QoC-Aware Control Traffic Engineering in Software Defined Networks

Vignesh Sridharan, *Student Member, IEEE,* Purnima Murali Mohan, *Member, IEEE,* and Mohan Gurusamy, *Senior Member, IEEE*

***Abstract*—In a distributed Software Defined Networking (SDN) architecture, the Quality of Service (QoS) experienced by a traffic flow through an SDN switch is primarily dependant on the SDN controller to which that switch is mapped. We propose a new controller-quality metric known as the Quality of Controller (QoC) which is defined based on the controller's reliability and response time. We model the controller reliability based on Bayesian inference while its response time is modelled as a linear approximation of the M/M/1 queue. We develop a QoC-aware approach for solving (i) the switch-controller mapping problem and, (ii) control traffic distribution among the mapped controllers. Each switch is mapped to multiple controllers to enable resilience with the switch-controller mapping and control traffic distribution based on the QoC metric which is the combined cost of controller reliability and response time. We first develop an optimization programming formulation that maximizes the minimum QoC among the set of controllers to solve the above problem. Since the optimization problem is computationally prohibitive for large networks, we develop a heuristic algorithm — Qoc-Aware switch-coNTroller Mapping (QuANTuM) — that solves the problem of switch-controller mapping and control traffic distribution in two stages such that the minimum of the controller QoC is maximized. Through simulations, we show that the heuristic results are within 18% of the optimum while achieving a fair control traffic distribution with a QoC min-max ratio of up to 95%.**

***Index Terms*—Quality of controller, reliability, response time, Software defined networking, switch-controller mapping**

I. Introduction

The separation of control plane from the data plane in Software Defined Networking (SDN) architecture brings in the benefits of dynamic and flexible network management through a programmable network interface. The control plane is comprised of one or more controllers that manage the switches in the data plane through communication protocols such as OpenFlow [1]. The control plane manages the flow tables in the switches that contain the forwarding rules to forward flows through the network. For example, when a new flow arrives at the switch for which there is no matching forwarding rule, the switch sends a flow-setup request in the form of a `packet-in` message to the controller. The controller determines the route for the flow (based on its network view and data plane traffic engineering policy) and updates the flow table of the switch to handle that flow by replying with a `packet-out` message.

Due to its ease of network management, SDN finds applications in several areas such as Data Centers (DC), Wide Area Networks (WAN), Internet of Things (IoT), and next generation networks to address several issues related to network traffic management such as load balancing, resilience, security, Quality of Service (QoS) etc. However, SDN architecture introduces resilience and scalability issues [2]. A single controller in the control plane becomes a central point of failure and it can cripple the network upon failure. The controller would also not be able to respond effectively to requests coming from the switches as it can get overwhelmed by the number of requests. To mitigate this issue, distributed controller architectures have been proposed in which multiple controllers manage a network, with each controller managing one part of the network [2] [3]. This raises the question of switch-controller mapping, i.e., how and on what basis should the switches be mapped to the controllers.

For efficient and reliable network operation, two important factors that need to be taken into consideration for switch-controller mapping are: i) response time to requests from the switch, and ii) reliability of the controller. Response time to flow-setup requests is an important aspect to ensure efficient network operation. The switch-controller mapping has an impact on the controller response time which is determined by the propagation delay and the load at the controller. Earlier works study switch-controller mapping with various objectives related to response time such as minimizing switch-controller communication latency, inter-controller latency etc. Other studies have approached this mapping problem from the perspective of controller load balancing [4], [5], [6].

Controller reliability is a critical factor to enable a resilient network. Controllers may experience downtime/failure due to many reasons such as software bugs, server or hardware failure, or due to external attacks such as Distributed Denial of Service (DDoS) attacks. Controller reliability is an aspect that should be considered by taking into account the historical data of controller operational issues (for example, a history of attacks/failure on a controller suggests that it is vulnerable to future attacks/failures). Earlier studies on switch-controller mapping do not consider controller reliability and focus only on link/switch reliability [7], [8], [9]. Different from the above works, we consider both controller response time and reliability to solve the switch-controller mapping problem.

If a switch is mapped to only one controller, its failure would affect the switch operation. To overcome this issue, existing works in the literature map each switch to two controllers — one active/primary controller and one passive/backup controller [10]. Upon failure of the primary controller, the backup controller (i) sends role-change request messages (from backup to primary) to the switches that were

The authors are with the Department of Electrical and Computer Engineering, National University of Singapore. Postal address: Block E4, Level 6, Room 12, Engineering Drive 3, Singapore 117583. E-mail: sridharan.vignesh@u.nus.edu, elepumm@nus.edu, elegm@nus.edu.sg.

earlier mapped to the failed primary, and (ii) handles all control traffic from those switches after changing its role to primary. This would however lead to packet losses when there are no active controllers to handle the control traffic immediately after the primary controller failure. To avoid this issue, we consider mapping each switch to multiple primary/active controllers and study the control traffic distribution problem among the mapped controllers such that a minimum fraction of the total control traffic load is guaranteed to be unaffected in an event of controller failure.

In this paper, we propose the Quality of an SDN Controller (QoC) metric based on which switches are mapped to multiple controllers in the distributed SDN controller architecture to enable resilience. The QoC metric takes into account both the reliability of the SDN controller and its response time to flow-setup requests from the switches. The QoC-based mapping thus ensures that the switches are mapped to the most reliable controllers while the controllers are not overloaded, thereby avoiding (i) controllers with higher probability of failure, and (ii) controllers with longer response times due to high load. An SDN controller may fail due to a software/hardware failure or due to an attack (e.g., Denial of Service). We determine the reliability of the controller based on its prior probability of failure using the Bayesian inference model. The controller response time (i.e., the time taken to respond to a `packet-in` message) can be modelled as a M/M/1 queue. Due to the non-linear nature of M/M/1 queue, we propose a linear approximation to the model. Our contributions in this paper are listed as follows:

- We develop an optimization programming formulation for the switch-controller mapping problem that maps each switch to multiple controllers and distributes control traffic between them such that the minimum QoC represented by the combined cost of controller reliability and response time to flow-setup requests is maximized while meeting the capacity and delay constraints.
- We develop a heuristic algorithm, Qoc-Aware switch-coNTroller Mapping (QuANTuM), to solve the above stated optimization problem in a reasonable time and compare *Quantum* with optimal results for small networks and with other heuristic approaches for large networks to highlight its performance.
- We develop a reliability model for controllers based on Bayesian inference and highlight the major factors that affect controller reliability.
- We model the controller response time as a linear function, which we show to be a reasonable approximation of the M/M/1 queue based on which a controller's response time is generally modeled.
- We develop an optimization formulation and a heuristic algorithm to manage the switch-controller mapping when the network experiences short term traffic variation. To avoid the overhead that is incurred if the switches and controllers are re-mapped, we adjust the control traffic distribution while keeping the switch-controller mapping fixed. In the performance study, we provide results and discuss the effect of short term traffic variation on the switch-controller mapping and how it can be managed effectively.
- We compare the performance of the *Quantum* heuristic with the optimal solution for small networks. For large networks, we compare the performance of the *Quantum* with that of a Baseline approach. The results show that *Quantum* achieves within 18% of the optimum in terms of minimum QoC while providing a fair control traffic distribution with a QoC min-max ratio of up to 95%. We also compare the performance of *Quantum* with two other approaches that perform switch-controller mapping to reduce controller response time and maximize reliability respectively.

The rest of the paper is organized as follows. In Section II, we discuss the related work. In Section III we present the framework of the QoC-aware mapping approach and discuss the reliability and controller response time models. We develop an optimization programming formulation to determine the optimal controller mapping and control traffic distribution for improved QoC in Section IV. We present the proposed heuristics and its computational complexity in Section V. We develop an optimization problem to manage the switch-controller mapping in the event of short term traffic variation in Section VI. We present the analysis of results in Section VII before concluding the paper in Section VIII.

II. Related Work

The switch-controller mapping problem is an important issue in distributed controller SDN architecture and has received significant attention in the research community.

There have been several studies that examine the mapping problem with objective of load balancing at the controller to manage the response time of the controller. In [5], the authors propose a dynamic controller assignment problem for SDN switches in data centers to minimize the average response time of the control plane. In [11], Wang et al. propose an online algorithm for dynamic controller assignment in SDN based data center networks to minimize the total cost due to response time and maintenance on a cluster of controllers. Cui et al. [6] develop a controller load balancing approach in which overloaded controllers are identified on the basis of real-time response time and controller load. An appropriate threshold to identify overloaded controllers is determined by taking into account both real-time response time and controller load. Filali et al. [12] propose a dynamic switch-controller assignment scheme to ensure load balancing at the controllers for better response time. They adopt a game theoretic approach to do the switch-controller assignment with the constraint that a controller must achieve a minimum level of utilization. Cheng et al. [4] formulate an optimization problem to place controllers and map switches to satisfy Quality of Service (QoS) requirement, which was expressed as the maximum time within which a flow-setup request must be processed. In [10], the authors study the assignment of slave controllers to switches and ensure load balancing between the slave controllers so that the network is able to support the control traffic load effectively upon controller failure. These studies consider an approach in which each switch is mapped to only one controller while we adopt a multiple mapping approach to provide protection against disruption in the event of controller failure. They also do not consider the reliability aspect of

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2019.2940863, IEEE Transactions on Network and Service Management

controllers while mapping the switches. In [13], we adopt the multiple mapping approach for control traffic engineering. However, we did not consider reliability of controllers in the switch-controller mapping approach.

