, average time of flows in system is used as a metric of QoS. Let $\mu$ is the service rate of the controller (or controller’s capacity to handle flows per time) and $\Delta T$ is the QoS requirement of the application or request. Now, in order to guarantee the QoS of application, we need to dynamically allocate resources. In general, $\mu$ is constant, therefore, to guarantee QoS, a flow must be diverted to the controller which satisfies QoS requirement $\Delta T$. Now, as we mentioned that $\mu$ is constant, but, it is possible that the processing capability of controller degrades as the number of incoming flow requests exceeds the threshold processing ability. Thus, the average time a flow spent in the system varies as the incoming flow rate changes. Therefore, we relate $\mu$ as $T(t)$ (response time of the controller to process a flow) for a given time point. In this case, to guarantee QoS, we need to dynamically allocate resources, and make sure that $(T(t) \leq \Delta T)$, for a given time point $t$. Below, using classical $M/M/1$ queueing model, $T(t)$ is calculated.

The probability that the system contains $k$ number of flows is denoted by $p_k$. Here, $p_k = p_0 \prod_{i=0}^{k-1} \frac{\lambda}{\mu + \lambda}$, and $p_0 = 1 - \frac{\lambda}{\mu}$. Further, $M/M/1$ system may be described by selecting birth and death coefficients as, $\lambda_k = \lambda$ where $k = 0, 1, 2, 3,...$, and $\mu_k = \mu$ where $k = 0, 1, 2, 3,...$ Applying these coefficients in $p_k$, we get, $p_k = p_0(\frac{\rho}{\mu})^k$, where, $\rho \geq 0$. For system stability, $0 \leq \rho < 1$, this ensures that $p_0 > 0$, we can say that $p_0$ is a constant. Finally, we get, $p_k = (1 - \rho)\rho^k$. Here, $\frac{\rho}{\mu} = \rho$ further, using Little’s law $\bar{N} = \lambda T$, we compute average number of flows, $\bar{N}$, awaiting in the queue to get process in the queueing system is given by; $\bar{N} = \sum_{k=0}^{\infty} kp_k$. Further simplifying, we get $\bar{N} = \frac{\rho}{1 - \rho}$, and finally

$$T(t) = \frac{1}{1 - \rho} = \frac{1}{\mu - \lambda}. \quad (4)$$

Naturally, from above equation we assume that $T(t)$ meets users’ expectations of QoS. Now, as the incoming flow requests exceeds the threshold processing ability, additional resources are required to facilitate in the network. In this regard, equation (4) needs to change, and the $M/M/m$ queue model is used to analyze the average response time of a controller. A general analyses based on $M/M/m$ queue model is given below.

Again, we assume a system with unlimited queue and with a constant arrival rate $\lambda$. The system provides a maximum $m$ controllers, so

$$\mu_k = \min\{k\mu, m\mu\} = \begin{cases} k\mu & 0 \leq k \leq m \\ m\mu & m \leq k \end{cases} \quad (5)$$

The condition for ergodicity is $\frac{\lambda}{m\mu} \leq 1$. Accordingly,

$$p_k = \begin{cases} \frac{(mp)^k}{k!} & k \leq m \\ \frac{p_0 (p)^k m^m}{m!} & k \geq m \end{cases} \quad (6)$$
Also, \( \rho = \frac{\lambda}{m \mu} \leq 1 \). Now we solve \( p_0 \) which gives us

\[
p_0 = \left[ 1 + \sum_{k=1}^{m-1} \frac{(m\rho)^k}{k!} + \sum_{k=m}^{\infty} \frac{(m\rho)^k}{k! m^{k-m}} \right]^{-1}
\]  

(7)

The probability that new incoming flow has to wait in the queue is given by

\[
p_{\text{queueing}} = \sum_{k=m}^{\infty} p_k = p_0 \frac{(m\rho)^m}{m! (1 - \rho)}
\]  

(8)

\[
= 1 - \sum_{k=0}^{m} \frac{(m\rho)^k}{k!}.
\]  

(9)

As we are interested in finding the average time a flow spent in the system, \( T(t) \), we observe that the number of controllers, \( m \), is another factor on \( T(t) \). More explicitly we express \( T(t) \) as,

\[
T(t, m) = \mathbb{E}[T(t, m)] = \frac{1}{\lambda} \left( m \rho + \rho \frac{(m \rho)^m}{m!} \frac{p_0}{(1 - \rho)^2} \right)
\]  

(11)

