ERA: Efficient Request Assignment for Servers in Data Center Networks with SDN

You-Chiun Wang
Department of Computer Science and Engineering
National Sun Yat-sen University
Kaohsiung, Taiwan
ycwang@cse.nsysu.edu.tw

Jun-Fu Zhang
Department of Computer Science and Engineering
National Sun Yat-sen University
Kaohsiung, Taiwan
cgf970126@gmail.com

***Abstract*—*Software-defined networking (SDN)* can facilitate network management by using a controller to monitor the network and instruct switches. It receives attention to manage a *data center network (DCN)* with multiple servers using SDN. In the paper, we propose an *efficient request assignment (ERA)* scheme to allocate requests sent from clients to the servers in an SDN-based DCN. For each request, ERA first finds candidate servers with sufficient resources and grades them. The calculation of grades considers the load degree, processing delay, and resource utilization of each server. Based on its grade, ERA picks a suitable server to handle the request, which can achieve load balance among servers and save time responding to requests. Through simulations, we show that the ERA scheme can efficiently improve the connection rate, response time, and resource utilization of servers.**

***Keywords*—data center network, load balance, request assignment, server, software-defined networking.**

I. INTRODUCTION

To satisfy the rapid increase in the demand for computation, there have been many data centers deployed to support data-intensive services such as web applications, querying, and data analysis. They can be viewed as warehouses that have multiple servers for data processing [1]. A *data center network (DCN)* is a network that connects these servers via switches to provide high-speed communications among them [2].

Software-defined networking (SDN) offers a new paradigm for network management by pulling the control plane away from switches and putting it on a central entity known as the *controller* [3]. In this way, the controller can easily monitor a network and govern switches. OpenFlow is a common protocol used to implement SDN, where switches maintain flow tables to store the controller's commands [4]. The controller builds secure connections with switches and adds flow entries to their flow tables. Each flow entry has match fields to help a switch check if a packet meets specified conditions. If so, the switch routes or drops the packet based on the entry's instructions.

Employing SDN to efficiently manage DCNs has attracted attention, and there are two issues widely discussed. One is *route management*, which considers how to adjust the routes of flows in a DCN to mitigate congestion. Many solutions [5]–[7] let the controller monitor the bandwidth utilization of links and then guide some flows on busy links to alternative paths to share their loads. The other issue is *request assignment*. Given the requests from clients, the issue asks how to allocate them to servers to improve performance (e.g., reducing the response time for requests). This usually involves how to balance loads among servers (the details will be discussed later in Section II).

This paper targets the second issue and proposes an *efficient request assignment (ERA)* scheme for an SDN-based DCN. In ERA, whenever a request arrives, the controller finds candidate servers that have enough CPU and memory resources to deal with the request. Then, it computes a grade for every candidate server, which takes account of the relationship between multiple factors (including the server's processing delay, CPU utilization, and memory utilization) and average response time. When the load on a server is large, its grade is increased by a value as a penalty. Hence, light-load servers are given priority to handle requests, which helps achieve load balance among servers. Simulation results reveal that our ERA scheme can efficiently assign requests to servers and improve the DCN's performance in terms of connection rate, response time, and resource utilization.

This paper is organized as follows: Section II surveys related work, and Section III gives the system model. Then, we detail the ERA scheme in Section IV and evaluate performance in Section V. Finally, Section VI concludes this paper and gives future work.

II. RELATED WORK

With load-balancing consideration, several studies [8]–[11] carry out the round-robin strategy for request assignment in SDN-based DCNs, where each server copes with a request in turn. This strategy is easy to implement and can theoretically balance loads of servers. However, when requests have diverse demands on resources, some servers that are processing large-demand requests still have their turn to handle other requests. In this situation, their processing delays will worsen, thereby prolonging the response time for requests.