Some studies investigating the controller placement problem, in which switch-controller mapping is also considered in some cases, have considered reliability in their models; however they focus on the reliability of the links and nodes in the data plane rather than the controller reliability. In [9], the authors propose a reliability metric that represents the expected control path loss probability. They develop a controller placement problem based on the proposed metric to maximize the reliability of the network and propose heuristic approaches to obtain solution in reasonable time. Ros et al. [7] formulate a fault tolerant controller placement problem to ensure that the network achieves a minimum level of reliability and propose a heuristic algorithm for the same. They consider the reliability of the links and nodes in the network in their problem. In [14], Zhong et al. consider the single link failure model and define two reliability metrics based on this model and solve the controller placement problem to maximize the reliability metric. Liu et al. [8] propose a heuristic approach for reliable controller placement based on a path based reliability factor. The above studies do not consider reliability of controllers in their approach and focus on modeling the reliability of the data plane links/nodes in their problem. We also develop a reliability model for the controller based on the possible reasons for its failure. Response time is not considered in the above studies while we consider response time as a main factor in our work along with controller reliability. All the above works also consider a switch-controller mapping the approach in which each switch is mapped to only one controller while we map each switch to multiple controllers.

III. QoC-Aware Mapping Approach

In our proposed approach, the QoC of an SDN controller is determined by i) the reliability of the controller and ii) the response time to the flow-setup requests. We first give an overview of our approach in Section III-A followed by a practical implementation framework in Section III-B. In Section III-C, we discuss the reliability model of the controller and in Section III-D, we model the controller response time and propose a linear approximation of the model. We then briefly discuss QoC of a controller and illustrate the need for QoC-aware mapping in Section III-E.

A. Overview

In our problem, we consider a distributed controller scenario where multiple controllers manage a switch. A controller can be mapped to a switch in one of three possible roles: i) Master, ii) Slave and iii) Equal role. In the master and equal role, a controller can manage the flow tables of the switches while in the slave role, a controller is restricted to read-only access of the switches [1]. The main difference between the master and equal role is that in the former role, there is only one master controller that has read-write access to a switch, whereas in the latter role, multiple equal/peer controllers have the read-write access to a switch. When a switch is mapped to multiple controllers in Master-slave mode (i.e., one Master controller and one or more slave controllers), the master controller handles all the flow-setup requests from the mapped switch. In an event of master controller failure, all the flow requests from the mapped switch is disrupted until one of the slave controller changes its role to master upon consent with the affected switch. Had the failed controller played the master role for multiple switches, the problem would become worse.

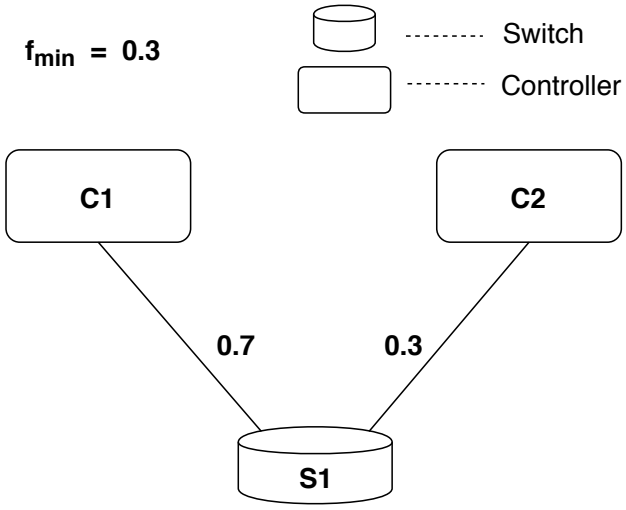


Figure 1: Illustration of Multiple Controller Mapping

Hence in our approach each switch is mapped to a specified number of controllers (represented by $\mathcal{M}$) in the equal role, and each switch distributes its flow-setup requests between the $\mathcal{M}$ controllers to which it has been mapped. This ensures that not all flow-setup requests are disrupted upon a controller failure. The ratio in which a switch distributes its flow-setup requests is decided by the solution to the problem formulated in Section IV. The purpose of mapping a switch to multiple controllers at a time is to provide a certain degree of protection to the flow-setup requests by ensuring that at least a certain minimum percentage of requests would not be disrupted in the event of a controller failure [13]. The minimum fraction of flow-setup requests that a switch must send to a mapped-controller is specified by f_{min}. We assume that the switch is capable of distributing the flow-setup request packets once the distribution ratios are calculated at the control plane by solving the optimization problem. Thus, a flow-setup request is sent to only one controller. We discuss this possibility of flow-setup request distribution from one switch to multiple controllers in Section III-B.

In Fig. 1, we illustrate the multiple mapping approach with a switch that is mapped to two controllers ($\mathcal{M} = 2$) and $f_{min} = 0.3$. As seen in Fig. 1, S_1 distributes 70% of its flow-setup requests to C_1 while it sends 30% of the requests to C_2. Thus, in the event of a controller failure, at least 30% of the requests that are sent to the controller are not disrupted.

B. Framework Implementation

In this section, we present a practical implementation framework to realize our proposed approach of mapping each switch to two controllers ($\mathcal{M} = 2$) and split the control traffic among the mapped controllers [15]. The controllers are mapped to the

switches in `EQUAL` role, i.e., both the controllers can manage a switch. We assume that the switch is capable of performing simple operations for the purpose of distributing the control traffic to the two controllers based on the distribution ratio calculated at the control plane (by solving the optimization problem in Section IV) [16]. When a new flow arrives at the switch, the switch sets a flag in the the packet header and forms the `packet-in` message (flow-setup request). The flag is set to either 0 or 1 based on the distribution ratio to decide whether the `packet-in` message has to be sent to the first or second controller. The `packet-in` message is then forwarded to the group-table entry that has `action-buckets` — one with the flag set to 0 is sent to controller C_1 (for example) and the second action for a packet with flag set to 1 is forwarded to C_2 which then responds with a corresponding `packet-out` message. Based on this `packet-out` message, the flow table in the switch is updated.

C. Reliability Model

SDN controllers are essentially a network operating system that manages the network operation from a logically centralized location. Thus, SDN controllers are generally hosted on servers which can be standalone or as part of a data center. The main reasons for failure of a controller are:

- **Software Failure** : The reason for a software failure could be software bugs that were not discovered in the testing process. Since all the controllers in a network operated by one operator utilize the same controller software, we do not consider software failure in our reliability model as all the controllers are equally susceptible .
- **Hardware Failure** : Since the controllers are run on a server, the hardware reliability is determined by the reliability of the server. In [17], the analysis shows that failure of the hard disk component in servers is the primary reason behind server failures in data centers. Data collected on server failures showed that 78% of the failures were due to hard disks [17].
- **External Attack** : In this case, the controller fails due to external entities attacking it. Since the controller is a very important component of the network, it can be a target for attacks such as Distributed Denial of Service (DDoS) attack. A controller that has a past history of attacks targeting it suggests that it is more vulnerable than a controller that has not been attacked.

We model the reliability of the SDN controller based on Bayesian inference. With SDN, it is possible to record an event of controller failure and this prior knowledge is used to determine the reliability of the controller. Periodically, we learn the instantaneous values of the number of times a controller has failed due to either hardware failure or an attack and buffer the statistics in buffer $\mathbb{B}$. At time t or when the buffer is full (whichever is earliest), we sample the buffer $\mathbb{B}$ and estimate the probability of the controller failure as $f(c)$, where c denotes the controller. We define γ as the weight factor that weighs the importance of historical data and current estimates. We then determine the prior probability, $p(c)$, by averaging the estimates over time as shown below.

$$p(c) = \begin{cases} f(c), & \gamma = 0; \\ \gamma.p(c) + (1-\gamma).f(c), & 0 < \gamma < 1; \\ p(c), & \gamma = 1; \end{cases} \quad (1)$$

We assume an exponential distribution for estimating the probability that a controller will fail, given the number of times it has failed currently. We denote this probability as $p(c'|c)$. Based on the prior probability that a controller will fail, given by $p(c)$, we determine the posterior probability $p(c|c')$ as shown in (2) using Bayes' theorem.

$$p(c|c') = \frac{p(c'|c).p(c)}{\sum_{\mathcal{C}} p(c'|c).p(c)} \quad (2)$$

where, the denominator is the normalizing factor.

The reliability of controller c denoted as R_c, is then determined as one minus the posterior probability as shown below.

$$R_c = 1 - p(c|c') \quad (3)$$

The closer the value of R_c is to one, the higher is the reliability of controller c.

D. Response Time Model

When a switch receives a new flow for which it does not have a corresponding forwarding rule in the flow table, the switch sends a flow-setup request in the form of a `packet-in` message to the controller, which processes the request and responds with the forwarding rule in a `packet-out` message. Thus, the response time to a flow-setup request is the sum of: i) two way switch-controller propagation delay and ii) time taken by the controller to process the request. In this section, we model the processing time of the controller.