Combining \( \rho = \frac{\lambda}{m \mu} \) with above equation, we have

\[
T(t, m) = \frac{1}{\mu} + \frac{1}{\lambda} \frac{(\frac{\lambda}{m \mu})^m}{m!} \frac{p_0}{(1 - \frac{\lambda}{m \mu})^2} \leq \frac{1}{\mu - \lambda}
\]  

(13)

As discussed before, in order to guarantee the QoS of the application, equation (4) must satisfied. Thus,

\[
f(r, m) = \frac{\lambda}{\mu} - (\mu - \lambda) \left( \frac{(r \lambda)^{m-1}}{m! \mu^m} \frac{p_0}{(1 - \frac{r \lambda}{m \mu})^2} \right)
\]  

(15)

VI. PERFORMANCE EVALUATION AND DISCUSSION

In this section, we present the performance results of our solution. We have conducted MATLAB simulations of above discussed scheme at each controller. We use real-datasets and evaluate the analytical theory to validate our findings. From our research, we set a definition.

Definition 1: We proposed to minimize the resources and operational costs subject to the response time constraint: \( T(t) \leq \Delta T \). Now, in the following literature, we validate this definition.

With the theoretical tools at hand, firstly we conduct performance evaluation on couple of aspects. Fig. 3 (a) illustrates the dependence of average time on the system stability factor. From equation (4), vary the value of \( \rho \) and at given \( \mu \) (the condition \( \rho = \frac{\lambda}{m \mu} \) must satisfy), we observe that at lesser values of \( \rho \) the average time a flow spent in the queue is lesser. But, as \( \rho \) approaches to higher values say 0.9, the average time a flow remains in the system grows exponentially. Further, the time spent by a flow in the system is dependent on the service rate of the controller. It can be seen that at high service rate of controller, say 20,000, flow spend relatively lesser time in comparison to other values of \( \mu \).

This analysis indicates that rather than deploying more resources in the network, it is better to deploy controllers with high service rate (or controller’s having least response time). So that the flow-setup time or response time to a flow can be minimized. This validates our finding that flow-diversion towards the controller having least response time helps guarantee the QoS requirements. Furthermore, we observe that the lesser workload waste some capability and resources, and on the otherhand high workload will degrades the QoS or if we attempt to run the system near (but below) its capacity, one has to pay an extreme penalty.
FIGURE 4. Determine queueing probability using $M/M/1$ and $M/M/m$ model. (a) The relationship between $p_0$ and additional resources required ($m$), at a given $\rho$. (b) The relationship between $p_{queueing}$ and additional resources required ($m$), at a given $\rho$.

Modifying the equation (4) according to [29], we get

$$T(t) = \frac{k + 1}{2(\mu - \lambda k)}$$

(16)

From this, we plotted Fig. 3 (b) indicates the relationship between flow-setup time and number of $k$ switches. In this experiment, we set $\mu = 20K$, and $\lambda = 70$, which are reasonably closed to the statistics and real-data (shown in Table I) provided by [29] and [30].

TABLE 1. NOX controller capacity [30].

<table>
<thead>
<tr>
<th>Flow installs</th>
<th>Flow set-up time</th>
</tr>
</thead>
<tbody>
<tr>
<td>30K/Sec</td>
<td>10ms</td>
</tr>
</tbody>
</table>

We can see that at $\lambda = 70$, and at $\mu = 20K$, the controller can manage approximate 80 switches with maximum flow-setup time is 0.5ms. Whereas, if $\mu = 5,000$, the controller’s capacity to manage 80 switches decreases dramatically, i.e., the flow-setup time to manage the switches (at same data rate) changes as $\mu$ changes. Now, this validates our finding that load determination and traffic offloading based on PACKET-IN events may not completely valid, thus, flow-based traffic diversion approaches needs to investigate.

From this experiment, we can see that the controller’s capacity significantly affects the application’s QoS requirements. As a consequence, in order to maintain QoS of specific application, the number of switches should be limited under a certain bound, and dynamic flow-balancing is necessary [29].

Using equation (7) from $M/M/1$ model, Fig. 4 (a) reveals that the probability of number of flows in the queue varies as traffic intensity ($\rho = \frac{\lambda}{\mu}$) varies. Now, using equation (9) from $M/M/m$ model, indicates that by investing more resources in the network, the probability of flows in the queue decreases considerably, as shown in Fig. 4 (b). Comparing Fig 4 (a) and 4 (b) at given $\rho = 0.7$, we observe that the probability of flows in the queue decreases when there are more resources deployed. Furthermore, adding and deleting VM controllers will helps to optimize the resource requirement provided equation (14) must satisfy. Physical controllers can be put into ideal/sleep mode, but must be decided by some optimization scheme. Hence, we observe that the response time decrease considerably and thus by forwarding the flow towards the controller which can handle the processing in acceptable time (Equation 14 must satisfy), we can guarantee the application specific QoS.