Besides the round-robin strategy, the work [12] implements the bandwidth-based approach, which assesses the bandwidth of each server for a fixed time interval (e.g., 14 s) and assigns the request to the server with the least bandwidth. Bhat et al. [13] choose the server with the fewest connections and build the shortest path between the client and the selected server via Dijkstra's algorithm. In [14], fuzzy logic is applied to select a server with the minimum load (based on its CPU, memory,

TABLE I
SUMMARY OF NOTATIONS.

notations	definitions
$\hat{S}$	set of all servers in the DCN
$\hat{S}_{\mathbf{C}}$	set of candidate servers to handle a request
$\varepsilon_i^{\text{CPU}}$, $\varepsilon_i^{\text{MEM}}$	CPU and memory resource capacities of a server s_i
u_i^{CPU}, u_i^{MEM}	CPU and memory resources used in s_i
z_i^{CPU}, z_i^{MEM}	s_i's z-scores on CPU and memory utilization
r_j^{CPU}, r_j^{MEM}	CPU and memory resources asked by a request λ_j
g_i	grade of s_i
$\tilde{D}_i$	normalized processing delay of s_i
$\tilde{P}_i$	CPU utilization of s_i ($\tilde{P}_i = u_i^{\text{CPU}}/\varepsilon_i^{\text{CPU}}$)
$\tilde{M}_i$	memory utilization of s_i ($\tilde{M}_i = u_i^{\text{MEM}}/\varepsilon_i^{\text{MEM}}$)
$\tilde{T}_i$	normalized response time of s_i
μ, σ	average and standard deviation

and bandwidth utilization), and the request is then delivered to that server. The work [15] combines the least-connection and weighted round-robin methods to select a server to handle each arriving request. In [16], a dynamic feedback load balancing method is proposed, which considers server load, service time, and line delays when assigning requests to servers. For each server, the study [17] computes a ratio of response time and weight (decided by server specifications). Afterward, requests are forwarded to the server whose ratio is the lowest.

Compared to existing solutions, our work contemplates not only the load degree but also the relationship between multiple factors (e.g., processing delay and resource utilization) and the response time of each server. So, we can balance the load on servers and also save their response time.

III. System Model

We consider an SDN-based DCN composed of a controller, switches, and a set $\hat{S}$ of servers. The controller takes charge of both network coordination and request assignment. Switches support OpenFlow and forward requests (sent from clients) to servers based on the controller's instructions. Servers process requests and then send responses to clients. All servers share a public IP address, so clients are not allowed to specify which servers handle their requests. Hence, the controller can assign requests to servers according to their conditions (e.g., loads).

Let $\varepsilon_i^{\text{CPU}}$ and $\varepsilon_i^{\text{MEM}}$ be the capacities of the CPU and memory resources of every server $s_i \in \hat{S}$, where $\varepsilon_i^{\text{CPU}}$ and $\varepsilon_i^{\text{MEM}} \in \mathbb{Z}^+$ [18]. Besides, we denote by u_i^{CPU} and u_i^{MEM} the number of CPU and memory resources used in s_i, where $0 \leq u_i^{\text{CPU}} \leq \varepsilon_i^{\text{CPU}}$ and $0 \leq u_i^{\text{MEM}} \leq \varepsilon_i^{\text{MEM}}$. When a client wants to issue a request $\lambda_j = (r_j^{\text{CPU}}, r_j^{\text{MEM}})$, where r_j^{CPU} and r_j^{MEM} indicate the number of CPU and memory resources required, it sends λ_j to the public IP address. In this case, the controller selects a server, say, s_i, from $\hat{S}$ such that $\varepsilon_i^{\text{CPU}} - u_i^{\text{CPU}} \geq r_j^{\text{CPU}}$ and $\varepsilon_i^{\text{MEM}} - u_i^{\text{MEM}} \geq r_j^{\text{MEM}}$ (that is, s_i has enough resources to handle λ_j). Then, s_i makes a connection with the client (via TCP three-way handshaking) and handles λ_j. After processing λ_j, s_i sends a response to the client (using the public IP address) and breaks the connection.

Given requests from clients, our problem is how to assign them to servers in $\hat{S}$ to improve the DCN's performance. Three performance metrics, including the *connection rate*, *response time*, and *resource utilization*, are used. The connection rate is defined by the number of connections established per second. The response time of a server s_i for a request λ_j is defined by the amount of time from when s_i receives λ_j to when s_i sends the response to the requesting client (including the amount of time used to build and terminate the connection). Then, the resource utilization for CPU and memory of s_i is measured by $u_i^{\text{CPU}}/\varepsilon_i^{\text{CPU}}$ and $u_i^{\text{MEM}}/\varepsilon_i^{\text{MEM}}$, respectively. Table I summarizes the notations used in this paper.