1) Linear Approximation: The controller is generally modeled as an M/M/1 queue [4]. If τ is the capacity of a controller (number of flow-setup requests processed per unit time), and the load at the controller is represented by l (number of flow-setup requests per unit time that the controller receives from the switches to which it is mapped), then average response time μ_n for a given load l of the controller as per the M/M/1 model (denoted by $\mu_n(l)$) is given below in Eq. (4):

$$\mu_n(l) = \frac{1}{\tau - l} \quad (4)$$

The response time is non-linear, and for the purpose of formulating the optimization problem, we approximate the response time as a linear function of controller load (Fig. 2). The M/M/1 response time rises exponentially as the load reaches close to capacity (Fig. 2); thus we assume that the controller load does not exceed a certain percentage of the capacity to avoid very high response times. This limit is expressed as a percentage of the total processing capacity τ and is represented by c_{max}. The linear response time (μ_{lin}) for controller load smaller than c_{max} is given by Eq. (5) below:

$$\mu_{lin}(l) = Al + B \quad (5)$$

$$A = \frac{1}{\tau^2(1 - c_{max})} \quad (6)$$

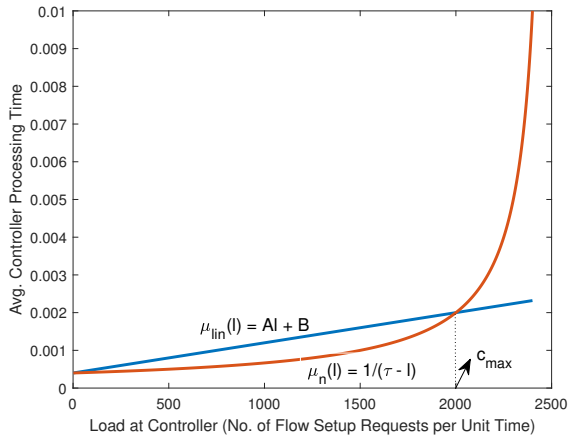


Figure 2: Linear Approximation of M/M/1 Response Time

$$B = \frac{1}{\tau} \tag{7}$$

The coefficients of the linear equation, A and B, are calculated by using the two points of intersection of the linear curve with the non linear curve (Fig. 2). The upper bound for the relative difference in response time between the M/M/1 model and the linear approximation is described below in Section III-D2.

2) Error Bound for Linear Approximation: The relative difference, μ_{diff}, between the M/M/1 response time and the linear approximation is,

$$\mu_{diff}(l) = \frac{\mu_{lin}(l) - \mu_n(l)}{\mu_n(l)} \tag{8}$$

The range of values that load l can take is [0,τc_{max}]. In this range, μ_{diff} is simplified to the form shown below in Eq. (9):

$$\mu_{diff}(l) = (Al + B)(\tau - l) - 1 \tag{9}$$

We find the maximum relative difference in the given range of controller loads ([0,τc_{max}]) to identify the load at which the approximation error is maximum. To find the maximum relative difference in the given range, we differentiate $\mu_{diff}(l)$ (from Eq. (9)) as shown in Eq. 10:

$$\mu^{'}_{diff}(l) = 0 \tag{10}$$

The solution obtained for Eq. (10) is,

$$l = \frac{\tau c_{max}}{2} \tag{11}$$

From the solution derived in Eq. (11), we observe that relative difference is maximum when the controller utilization is $\frac{c_{max}}{2}$. If we assume that controllers are expected to have a reasonably high utilization rate of at least 80%, then the maximum relative difference between the actual response time (as given by Eq. (4)) and the linear approximation (Eq. (5) occurs when utilization is 40%. However, given that controllers are not expected to have such low utilization for long periods of time, we examine the maximum relative difference when controller utilization is in the range [70%,80%]. Since $\mu_{diff}(l)$ is a concave function and its maximum value occurs when $l = \frac{c_{max}}{2}$, the maximum value of $\mu_{diff}(l)$ for controller utilization in the range [70%,80%] will occur when controller utilization is 70%. When $c_{max} = 0.8$, the maximum value of $\mu_{diff}(l)$ in the range of controller utilization [70%,80%] is determined to be 35%. Thus, in the range of interest, the linear approximation is within 35% of the M/M/1 response time.

E. Quality of Controller (QoC)

The Quality of Controller (QoC) that is perceived by a switch takes into account both the reliability of a controller and the flow-setup latency. The controller response time (as described in Section III-D1) and the two-way propagation delay between the switch and the controller define the flow-setup latency for a switch.

We illustrate the need for a QoC-aware mapping with a toy model network as shown in Fig. 3. We consider a network with 3 controllers (C_1, C_2, C_3) and 2 switches (S_1, S_2). Each switch is to be mapped to two controllers ($\mathcal{M} = 2$), and let the minimum fraction of flow-setup requests that need to be sent from a switch to a controller be 0.2 ($f_{min} = 0.2$). The load from each switch in terms of `packet-in` messages per unit time is given in Fig. 3a and the switch-controller propagation delay is shown in Fig. 3b. Each controller has a processing capacity of 220 `packet-in` messages per unit time, and a maximum capacity utilization threshold of 80% ($c_{max} = 80\%$). The reliability of controllers C_1, C_2 and C_3 are 0.9, 0.8 and 0.6 respectively (represented by R_1, R_2, R_3 in Fig. 3c and 3d). The time units for controller response time and switch-controller propagation delay are the same.

We show two scenarios for switch-controller mapping with control traffic distribution in Fig. 3c and 3d. The fraction of control traffic distributed from the switches to the controllers is shown along the lines that map the switch to the controller (Eg.: S_1 sends 70% of its load to C_1 and 30% to C_2 in Fig. 3c). In scenario 1, most of the load from the switches is distributed to controllers with higher reliability while in scenario 2, the load is distributed in higher proportion to controllers that are closer to the switches with better load balancing at the controllers. As seen in Fig. 3e and 3f, in scenario 1, the switches have higher average reliability by up to 10% (weighted sum of the reliability experienced by the load from a switch) and experience high flow-setup latency while in scenario 2, the switches have lower average reliability and experience lower flow-setup latency by up to 35%. The flow-setup latency is calculated as the sum of the controller response time (described in Section III-D1) and the two-way switch-controller propagation delay.

Thus, there is a need to consider both reliability and flow-setup latency while mapping switches to controllers and distributing the control traffic among them. Based on operational requirements, one of the factors may have higher precedence over the other. These factors are taken into account by QoC, which is explained in further detail with mathematical definitions in the following section (Section IV).

IV. Problem Formulation

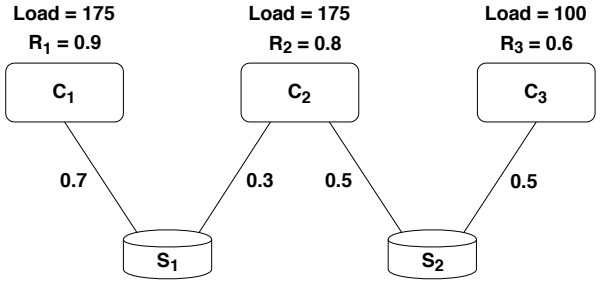
In this section, we present the optimization problem formulation for the $\mathcal{M}$-controller mapping problem that determines

Switch	Load (packet-ins/unit time)
S_1	250
S_2	200

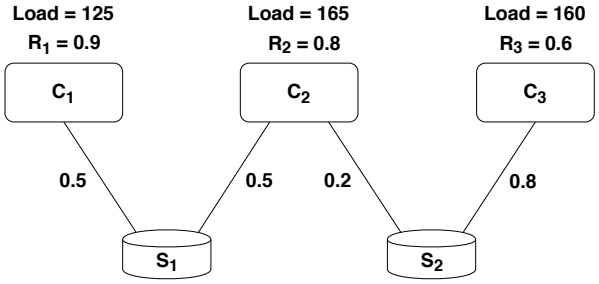
(a) Load at Switches

	C_1	C_2	C_3
S_1	0.04	0.02	0.08
S_2	0.07	0.04	0.01

(b) Switch-Controller Propagation Delay



(c) Mapping Scenario 1



(d) Mapping Scenario 2

Switch	Avg. Reliability	Avg. Response Time
S_1	0.87	0.090
S_2	0.70	0.069

(e) Metrics for Scenario 1

Switch	Avg. Reliability	Avg. Response Time
S_1	0.85	0.0767
S_2	0.64	0.0514

(f) Metrics for Scenario 2

Figure 3: Illustration to Show Need for QoC Aware Mapping

the optimal switch-controller mapping for better controller QoC. Let $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ denote the graph representing the network topology where $\mathcal{N}$ is the set of nodes in the network, and $\mathcal{E}$ is the set of edges which represent the links among the nodes. Let $\mathcal{C}$ represent the total set of controllers in the network. Each controller in the network has a reliability value R_j associated with it. The packet processing time at the controller (and hence the response time) depends on the controller load (as per the model described in Section III-D). We first present the problem statement formally before the problem formulation.