Now, we determine the relationship $f(r, m)$, between $r$ and $m$. The values of function at $\lambda = 0.5$ and at given $r$ and $m$ is lesser in comparison to the value at $\lambda = 0.9$. The Fig 5 (a) and (b) is studied into two parts: left and right side. Consider low $r$ and high $m$, i.e., left side, here even if the additional traffic strength is low, more number of resources are required. This is exactly opposite what we expect. We want to minimize resources investment at any time. But to maintain the QoS the investment of resource is inevitable, and $f(r, m) \geq 0$, further equation (14) must satisfy to meet QoS demands.

In the following experiment, using real dataset (shown in Table II) we show a financial cost estimation results, shown in Fig. 6. Here, as long as the resource is active or working, organizations pay accordingly, for e.g., in SDN based cloud environment. Here, we take Amazon EC2 on-demand small Linux instance, whose price is 0.08/hour. Now, to calculate how many number of resources are needed, on-demand, to meet application QoS requirement, a simple method is; $minR(t) = \frac{1}{\lambda_{max}}$, $\lambda$ is the flow arrival rate and $\lambda_{max}$ is the maximum rate of flow a resource can handle at a time. Fig. 6. depicts linear economic relationship, i.e., that flow arrival rate, resources and cost obeys a linear investment relationship. Moreover, for
Determine the relationship between traffic strength and number of resources using function $f(r, m)$. (a) The function value at a given $\lambda = 0.5$. (b) The function value at a given $\lambda = 0.9$.

TABLE 2. Service rate of VM instances [10].

<table>
<thead>
<tr>
<th>Type</th>
<th>Configuration</th>
<th>Price, U.S dollars</th>
<th>Service rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small</td>
<td>1 ECU, 17GB RAM, 160GB disk</td>
<td>0.080</td>
<td>58</td>
</tr>
<tr>
<td>Extra large</td>
<td>8 ECU, 15GB RAM, 1690GB disk</td>
<td>0.640</td>
<td>468</td>
</tr>
</tbody>
</table>

FIGURE 6. Cost estimation at given $\lambda$ and resources.

VII. FURTHER DISCUSSION

Inspite of the widespread interest in SDN networks, very less has been published in-line to the presented work. This paper is an early attempt to explore the dependencies among controller load (or flow management), number of required resources, and operational cost. Therefore, in this paper, we addressed QoS guaranteed resources provisioning costs minimization problem.

To avoid a significant bottleneck at the centralized SDN controller, in order to enhance scalability, the deployment of distributed controllers has been proposed. This is a general approach to achieve scalability. However, the optimal number of controller, their placement, and workload distribution like issues are still remain there. Thus, to guarantee the QoS is very challenging. In network design, QoS is always a priority for network operators in order to deliver a guaranteed services. The existing solutions, in order to maintain QoS, attempted to decreases the controller-switch delay using $K$—center approaches, or $K$—median approaches. However, all existing solution are either topology depended or do not count the workload of controller or resources. With this motivation, we proposed a solution at control layer that minimize the resources provisioning costs and maintain the QoS, further the solution consider application-wise QoS management, and is topology independent.

Now, assume that the performance is satisfactorily maintained by proposing novel design algorithms and architectures, is there any single performance enhancing methodology that best fits on all complex, heterogeneous, and multi-vendor operated SDN environments? Now, again assume that the whole network is replaced with SDN enabled...
devices, therefore, the performance is no longer a critical issue. However, we emphasize that it will still not be possible for all underlying switches (both hardware and software switches) to have similar hardware capabilities and control software behaviors. These diversities significantly affect the performance and are always be a critical issue, needs to urgently address. Thus, we foresee that our proposed solution will helps to alleviate the resource management and QoS issues in hybrid SDN deployment scenarios.

Further, this research sparks many directions. Firstly, a general theoretical analysis is further required and need to analyze the model with different distributions. Secondly, this models needs to capture the significance of multiple data plane nodes in the network, thus a Jackson-feedback mechanism concept can be added to extend the model. Finally, real time prototype will be established to compare the model more rigorously.