Algorithm 1: The ERA Scheme

```
   Input: request $\lambda_j = (r_j^{\text{CPU}}, r_j^{\text{MEM}})$
   Output: server $s_i$ to handle $\lambda_j$
1  $\hat{S}_{\mathbf{C}} \leftarrow \hat{S}$;
2  foreach $s_i \in \hat{S}_{\mathbf{C}}$ do
3      if $\varepsilon_i^{\text{CPU}} - u_i^{\text{CPU}} < r_j^{\text{CPU}}$ or $\varepsilon_i^{\text{MEM}} - u_i^{\text{MEM}} < r_j^{\text{MEM}}$ then
4          $\hat{S}_{\mathbf{C}} \leftarrow \hat{S}_{\mathbf{C}} \setminus \{s_i\}$;
5  if $\hat{S}_{\mathbf{C}} = \emptyset$ then
6      return null;
7  foreach $s_i \in \hat{S}_{\mathbf{C}}$ do
8      Compute $s_i$'s z-scores $z_i^{\text{CPU}}$ and $z_i^{\text{MEM}}$;
9      if $\max\{z_i^{\text{CPU}}, z_i^{\text{MEM}}\} > \delta$ then
10         mark $s_i$ as an HL server;
11 foreach $s_i \in \hat{S}_{\mathbf{C}}$ do
12     Compute $s_i$'s grade $g_i$;
13     if $s_i$ is an HL server then
14         $g_i \leftarrow g_i + \xi$;
15 return $\arg\min_{s_i \in \hat{S}_{\mathbf{C}}} g_i$;
```

IV. The Proposed ERA Scheme

Algorithm 1 shows the pseudocode of ERA, which helps the controller pick a suitable server to handle each arriving request λ_j. In line 1, $\hat{S}_{\mathbf{C}}$ is the set of candidate servers. Initially, $\hat{S}_{\mathbf{C}}$ is set to $\hat{S}$ (i.e., all servers in the DCN). Then, the for-loop in lines 2–4 removes the servers in $\hat{S}_{\mathbf{C}}$ without enough CPU or memory resources to process λ_j. In the case that no servers can handle λ_j because of insufficient resources (i.e., $\hat{S}_{\mathbf{C}} = \emptyset$), ERA returns a null value, as shown in lines 5 and 6.

Then, the code in lines 7–10 checks whether some servers in the candidate set $\hat{S}_{\mathbf{C}}$ are *high-load (HL)* servers. Though these servers have enough resources to deal with λ_j, we should avoid using them due to load-balancing considerations. In particular, we compute the z-scores z_i^{CPU} and z_i^{MEM} of each server $s_i \in \hat{S}_{\mathbf{C}}$ in terms of its CPU and memory utilization. Once z_i^{CPU} or z_i^{MEM} exceed a threshold δ (i.e., $\max\{z_i^{\text{CPU}}, z_i^{\text{MEM}}\} > \delta$), s_i is treated as an HL server. How to compute z-scores and decide threshold δ will be discussed in Section IV-A.

The code in lines 11–14 calculates a grade g_i for each server s_i in $\hat{S}_{\mathbf{C}}$. These grades are used to select a server to handle λ_j. As mentioned earlier, we should avoid selecting HL servers. Thus, the grade of each HL server will be increased by a value

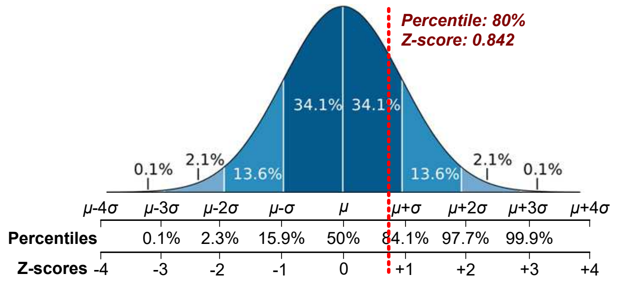


Fig. 1. The relationship between a normal distribution and z-scores.