A. Problem Statement

The $\mathcal{M}$-controller mapping problem is formally stated as follows. *" Given a network $\mathcal{G}(\mathcal{N}, \mathcal{E})$ and a set of controllers $\mathcal{C}$ placed at predefined location, find the optimal mapping and fraction of control traffic sent from each switch to $\mathcal{M}$ controllers ($\mathcal{M} \geqslant 2$) that maximizes the minimum of the QoC (among all controllers), which is defined as the weighted function of controller reliability and response time (propogation and processing delay) while meeting the controller capacity and switch-controller delay constraints"*

B. Objective Function

The objective of our proposed problem is to maximize the minimum QoC of the set of controllers. The quality provided by a controller j mapped to a switch i is given by $\theta_{i,j}$, which is defined as the weighted sum of the controller reliability and the response time (i.e., the time required to respond to a request from the mapped switch).

$$Max. \min_{j} \left\{ \sum_{i \in \mathcal{N}} \theta_{i,j} \right\} \tag{12}$$

where,

$$\theta_{i,j} = \left(\alpha R_j + (1 - \alpha) \left(1 - \frac{T_{i,j}}{T_{max}} \right) x_{i,j} \right) \tag{13}$$

The controller reliability, R_j, is determined based on the server reliability and history of attacks on the controller, as described in Section III-C. Whereas the response time for a flow-setup request, $T_{i,j}$, is defined as the sum of the controller processing delay and twice the propagation delay (request from switch to controller and response from controller to switch). The controller processing delay is obtained through the linear approximation function described in Section III-D. T_{max} is a constant that represents that maximum possible response time to a flow-setup request in the network (sum of largest possible propagation delay in a given network and the response time of controller at c_{max} utilization). The purpose of T_{max} is to normalize response time in the range [0,1]. Since a lower value of $\frac{T_{i,j}}{T_{max}}$ implies better performance in terms of response time, we model the response time component of $\theta_{i,j}$ as shown in Eq. (13). We introduce a weight factor, α, that determines the relative importance of controller reliability and response time. $x_{i,j}$ is a binary variable that denotes whether a switch $i \in \mathcal{N}$ is mapped to controller $j \in \mathcal{C}$.

We introduce another variable $f_{i,j}$ that indicates the fraction of control traffic load from switch i that is sent to controller j. The control traffic load from switch i is represented in terms of `packet-in` messages per second by L_i. The total response time of controller j to respond to a request from switch i is as shown below.

$$T_{i,j} = \left(2D_{i,j} + A \left(\sum_{i' \in \mathcal{N}} f_{i',j} \cdot L_{i'} \right) + B \right) \tag{14}$$

Table I: Mathematical notations

Notation	Description
$\mathcal{G}(\mathcal{N}, \mathcal{E})$	Graph representing the network topology
$\mathcal{N}$	Set of nodes/switches in the network
$\mathcal{E}$	Set of edges/links in the network
$\mathcal{C}$	Set of controllers available in the network
$\mathcal{M}$	Number of controllers to which each switch has to be mapped for improved resilience
R_j	Reliability of controller j
τ_j	Maximum processing capacity of controller j
L_i	Load in terms of `packet-in` messages from switch i
$D_{i,j}$	Propagation delay between switch i and controller j
$T_{i,j}$	Total response time for switch i to receive response from controller j
f_{min}	Minimum guaranteed fraction of control traffic upon a controller failure
α	Factor that determines importance of controller reliability and response time; Range: [0,1]
A, B	Constants to represent the response time as a linear function
$\theta_{i,j}$	Quality provided by controller j for switch i
c_{max}	Maximum controller capacity utilization threshold
T_{max}	Constant that defines the maximum possible response time to flow-setup request
δ	Maximum allowed delay between a switch and its mapped controller
$x_{i,j}$	binary decision variable $x_{i,j} = 1$ if switch i is mapped to controller j, $x_{i,j} = 0$ otherwise
$f_{i,j}$	auxiliary variable that represents the fraction of control traffic sent from switch i to controller j

where the linearization constants A and B are defined as,

$$A = \frac{1}{\tau_j^2(1 - c_{max})}; \quad B = \frac{1}{\tau_j} \tag{15}$$

C. Constraints

1) Mapping constraint: The mapping constraint shown below ensures that each switch in the network is mapped to $\mathcal{M}$ controllers. To enable resilience towards controller failures, the value of $\mathcal{M}$ has to be at least two.

$$\sum_{j \in \mathcal{C}} x_{i,j} = \mathcal{M}, \quad \forall i \in \mathcal{N} \tag{16}$$

where,

$$x_{i,j} \in \{0, 1\}, \quad \forall i \in \mathcal{N}, \forall j \in \mathcal{C},$$

2) Capacity constraint: Since the controllers are limited in processing capacity and also we assume that controllers are not heavily loaded to avoid exponential rise in processing time (as described in Section III-D), we limit the load handled at each controller by c_{max} times the processing capacity of that controller.

$$\sum_{i \in \mathcal{N}} x_{i,j} \cdot f_{i,j} \cdot Li \leqslant c_{max} \cdot \tau_j, \quad \forall j \in \mathcal{C} \tag{17}$$

3) Fraction constraints: Each switch distributes the control traffic load among its $\mathcal{M}$ mapped controllers. The constraint in (18) ensures that, the sum of all fractions of control traffic load distributed from each switch to its mapped controllers is one.

$$\sum_{j \in \mathcal{C}} f_{i,j} = 1, \quad \forall i \in \mathcal{N} \tag{18}$$

$$f_{i,j}\,(1 - x_{i,j}) \leq 0, \quad \forall i \in \mathcal{N}, \forall j \in \mathcal{C} \tag{19}$$

$$x_{i,j}\,(f_{i,j} - f_{min}) \geq 0, \quad \forall i \in \mathcal{N}, \forall j \in \mathcal{C} \tag{20}$$

The constraint in (19) ensures that a fraction of control traffic load (`packet-in` messages) from a switch i is sent to controller j only if the switch is mapped to that controller. Whereas, the constraint in (20) ensures that the fraction of control traffic load that is sent from a switch to its mapped-controller is at least f_{min}, i.e., a minimum guaranteed fraction of control traffic will always be responded to even when one or more controllers fail.

4) Delay constraint: The delay constraint shown below in (21) ensures that the total response time (which is the sum of propagation and controller processing time) experienced by any switch in the network is less that a given threshold.

$$T_{i,j} \leq \delta, \quad \forall i \in \mathcal{N}, \forall j \in \mathcal{C} \tag{21}$$

We note that the non-linear constraints in (17) and the objective function can be linearized using standard linearization techniques described in [18]. The linear optimization program is solved using the Linux GLPK[1] solver on SageMath[2] platform running on an Intel Xeon E5-2630 server with 64 GB RAM. The running time of the GLPK solver to obtained an optimal solution is empirically found to be about 2 hours for a 10 node random topology. Since the optimization problem is computationally prohibitive for larger networks, (≥ 10 nodes) we develop a fast heuristic algorithm (in Section V) to solve the $\mathcal{M}$-controller mapping problem.

V. Heuristic Algorithm

In this section, we describe our heuristic algorithm, QoC-Aware coNTroller Mapping (QuANTuM), to obtain a mapping solution in reasonable time. The pseudocode is presented in Algorithm 1. The input parameters for the algorithm are network features such as the number of switches and controllers, topological details such as propagation delay between the switches and controllers, load at each switch and operational requirements including $\mathcal{M}$, delay constraint (Eq. (21)), maximum capacity of controllers c_{max} (Eq. (17)) and f_{min} (Eq. (20)). The heuristic produces the switch-controller mapping matrix X as the output, which determines the fraction of packet-in messages that a switch sends to a controller.

[1]GLPK: https://www.gnu.org/software/glpk

[2]SageMath: http://www.sagemath.org/

Algorithm 1 Qoc-Aware coNTroller Mapping (QuANTuM)

Input: D, τ, c_{max}, f_{min}, L, $\mathcal{N}$, C, $\mathcal{M}$.
Output: $X = (x_{i,j}) \in [0,1)^{C \times \mathcal{N}}$

1: /*Start of Initialization Stage*/
2: $E_j = \{i | D_{i,j} \geq \delta, \forall i \in \mathcal{N}\}$ /* Eligible set of switches for controller j */
3: Initialize $q_{i,j} = 0, \forall i \in C, j \in \mathcal{N}$
4: $q_{i,j} = \frac{\alpha R_j L_j}{max_j(L)} + \frac{(1-\alpha) D_{i,j} L_j}{max_j(L) max_{i,j}(D_{i,j})}$
5: $q_{i,j} = \frac{q_{i,j}}{\sum_j (1 \; if D_{i,j} \geq \delta, \; else \; 0)}$
6: For each controller j, select eligible switch i with highest $q_{i,j}$ and assign f_{min} traffic to controller j /* End of Initialization */
7: **while** $\left(\sum_{i}^{C} \sum_{j}^{\mathcal{N}} x_{i,j} L_j < \sum_{j}^{\mathcal{N}} L_j \right)$ **do**
8: Select controller C_j with lowest QoC
9: For C_j, select node $N_i \in E_j$ such that $q_i = min(q)$ for all switches in E_j
10: **if** $x_{i,j} = 0$ **then**
11: $x_{i,j} \leftarrow f_{min}$ /* f_{min} constraint guarantee */
12: **else**
13: $x_{i,j} \leftarrow x_{i,j} + min\_block$
14: **end if**
15: Update E_j
16: **end while**
17: /*End of Initialization Stage*/
18: /* Improve mapping after initial mapping*/
19: θ_j^{curr}: Current QoC of controller j
20: $\theta_j^{imp} \leftarrow max(\theta_j^{curr})$/* Initialize for comparing current QoC against improved QoC
21: **while** $\frac{min(\theta_j^{imp}) - min(\theta_j^{curr})}{min(\theta_j^{curr})} > 10^{-10}$ **do**
22: Find θ_j: Controller j with min. QoC
23: Find θ_k: Controller k with max. QoC
24: Find W: Set of common, eligible switches between controller j and k
25: **if** $W = \varnothing$ **then**
26: Find θ_k: Controller k with max. QoC where $W \neq \varnothing$; exit while loop otherwise
27: **end if**
28: **for** $N_i \in W$ **do**
29: **if** Transfer of N_i improves θ_j **then**
30: Transfer N_i between C_j and C_k.
31: Update X, θ_j^{imp}
32: **end if**
33: **end for**
34: **end while**