VIII. CONCLUSION AND FUTURE WORK

The emergence of SDN is imposing novel requirements due to diverse infrastructural entities and architectures. In this paper, firstly we discussed that the complex, heterogeneous, and hierarchical SDN deployments affect the application performance (QoS) and end-user experience. After that, we analytically studied that the flow arrival rate, number of required resources and associated cost have mutual dependencies that affects controller’s response time. We showed that effective flow-balancing strategies resulting in resources minimization, cost savings, and QoS improvements. We revealed that controller’s high service capability is always better than deploying multiple controllers with low service rate.

In future, firstly, we will analyze the similar concept using $M/M/1/c$ and $M/M/m/c$ models which are more realistic. Secondly, we plan to test this strategy on real SDN network, post that it can be extend to a SDN-based cloud network. We believe that the investigations in this area will accelerate the SDN deployment, as well as provide more chances for SDN adoption in multi-technological domains.

REFERENCES

KESHAV SOOD (GSM’15) received the B.Tech. degree (Hons.) in electronics engineering and the M.Tech. degree in optical fiber engineering from Punjab Technical University, India, in 2007 and 2012, respectively. He is currently pursuing the Ph.D. degree in information technology with Deakin University, Melbourne, VIC, Australia.

He was a trainee with the Terminal Ballistic Research Laboratory, Chandigarh, India. His research interests include SDN security and flow management in SDN.

Mr. Sood is a Professional Engineer, as recognized by Engineers Australia, Barton, ACT., Australia. He served as a TPC Member of various IEEE conferences, including the IEEE INFOCOM, the IEEE Bigdata service, and the IEEE ITNAC 2015.

SHUI YU (M’05–SM’12) is currently a Senior Lecturer with the School of Information Technology, Deakin University. He has authored two monographs and edited two books on big data, over 150 technical papers, including top journals and top conferences, such as the IEEE TPDS, the IEEE TCC, the IEEE TCSS, the IEEE TC, the IEEE TIFS, the IEEE TMC, the IEEE TKDE, the IEEE TETC, and the IEEE INFOCOM. He initiated the research field of Networking for big data in 2013. His h-index is 22. His research interests include networking theory, big data, and mathematical modeling. He is a member of Deakin University Academic Board from 2015 to 2016. He is a member of the AAAS and the ACM. He is the Vice Chair of Technical Subcommittee on big data Processing, Analytics, and Networking of the IEEE Communication Society.

Dr. Yu actively serves his research communities in various roles. He served on the IEEE Transactions on Parallel and Distributed Systems as an AE from 2013 to 2015. He is currently serving the editorial boards of the IEEE Communications Surveys and Tutorials (Exemplary Editor for 2014), the IEEE Access, the IEEE Journal of Internet of Things, the IEEE Communications Letters, the IEEE Communications Magazine, and a number of other international journals. Moreover, he has organized several Special Issues either on big data or cybersecurity. He has served on over 70 international conferences as a member of organizing committee and also as the Publication Chair for the IEEE GLOBECOM in 2015 and the IEEE INFOCOM in 2016. He has served as a TPC Co-Chair for the IEEE BigDataService in 2015, the IEEE ITNAC in 2015, and the Executive General Chair for the ACSW in 2017.

YONG XIANG (SM’12) received the Ph.D. degree in electrical and electronic engineering from the University of Melbourne, Melbourne, VIC, Australia, in 2003.

He is a Professor and the Director of the Artificial Intelligence and Image Processing Research Cluster with the School of Information Technology, Deakin University, Melbourne. He has authored over 110 refereed journal and conference papers. His research interests include signal and system estimation, information and network security, multimedia (speech/image/video) processing, and wireless sensor networks.

Dr. Xiang is an Associate Editor of the IEEE Signal Processing Letters and the IEEE Access. He has served as the Program Chair, the TPC Chair, the Symposium Chair, and the Session Chair for a number of international conferences.

HONGBING CHENG received the Ph.D. degree in network and information security from the Nanjing University of Posts and Telecommunications in 2008. He was a Research Fellow with Nanjing University, China, University of Stavanger, Norway, and Manchester University, England, from 2010 to 2013. He is a Professor with the College of Computer and Software, Zhejiang University of Technology, Hangzhou, China. He has authored over 50 refereed journal and conference papers. His research interests include cloud computing, information and network security, big data security, and wireless sensor networks.

Dr. Cheng has served as the Symposium Chair and the Session Chair for several international conferences. He is an Associate Editor of the Journal of Network and Information Security.