ξ as a penalty, as indicated in lines 13 and 14. How to decide the grades of servers will be detailed in Section IV-B.

Among all candidate servers in $\hat{S}_{\mathbf{C}}$, we select the one with the smallest grade to handle λ_j, as line 15 shows. Section IV-C discusses the rationale of our ERA scheme.

A. Judgment of HL Servers Using Z-Scores

The controller uses a *server condition table* to store information about servers. For each server s_i in the DCN, there is a record $(s_i, d_i, u_i^{\mathrm{CPU}}/\varepsilon_i^{\mathrm{CPU}}, u_i^{\mathrm{MEM}}/\varepsilon_i^{\mathrm{MEM}}, t_i)$ in the table, where d_i is s_i's average processing delay for requests and t_i denotes the average response time of s_i. Here, whenever a client builds a connection with s_i, we can measure the amount of time taken by s_i to handle the request and update d_i and t_i accordingly.

With this table, we compute the average CPU and memory utilization of servers, as denoted by μ_{CPU} and μ_{MEM}:

$$\mu_{\mathrm{CPU}} = \frac{1}{|\hat{S}_{\mathbf{C}}|}\sum_{\forall s_i\in\hat{S}_{\mathbf{C}}}\frac{u_i^{\mathrm{CPU}}}{\varepsilon_i^{\mathrm{CPU}}},\quad \mu_{\mathrm{MEM}} = \frac{1}{|\hat{S}_{\mathbf{C}}|}\sum_{\forall s_i\in\hat{S}_{\mathbf{C}}}\frac{u_i^{\mathrm{MEM}}}{\varepsilon_i^{\mathrm{MEM}}}. \quad (1)$$

Besides, we calculate the standard deviation of their CPU and memory utilization (denoted by σ_{CPU} and σ_{MEM}):

$$\sigma_{\mathrm{CPU}} = \sqrt{\left(\textstyle\sum_{\forall s_i\in\hat{S}_{\mathbf{C}}}\left(u_i^{\mathrm{CPU}}/\varepsilon_i^{\mathrm{CPU}} - \mu_{\mathrm{CPU}}\right)^2\right)/|\hat{S}_{\mathbf{C}}|},$$
$$\sigma_{\mathrm{MEM}} = \sqrt{\left(\textstyle\sum_{\forall s_i\in\hat{S}_{\mathbf{C}}}\left(u_i^{\mathrm{MEM}}/\varepsilon_i^{\mathrm{MEM}} - \mu_{\mathrm{MEM}}\right)^2\right)/|\hat{S}_{\mathbf{C}}|} \quad (2)$$

Then, z-scores z_i^{CPU} and z_i^{MEM} of CPU and memory utilization on a server $s_i \in \hat{S}_{\mathbf{C}}$ are defined by

$$z_i^{\mathrm{CPU}} = \frac{u_i^{\mathrm{CPU}}/\varepsilon_i^{\mathrm{CPU}} - \mu_{\mathrm{CPU}}}{\sigma_{\mathrm{CPU}}},\quad z_i^{\mathrm{MEM}} = \frac{u_i^{\mathrm{MEM}}/\varepsilon_i^{\mathrm{MEM}} - \mu_{\mathrm{MEM}}}{\sigma_{\mathrm{MEM}}}. \quad (3)$$

To decide threshold δ, we refer to the *Pareto principle* (also called the 80/20 rule), where for many outcomes, about 80% of the consequences come from 20% of the causes [19]. If the resource utilization (i.e., $u_i^{\mathrm{CPU}}/\varepsilon_i^{\mathrm{CPU}}$ or $u_i^{\mathrm{MEM}}/\varepsilon_i^{\mathrm{MEM}}$) of a server s_i is greater than that of 80% of the servers in $\hat{S}_{\mathbf{C}}$, s_i is an HL server. Fig. 1 illustrates the relationship between a normal distribution and z-scores. Here, the z-score is 0.842 for the 80 percentiles in the distribution, so we suggest setting $\delta = 0.842$.