1) Quantum Overview: Quantum follows a round-robin approach to obtain a switch-controller mapping solution. *Quantum* has two stages in which the final switch-controller mapping is obtained:

- Initialization Stage: To obtain an initial switch-controller mapping and control traffic distribution.
- Mapping Improvement Stage: To improve upon the initial solution and obtain the final switch-controller mapping and control traffic distribution.

We first obtain an initial mapping solution in a greedy manner and then we refine the initial solution to improve the mapping. We first initialize the mapping matrix X and subsequently in each round, we assign a fraction of requests from a switch to the controller with least QoC, thus improving the minimum controller QoC value in every round of assignment. After the initial mapping is obtained, the mapping solution is further improved in an iterative manner, where switch-controller assignments are updated so as to improve the minimum QoC of the set of controllers in the network.

2) Initialization Stage: In the initialization stage of the algorithm (line 2- 6), for each controller, we obtain a set of eligible switches that can be mapped to it (line 2). A switch is considered as a part of the eligible set E_j for controller C_j if it satisfies the delay constraint and does not overload the C_j on sending at least f_{min} fraction of requests to it. To identify which switch should be mapped to a controller in each iteration, we define a switch-controller pair metric for the heuristic, $q_{i,j}$ (line 3-line 6), that quantifies the quality of a switch-controller mapping pair by taking into account the reliability and response time, which is normalized based on the maximum load of a switch in the network and the maximum propagation delay between a switch-controller pair.

After the initialization process, we map switches to controllers in a round robin manner. In each round, the controller with the least QoC is chosen and the switch with the highest value of $q_{i,j}$ in the eligible set of the controller is selected and the minimum fraction of traffic that can be mapped in each iteration (min_block, line 13) is assigned to the chosen controller. The value of min_block determines the granularity with which control traffic is assigned to a controller in an iteration. For instance, if $min\_block = 0.1$, and control traffic from switch i is assigned to controller j, then the value of $f_{i,j}$ is increased by 0.1. In other words, 10% more of control traffic is assigned from switch i to controller j. However, if a switch is being assigned to a controller for the first time, then f_min fraction of traffic is assigned in that iteration to satisfy the fraction constraints set in Eq. (20). The switch-controller mapping is continued in this manner until all the control traffic from the switches is mapped (as represented by condition in line 7). Thus, at this stage, we obtain the initial mapping solution.

3) Mapping Improvement Stage: After the initial mapping solution, we improve mapping solution by considering the controllers with maximum and minimum QoC in each iteration and re-assign common switches between them to improve the minimum QoC of the set of controllers (line 19-line 31). We denote the minimum QoC of the set of controllers in the current iteration as θ_j^{curr} and the improved minimum QoC of the controller set after the iteration as θ_j^{imp}. We initialize θ_j^{curr} and θ_j^{imp} as shown in lines 22 and 23. This improvement phase runs until the improvement obtained is very small (line 21).

4) Computational Complexity: The complexity of *Quantum* is O($N^2.\binom{|C|}{\mathcal{M}}$), where $|C|$ represents the number of controllers in the network and $\mathcal{M}$ is the number of controllers to which each switch has to be mapped. In a realistic network setting, the number of controllers in the network is likely to be a small number as the main principle of SDN is centralization of control. The value of $\mathcal{M}$ is also expected to be small since mapping a switch to a few controllers is good enough

for ensuring resilient service and mapping to many multiple controllers at a time will lead to switch-controller communication overhead. Thus, $\binom{|C|}{\mathcal{M}}$ will be a small value. Hence, the complexity of *Quantum* can be stated as O(N^2).

VI. Short Term Traffic Variation

In Section IV and Section V, we developed the problem formulation and heuristic algorithm to obtain the switch-controller mapping and control traffic distribution based on long term estimates of the load from the switches (represented by L). It is possible that due to the dynamic nature of network traffic, the load from the switches can experience short term variation [19], [20], which will then impact the minimum QoC of the controllers in the network. Re-running *Quantum* to find a new mapping solution for the short term variation scenario would result in significant overhead between controllers due to the changes in switch-controller mapping pairs; i.e. switches can be assigned to new controllers. Additional overhead will be further incurred when the short term traffic conditions subside and normal network traffic is experienced, requiring further changes to mapping to revert to the original switch-controller mapping that was present before experiencing traffic variation.

To avoid this overhead, for periods of short term traffic variation, we keep the existing switch-controller mapping (that was obtained based on long term traffic estimate) and adjust the control traffic distribution for the existing switch-controller mapping pairs to alleviate the situation while avoiding high overhead. We present the modified problem formulation for Short Term Variation Management (STVM), based on the long term mapping formulation (as described in Section IV) below.

A. Problem Formulation

The objective for STVM remains the same, i.e. to maximize the minimum QoC among the set of controllers:

$$Max. \min_{j} \left\{ \sum_{i \in \mathcal{N}} \theta_{i,j} \right\} \tag{22}$$

where,

$$\theta_{i,j} = \left(\alpha R_j + (1-\alpha) \left(1 - \frac{T_{i,j}}{T_{max}} \right) X_{i,j} \right) \tag{23}$$

The key difference is that the switch-controller mapping ($X_{i,j}$) is kept the same as the long term switch-controller mapping. The control traffic distribution ($f_{i,j}$) is adjusted to account for short term variation.

We maintain the same constraints for STVM with the only difference being that switch-controller mapping is fixed ($X_{i,j}$ is a constant) based on the long term mapping obtained by *Quantum*.

1) Capacity Constraints:

$$\sum_{i \in \mathcal{N}} X_{i,j} \cdot f_{i,j} \cdot L_i^s \leqslant c_{max} \cdot \tau_j, \quad \forall j \in \mathcal{C} \tag{24}$$

The load from switch i is the load value that has changed due to traffic variation and is represented by L_i^s. This is the varied load from the switch i for a short duration.

2) Fraction Constraints:

$$\sum_{j \in \mathcal{C}} f_{i,j} = 1, \quad \forall i \in \mathcal{N} \tag{25}$$

$$f_{i,j} \left(1 - X_{i,j}\right) \leq 0, \quad \forall i \in \mathcal{N}, \forall j \in \mathcal{C} \tag{26}$$

$$X_{i,j} \left(f_{i,j} - f_{min}\right) \geq 0, \quad \forall i \in \mathcal{N}, \forall j \in \mathcal{C} \tag{27}$$

Since for frequent short-term traffic variations the fraction distribution solution has to be obtained quickly, we propose a heuristic algorithm for the STVM formulation to obtain the adjusted control traffic distribution while keeping the switch-controller mapping fixed as per the STVM problem formulation above.

B. STVM Heuristic Algorithm

We consider a greedy approach which we describe below as follows:

- Step 1: We first identify the controller with least QoC.
- Step 2: We then find the switch that contributes the maximum load and adjust the traffic distribution of that switch to reduce load on the controller with least QoC and increase load at the other controller to which the switch is mapped.
- Step 3: The traffic distribution is updated if this change leads to a higher minimum QoC as compared to before.
- Step 4: Steps 1-3 are repeated until there is no possible switch that is mapped to the controller with the least QoC that can re-distribute the load to improve QoC.

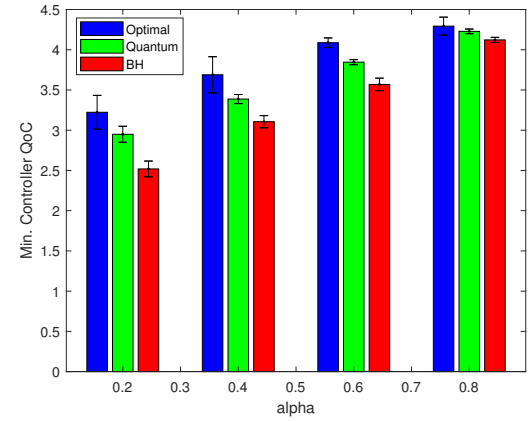
VII. Performance Study

In this section, we present the performance study of *Quantum* for $\mathcal{M}$-controller mapping problem. We first present the simulation setting and then analyze the results obtained.

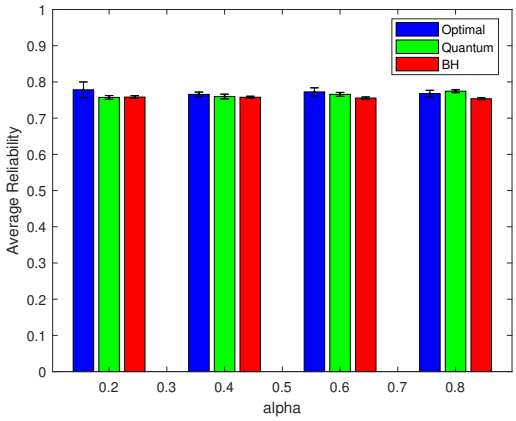
A. Simulation Settings

Since we have large number of decision variables in the optimization programming formulation, we use 10 node random topologies to obtain the optimal mapping solution. For the 10 node topologies, the network has 3 controllers. Whereas for evaluating the heuristics, we run the simulation for random topologies of size in the range $30 - 50$ nodes with an edge creation probability of 0.6. We set the edge/link delay in the range $0.5 - 5$ ms. The load from each switch is distributed uniformly between $(100, 300)$ `packet-in` messages. The processing capacity of each controller is set to 1500 `packet-in` messages per second and the controller reliability is uniformly distributed between [0.75,1]. The number of controllers in the network is scaled according to the network size. We set $\mathcal{M} = 2$ in our simulations, i.e. each switch is mapped to

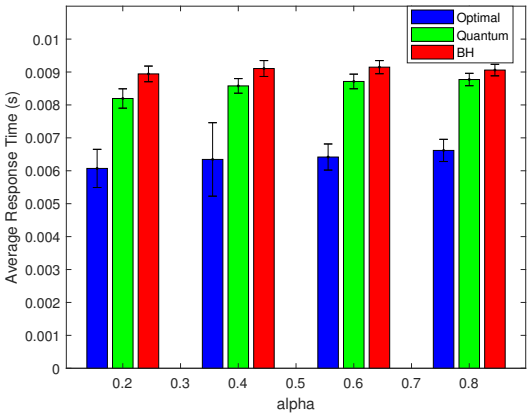
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2019.2940863, IEEE Transactions on Network and Service Management



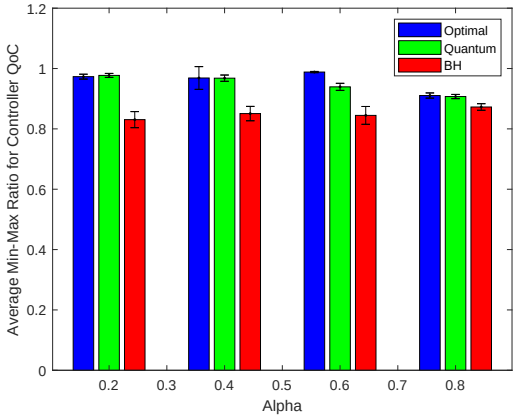
(a) Minimum QoC vs. α



(b) Average Reliability vs. α



(c) Average Response Time vs. α



(d) Min-Max Ratio of Controller QoC vs. α

Figure 4: Comparison with Optimal Results

two controllers. We set $\mathcal{M} = 2$ to minimize switch-controller communication overhead. We run each simulation several times until we achieve a small 95% confidence interval.

B. Performance Metrics

We use the following performance metrics to evaluate the proposed approach.

- **Minimum QoC**: This metric refers to the minimum QoC delivered by a set of controllers. This is the metric that is maximized in our optimization problem formulation in Eq. (12).
- **Average Reliability**: We compare the performance based on the average, weighted reliability experienced by a request in the network. This metric is calculated as shown in Eq.

$$\text{Avg. Reliability} = \frac{\sum_{i\in\mathcal{M}}\sum_{j\in\mathcal{N}} L_j f_{i,j} R_i}{\sum_{j\in\mathcal{N}} L_j} \quad (28)$$

- **Average Response Time**: This refers to the average response time (sum of controller processing time and two way propagation delay) experienced by a request in the network.

$$\text{Avg. Response Time} = \frac{\sum_{i\in\mathcal{M}}\sum_{j\in\mathcal{N}} L_j f_{i,j} T_{i,j}}{\sum_{j\in\mathcal{N}} L_j} \quad (29)$$

- **Fairness in QoC**: We obtain the min-max ratio of the QoC delivered by the set of controllers to quantify the fairness of the performance of the controllers.

C. Baseline Algorithm for Comparison

We consider a baseline heuristic (BH) that is greedy in nature for comparison. In the BH approach, each controller greedily maps and distributes control traffic from the switches in a round robin manner based on the reliability and response time between the switch and the controller. In each iteration, the metric based on which the controller selects the switch that is mapped to it is given in Eq. (30) below:

$$q_{i,j} = \frac{\alpha R_j L_j}{max_j(L)} + \frac{(1-\alpha) D_{i,j} L_j}{max_j(L) max_{i,j}(D_{i,j})} \quad (30)$$

The difference between BH and *Quantum* is that *Quantum* follows a more flexible approach to mapping that first obtains an initial solution in a greedy manner before improving

the mapping solution in an iterative manner as described in Section V-3.

D. Analysis of Results

We first compare our approach on a small, randomly generated topology with 10 nodes in Section VII-D1. We then analyze the results on a larger topology in Section VII-D2.

1) Small Topology: We first vary the value of the weight, α, between $[0.2, 0.8]$ in steps of 0.2 to observe the performance of *Quantum*. In Fig. 4a, we plot the minimum QoC obtained by solving the optimization problem, *Quantum* and BH with varying α. This is the value that we maximize in the optimization problem formulation (Eq. (12)). The minimum QoC that is achieved through the mapping solution obtained by *Quantum* is at least 82% of the minimum QoC attained by the optimal solution to (Eq. (12)) while BH obtains less than 35.4% of the optimal solution. The minimum QoC value increases with increasing value of α. This can be attributed to the greater contribution of the reliability factor with increasing value of α. Reliability has higher value than response time numerically, which explains the increase in minimum QoC value. Fig. 4b shows the average weighted reliability (as calculated based on Eq. (28)) of a node. *Quantum* achieves a similar performance to the optimal mapping solution. The average response time in the mapping solution obtained by *Quantum* is within 22% of the response time obtained by the optimal mapping solution while BH achieves a response time within 25% (Fig. 4c). Fig. 4d shows the fairness in performance of the controllers, with the optimal solution and *Quantum* achieving upto 95% min-max ratio, signifying that the controllers provide relatively similar performance on average to the switches while BH achieves a min-max ratio close to 80%.

2) Large Topologies: We compare the performance of *Quantum* against BH algorithm for larger topologies in this section. We first compare them by varying values of α for randomly generated topologies with 40 nodes and 6 controllers, each with a processing capacity of 2500 `packet-in` messages per second. Fig. 5a shows that the minimum QoC obtained by *Quantum*'s mapping solution is better than the minimum QoC of BH by up to 20%. The average reliability of a switch is similar in both the heuristic solutions (Fig. 5b); however, *Quantum* improves the average response time by up to 28.5%. *Quantum* achieves higher min-max ratio by up to 20% as seen in Fig. 5d. The running time of *Quantum* is about 25ms.

We also compare the performance on several random networks of different size. We vary the number of controllers in the range $[5, 7]$ as the size of the network increases. We keep $\alpha = 0.4$ for all the topologies in this comparison. We observe the same trend across different network sizes. *Quantum* achieves higher minimum controller QoC up to 25% (Fig. 6a) as compared to BH. The average reliability obtained by both heuristics is similar while *Quantum* outperforms BH in terms of average response time by at least 12.5% as seen in Fig. 6c.

Quantum outperforms BH in all the metrics due to the mapping improvement stage of *Quantum* (SectionV-3) that modifies the mapping and control traffic distribution solution obtained initially through a greedy approach.

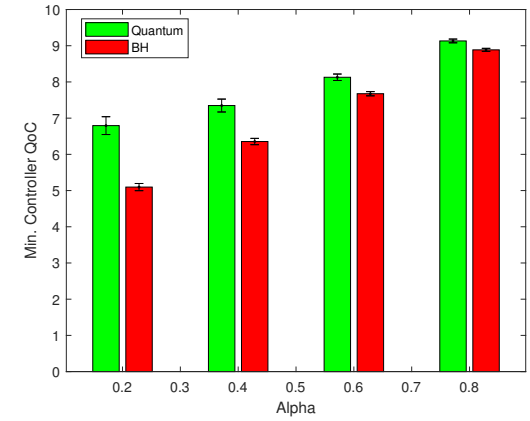
3) Comparison with Other Approaches: We compare the performance of *Quantum* against two other approaches that perform switch-controller mapping with the objective of improving controller response time and reliability, respectively. For response time, we consider the algorithm proposed in [5], where the objective is to assign switches to controllers to minimize the average response time of the control plane and formulate their objective as a stable matching problem with transfers. It is a two-phase algorithm to obtain a switch-controller mapping that aims to minimize the above stated objective. Here, each switch is mapped to a single controller. We refer to this approach as "Stable Matching With Transfers" (SMWT).