B. Calculation of Grades for Servers

To select a suitable server to process each arriving request, we take account of not only the resource utilization of servers but also their processing delays. Here, the controller calculates grades for servers using TOPSIS (standing for the *technique for order preference by similarity to an ideal solution*), which is a multi-factor decision analysis approach [20]. Specifically, TOPSIS compares a set of alternatives, normalizes grades for factors, and calculates the geometric distance between each alternative and the ideal one (i.e., with the best grade in every factor). It then chooses an alternative that can allow tradeoffs between different factors, where the bad result in one factor may be neutralized by the good result in another factor.

For each server $s_i \in \hat{S}_{\mathbf{C}}$, its grade g_i is computed as follows:

$$g_i = w_1 \times \tilde{D}_i + w_2 \times \tilde{P}_i + w_3 \times \tilde{M}_i, \quad (4)$$

where $\tilde{D}_i$ is s_i's normalized processing delay, $\tilde{P}_i = u_i^{\mathrm{CPU}}/\varepsilon_i^{\mathrm{CPU}}$ (i.e., CPU utilization), and $\tilde{M}_i = u_i^{\mathrm{MEM}}/\varepsilon_i^{\mathrm{MEM}}$ (i.e., memory utilization). For normalization, we set $\tilde{D}_i = d_i/\max_{\forall s_j\in\hat{S}_{\mathbf{C}}} d_j$.

To decide the weight w_k ($k = 1, 2,$ or 3) in Eq. (4), we adopt the *Pearson correlation coefficient (PCC)*, as denoted by $r_{x,y}$, which is used to assess the strength of the relationship between two variables x and y [21]. More concretely, we set

$$w_k = \frac{|r_{x,y}|}{\max\{1 - |r_{x,y}|, \varphi\}},\quad k = 1, 2, 3, \quad (5)$$

where x is the processing delay (i.e., $\tilde{D}_i$), CPU utilization (i.e., $\tilde{P}_i$), and memory utilization (i.e., $\tilde{M}_i$) when k is 1, 2, and 3, respectively. Moreover, y is the response time. In other words, we contemplate the relationship between each factor in Eq. (4) and the response time. In Eq. (5), φ is a small positive value (e.g., $\varphi = 0.1$). Then, PCC is calculated by

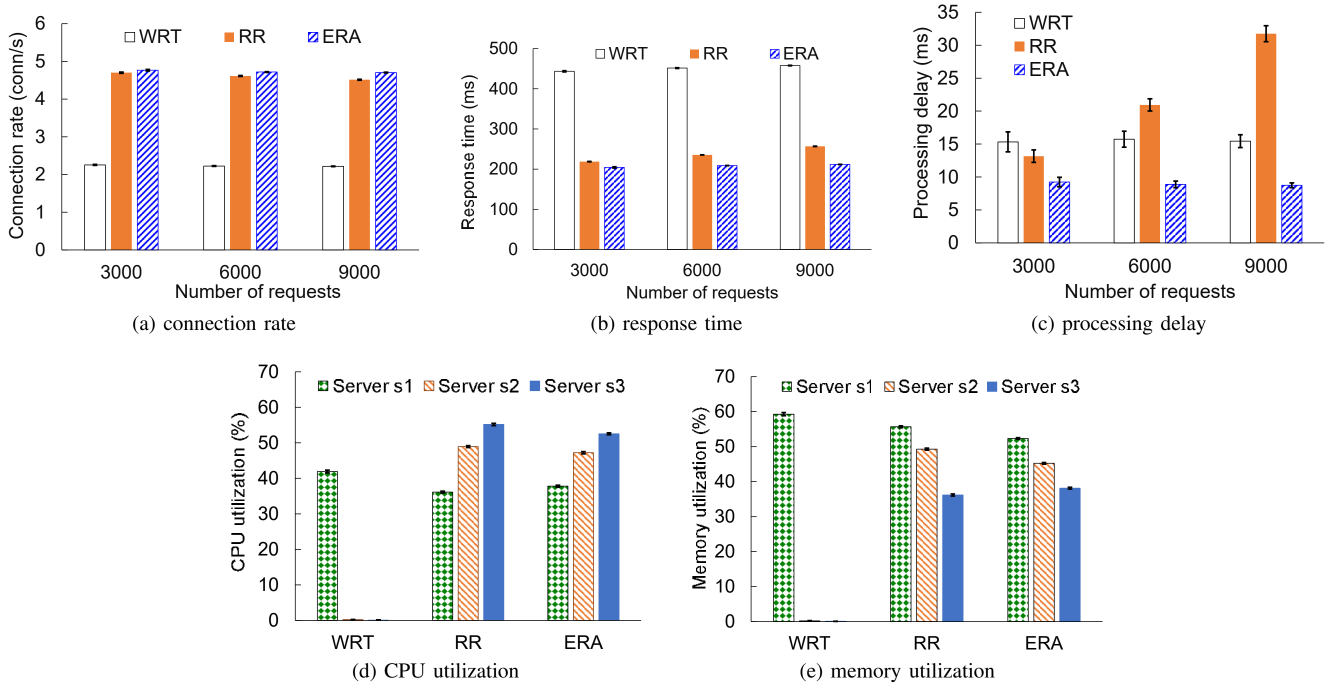
$$r_{x,y} = \begin{cases} \frac{\sum_{\forall s_i\in\hat{S}_{\mathbf{C}}}(\tilde{D}_i - \mu_{\mathrm{DLY}})(\tilde{T}_i - \mu_{\mathrm{RES}})}{\sigma_{\mathrm{DLY}}\times\sigma_{\mathrm{RES}}} & k = 1 \\ \frac{\sum_{\forall s_i\in\hat{S}_{\mathbf{C}}}(\tilde{P}_i - \mu_{\mathrm{CPU}})(\tilde{T}_i - \mu_{\mathrm{RES}})}{\sigma_{\mathrm{CPU}}\times\sigma_{\mathrm{RES}}} & k = 2 \\ \frac{\sum_{\forall s_i\in\hat{S}_{\mathbf{C}}}(\tilde{M}_i - \mu_{\mathrm{MEM}})(\tilde{T}_i - \mu_{\mathrm{RES}})}{\sigma_{\mathrm{MEM}}\times\sigma_{\mathrm{RES}}} & k = 3, \end{cases} \quad (6)$$