We also consider a reliability-based greedy heuristic in which each switch is assigned to a controller based on the controller reliability. In a round robin manner, each switch finds the most reliable controller that has sufficient capacity to serve the switch. The switch is then mapped to that controller. This continues until all the switches are mapped. We refer to this heuristic as Reliable Controller Mapping Strategy (RCMS).

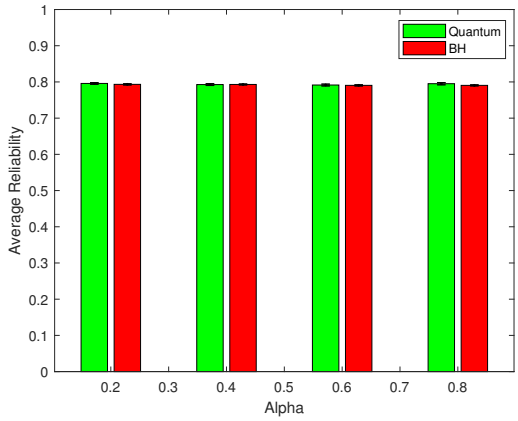
In order to have a fair comparison between *Quantum*, SMWT and RCMS, we consider a scenario where each switch is mapped to a single controller, i.e. a switch sends all its flow setup requests to a single controller. In the context of *Quantum*, it implies that $\mathcal{M} = 1$ and all the control traffic is sent to a single controller. We also consider smaller propagation delay in the range of $[0.5\mu s, 5\mu s]$ to ensure that the comparison is not drastically affected by it since SMWT is proposed in the context of data centers and RCMS does not take propagation delay into account. We also relax the delay constraint for the purpose of comparison.

In Fig. 7, we present the results of the comparison. Fig. 7a shows the minimum QoC achieved by each approach for different values of α. *Quantum* achieves at least 30% better minimum QoC as compared to BH, SMWT and RCMS. *Quantum* performs better than SMWT and RCMS in this regard since it takes into consideration both reliability and response time of controllers while obtaining the switch-controller mapping. Fig. 7b shows the average reliability for different values of α. RCMS performs better than the other approaches by 5% for all values of α since it greedily maps to the most reliable controller. SMWT achieves similar performance to *Quantum* and BH since it aims to balance load between the controllers, thus resulting in a balanced load across all controllers. Fig. 7c shows the performance in terms of average response time for different values of α. SMWT achieves better response time compared to other approaches for $\alpha \geq 0.4$ by up to 23% (as higher value of α implies higher importance for reliability over response time). When $\alpha = 0.2$, *Quantum* achieves marginally better response time as compared to SMWT since it takes into account propagation delay as well, while SMWT considers load balancing at the controller. Thus, the comparisons show that *Quantum* is effective in achieving better minimum QoC.

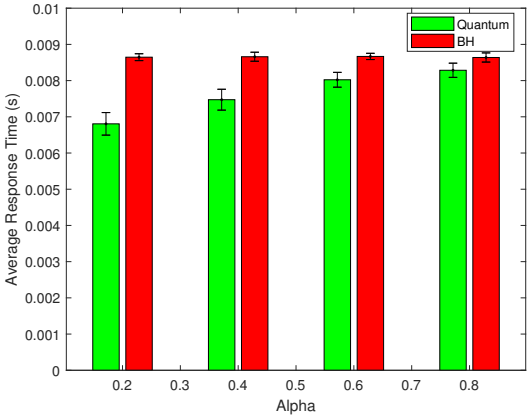
4) Short Term Traffic Variation: To obtain insights into the effect of short term variation in traffic on QoC, we evaluate the STVM optimal solution and STVM heuristic solution (presented in Section VI) in comparison with the case of



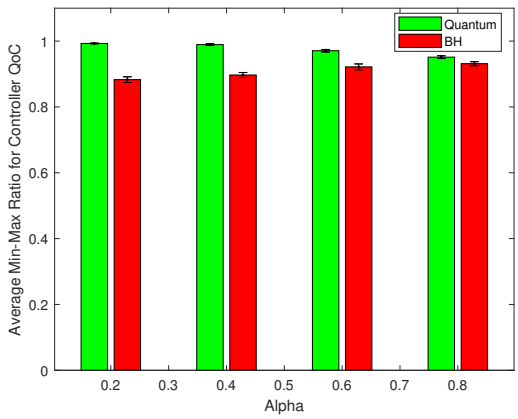
(a) Minimum QoC vs. α



(b) Average Reliability vs. α



(c) Average Response Time vs. α



(d) Min-Max Ratio of Controller QoC vs. α

Figure 5: Performance Study for Large Topology

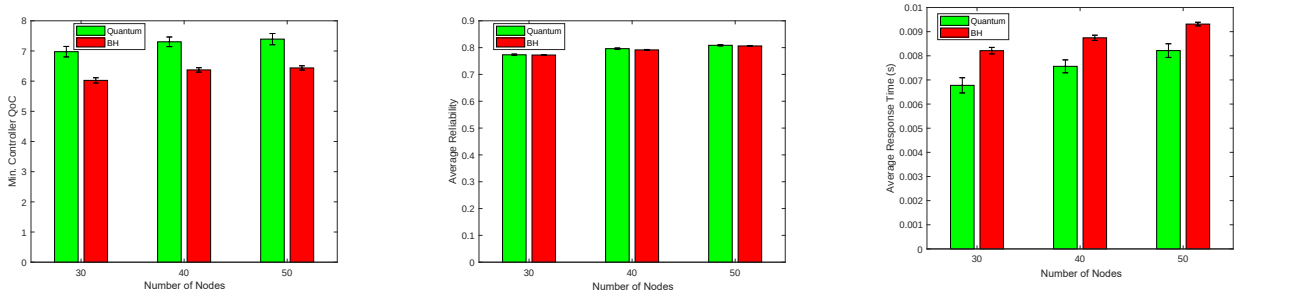
Quantum with no change in mapping or traffic distribution.

We consider a 40 node topology with 6 controllers managing the network. We first generate a long term load estimate for each node in the range of $[100, 300]$ `packet-in` messages/s (as done for simulations in Section VII-D1-VII-D2). For each node, we randomly vary the long term load value in a range of $[0\%, max\%]$ where max is the maximum percentage of change in the load at the node for a short duration. This effectively leads to an increase of $\frac{max}{2}\%$ in total load from the nodes to the control plane. We generate many such varied load values and evaluate the performance of STVM-Heuristic against STVM-Optimal and *Quantum* as shown in Fig. 8 for different ranges of variation.

As seen in Fig. 8, STVM-Heuristic achieves minimum QoC within 5% of STVM-Optimal while achieving up to 20% better minimum QoC than the retained, *Quantum* based solution. The minimum QoC decreases for higher range of variation as observed in Fig. 8. This is due to the net increase in load for the short duration which increases the response time of the controller, thus affecting the minimum QoC. This is more prominent when the propagation delay is low and the controller response time dominates the overall response time. We observe from Fig. 8 that for higher range of variation, there is higher benefit in utilizing STVM-heuristic as compared to the unchanged mapping based on *Quantum* solution. The running time of the STVM-heuristic is about 15ms.

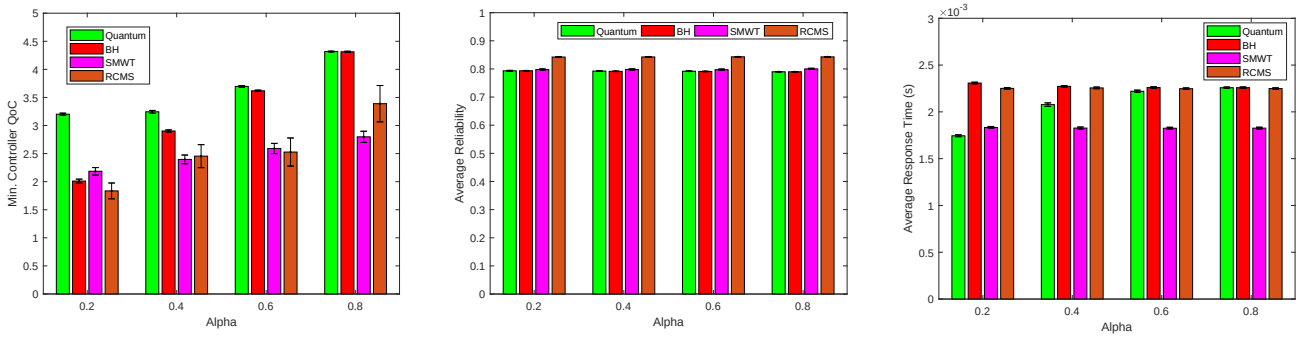
5) Mininet Emulation: In this section, we discuss our implementation of the multiple mapping approach and the performance upon controller failure [15]. Our implementation is set up on an Intel Xeon E5-2630 server (64GB RAM). We run 4 Floodlight controllers on dedicated Virtual Machines (VMs) while Mininet and Wireshark are set up on a separate VM. We create a network in Mininet based on Abilene topology with 11 nodes and examine the recovery time in the multiple mapping approach and compare it against a master-slave mapping approach. In the master-slave approach, each switch in the network is mapped to two controllers; one controller in the master role and the other controller in the slave role. In the event of failure of the master controller, the slave controller assumed the master role for the switch. In the multiple mapping approach, each switch is mapped to two controllers in the equal role (as described in III-A). When one of the controllers is taken down, the flow-setup requests are sent to the other controller to which the switch is connected. Since the other controller is in equal role for