where $\tilde{T}_i$ is the normalized response time of a server s_i ($\tilde{T}_i = t_i/\max_{\forall s_j\in\hat{S}_{\mathbf{C}}} t_j$), μ_{RES} is the average (normalized) response time of servers in $\hat{S}_{\mathbf{C}}$, and σ_{RES} is the standard deviation of the normalized response time of servers in $\hat{S}_{\mathbf{C}}$. In addition, μ_{DLY} and σ_{DLY} represent the average and standard deviation of the normalized processing delays of all servers in $\hat{S}_{\mathbf{C}}$, respectively.

The value of PCC (i.e., $r_{x,y}$) in Eq. (6) is within $[-1, 1]$. When $r_{x,y}$ is closer to 1 or -1, there will be a stronger positive or negative linear correlation between x and y. In this situation, the corresponding factor (i.e., $\tilde{D}_i$, $\tilde{P}_i$, or $\tilde{M}_i$) is given a larger weight w_k by Eq. (5). As $1-|r_{x,y}|$ may be equal to zero (when $r_{x,y} = \pm 1$), we add a positive constant φ to the denominator to ensure the correctness of Eq. (5).

C. Discussion on Rationale

In ERA, when a request λ_j arrives, a candidate set $\hat{S}_{\mathbf{C}}$ is first built by removing those servers from $\hat{S}$ whose resources cannot meet λ_j's demand $(r_j^{\mathrm{CPU}}, r_j^{\mathrm{MEM}})$. Based on z-scores on CPU and memory utilization, we partition the servers in $\hat{S}_{\mathbf{C}}$ into two groups: HL and non-HL servers. Then, we calculate a grade for each server in $\hat{S}_{\mathbf{C}}$ using TOPSIS. Through PCC, we measure how each factor (including the processing delay, CPU



(a) connection rate (b) response time (c) processing delay (d) CPU utilization (e) memory utilization

Fig. 2. Performance comparison of different methods.

utilization, and memory utilization) affects the result (i.e., the response time) and assign an appropriate weight to reflect its effect using Eq. (5). Since these factors have a negative impact on the result, ERA chooses the server with the minimum grade (referring to Eq. (4) and line 15 in Algorithm 1) to handle λ_j to reduce the response time.