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2019.2940863, IEEE Transactions on Network and Service Management



(a) Minimum QoC vs. No. of Nodes (b) Average Reliability vs. No. of Nodes (c) Average Response Time vs. No. of Nodes

Figure 6: Performance Study for Different Network Sizes



(a) Minimum QoC vs. α (b) Average Reliability vs. α (c) Average Response Time vs. α

Figure 7: Comparison of *Quantum* with BH, SMWT and RCMS

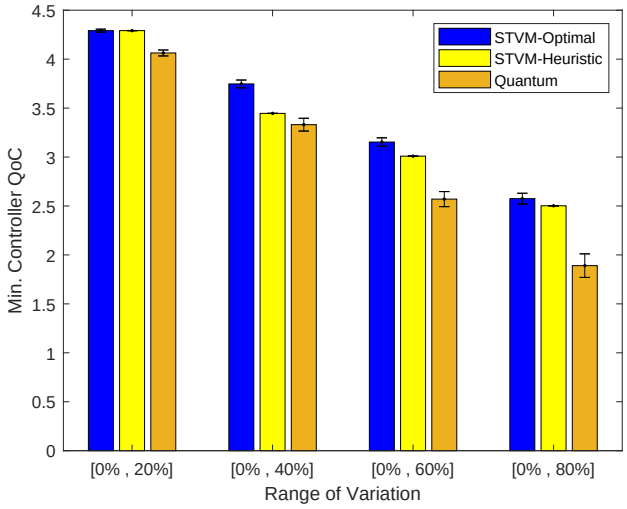


Figure 8: Minimum QoC vs. Range of Variation

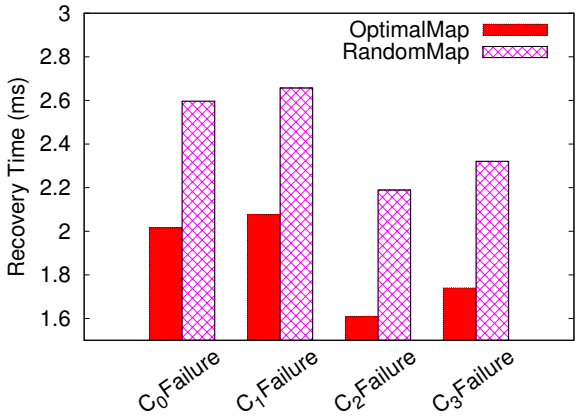


Figure 9: Mininet Emulation: Recovery time per controller failure

the switch, the recovery time is faster as compared to the master-slave approach by by 27.3% (Fig. 9). The reason for the faster recovery time is because in the case of the master-slave approach, upon detection of controller failure, messages are exchanged between the switch and the controller to change the role of the slave controller to the master controller (upon failure of the master controller) while there is no role change in the case of the multiple mapping approach that we have considered in our work.

VIII. Conclusion

Switch-controller mapping is an important aspect of SDN based networks with distributed controller architecture. Two critical factors that need to be taken into account while mapping the switches to the controllers are the reliability of the controller and the response time to flow-setup requests from the switches. In our work, we first developed a reliability model for the controller based on the Bayesian inference approach and a linear approximation of the controller's response time model. We then proposed a Quality of Controller (QoC) metric that considers both controller reliability and response time and developed an optimization problem to find

the switch-controller mapping that maximizes the minimum of QoC of the controllers. We adopted the multiple controller mapping approach to provide further resilience in the event of a controller failure. Owing to the computation complexity of the optimization program, we propose a heuristic algorithm, *Quantum*, to obtain a mapping solution in reasonable time. We compared the performance of *Quantum* against the optimal solution and a greedy heuristic for a small topology and found that *Quantum*'s mapping solution achieved within 18% of the minimum QoC obtained by the optimal solution and outperformed the baseline heuristic for both small and large topologies, while achieving better average response time, average reliability and fairness in performance as compared to the baseline heuristic. We compared the performance of *Quantum* against other approaches that focus on mapping to improve controller response time and reliability respectively and show that *Quantum* is more effective in achieving better minimum QoC. We also presented a heuristic algorithm to manage the switch-controller mapping during short term traffic variation and demonstrate its effectiveness in the performance study by comparing it with the optimal solution.

ACKNOWLEDGMENT

This work was supported by Singapore MoE AcRF Tier-2 Grant MOE2015-T2-2-116, NUS WBS R-263-000-C17-112.

REFERENCES

[1] ONF, "OpenFlow Switch Specification," Open Networking Foundation, Tech. Rep. ONF TS-012, Oct. 2013.

[2] S. H. Yeganeh *et al.*, "On scalability of Software Defined Networking," *IEEE Communications Magazine*, vol. 51, no. 2, pp. 136–141, 2013.

[3] A. Tootoonchian and Y. Ganjali, "Hyperflow: A Distributed Control Plane for Openflow," in *INM/WREN'10*, San Jose, USA, Apr. 2010.

[4] T. Y. Cheng, M. Wang, and X. Jia, "QoS-guaranteed Controller Placement in SDN," in *IEEE GLOBECOM 2015*, San Diego, USA, 2015.

[5] T. Wang, F. Liu, J. Guo, and H. Xu, "Dynamic SDN Controller Assignment in Data Center Networks: Stable Matching with Transfers," in *IEEE INFOCOM 2016*, San Francisco, USA, Apr. 2016, pp. 1–9.

[14] Q. Zhong, Y. Wang, W. Li, and X. Qiu, "A min-cover based controller placement approach to build reliable control network in sdn," in *Network Operations and Management Symposium (NOMS), 2016 IEEE/IFIP*. IEEE, 2016, pp. 481–487.

[6] J. Cui, Q. Lu, H. Zhong, M. Tian, and L. Liu, "A load-balancing mechanism for distributed sdn control plane using response time," *IEEE Transactions on Network and Service Management*, vol. 15, no. 4, pp. 1197–1206, Dec 2018.

[7] F. J. Ros and P. M. Ruiz, "Five nines of southbound reliability in software-defined networks," in *Proceedings of the third workshop on Hot topics in software defined networking*. ACM, 2014, pp. 31–36.

[8] J. Liu, J. Liu, and R. Xie, "Reliability-based controller placement algorithm in software defined networking," *Computer Science and Information Systems*, no. 00, pp. 14–14, 2016.

[9] Y. Hu, W. Wang, X. Gong, X. Que, and S. Cheng, "On reliability-optimized controller placement for software-defined networks," *China Communications*, vol. 11, no. 2, pp. 38–54, 2014.

[10] T. Hu, Z. Guo, J. Zhang, and J. Lan, "Adaptive slave controller assignment for fault-tolerant control plane in software-defined networking," in *2018 IEEE International Conference on Communications (ICC)*. IEEE, 2018, pp. 1–6.

[11] Wang, Tao and Liu, Fangming and Xu, Hong, "An efficient online algorithm for dynamic SDN controller assignment in data center networks," *IEEE/ACM Transactions on Networking*, vol. 25, no. 5, pp. 2788–2801, 2017.

[12] A. Filali, A. Kobbane, M. Elmachkour, and S. Cherkaoui, "Sdn controller assignment and load balancing with minimum quota of processing capacity," in *2018 IEEE International Conference on Communications (ICC)*. IEEE, 2018, pp. 1–6.

[13] V. Sridharan, M. Gurusamy, and T. Truong-Huu, "Multi-controller traffic engineering in software defined networks," in *Local Computer Networks (LCN), 2017 IEEE 42nd Conference on*. IEEE, 2017, pp. 137–145.

[15] L. Y. Zhi, P. M. Mohan, V. Sridharan, and M. Gurusamy, "Secondary controller mapping for reliable control traffic forwarding in sdn," in *2018 27th International Conference on Computer Communication and Networks (ICCCN)*, July 2018, pp. 1–2.

[16] Curtis, Andrew R and Mogul, Jeffrey C and Tourrilhes, Jean and Yalagandula, Praveen and Sharma, Puneet and Banerjee, Sujata, "DevoFlow: Scaling flow management for high-performance networks," *ACM SIGCOMM CCR*, vol. 41, no. 4, pp. 254–265, 2011.

[17] K. V. Vishwanath and N. Nagappan, "Characterizing cloud computing hardware reliability," in *Proceedings of the 1st ACM symposium on Cloud computing*. ACM, 2010, pp. 193–204.

[18] FICO Xpress team, "MIP formulations and linearizations," FICO, Tech. Rep., June 2009.

[19] Benson, Theophilus and Anand, Ashok and Akella, Aditya and Zhang, Ming, "Understanding data center traffic characteristics," in *Proceedings of the 1st ACM workshop on Research on enterprise networking*. ACM, 2009, pp. 65–72.

[20] Liu, Fangming and Guo, Jian and Huang, Xiaomeng and Lui, John CS, "eBA: Efficient bandwidth guarantee under traffic variability in datacenters," *IEEE/ACM Transactions on Networking*, vol. 25, no. 1, pp. 506–519, 2016.