The purpose of dividing $\hat{S}_{\rm C}$ is for load-balancing consideration. To avoid allocating many requests to a few servers, non-HL servers (with relatively low CPU and memory utilization) are given precedence over HL ones to handle requests. Instead of simply excluding HL servers from $\hat{S}_{\rm C}$, we add a value ξ to the grade of each HL server (i.e., line 14 in Algorithm 1). Doing so allows HL servers with *good conditions* to partake in request assignment. The resource utilization of these servers may be slightly higher than others, making them HL servers (due to high z-scores). However, they have shorter processing delays, which results in lower TOPSIS grades. In this case, we can ask them to help handle requests to improve performance.

V. Performance Evaluation

To evaluate system performance, we simulate an SDN-based DCN by using Mininet [22]. The controller and switches are implemented by Ryu [23] and Open vSwitch [24], respectively. The DCN contains three servers, s_1, s_2, and s_3, whose CPU and memory capacities are set by $(\varepsilon_1^{\rm CPU}, \varepsilon_1^{\rm MEM}) = (900, 600)$, $(\varepsilon_2^{\rm CPU}, \varepsilon_2^{\rm MEM}) = (700, 700)$, and $(\varepsilon_3^{\rm CPU}, \varepsilon_3^{\rm MEM}) = (600, 900)$. The above configuration allows us to imitate a general scenario in which servers possess diverse resource capacities. All servers share a public IP address 10.0.0.100. Moreover, there are five clients to generate requests, which is done through the httperf tool [25]. As mentioned in Section III, clients send requests to the public IP address, and the controller assigns these requests to servers. When assigning a client's request to a server, the server makes a connection with the client, handles the request, sends a response to the client, and terminates the connection. On getting a response, each client issues the next request right away. In this way, we can stress-test each method.

We consider two types of resource demands. A *large demand (LD)* requires 280–320 units of resources, while a *small demand (SD)* needs 140–160 units of resources. Then, there are four combinations of requests: LD CPU with LD memory, LD CPU with SD memory, SD CPU with LD memory, and SD CPU with SD memory. Each combination of requests accounts for 25% of the overall requests. There will be 3000, 6000, and 9000 requests generated by clients in total.

We compare our ERA scheme with two methods discussed in Section II: *weighted response time (WRT)* [17] and *round-robin (RR)* [11]. In the WRT method, requests are forwarded to the server with the lowest ratio of response time and weight. The RR method asks servers to handle requests in turn. Each experiment is repeated 100 times, and we take their average and also the 95% confidence interval.

Fig. 2(a) compares the connection rates of different methods. A higher connection rate means that more requests can be handled per second. This rate slightly decreases as the number of requests grows. WRT updates the response time of a server only when a connection is built. Once the response time of a server in a connection is too long, it may reduce the probability of selecting that server. Eventually, WRT will assign requests to merely a few servers. That is why WRT's connection rate is much lower than other methods. By using Algorithm 1 to pick

TABLE II
JAIN'S FAIRNESS INDEX OF RESOURCE UTILIZATION ON SERVERS.

resource	WRT	RR	ERA
CPU	0.338	0.972	0.983
memory	0.337	0.971	0.984

servers to deal with requests, ERA can increase the connection rate. As compared to WRT and RR, our ERA scheme improves 111.72% and 2.60% of the connection rate, respectively.

Fig. 2(b) shows the average response time of servers using each method. In particular, the response time increases when there are more requests. Since WRT concentrates most request processing on a few servers (the evidence will be given later), these servers become busy, thereby raising their response time. By assigning requests to servers in a round-robin way, loads are distributed over servers. Thus, RR's response time can be greatly reduced. ERA uses z-scores to distinguish HL servers from others and gives precedence to non-HL servers to handle requests. Hence, ERA can further decrease the response time. Compared with WRT and RR, the ERA scheme saves 53.81% and 12.10% of the response time, respectively.

Fig. 2(c) presents the average processing delay for requests. For a server, the response time is the sum of the time to build a connection with a client, the processing delay, the time to send a response to the client, and the time to cut off the connection. Thus, the processing delays in Fig. 2(c) are much lower than the response time in Fig. 2(b). From Fig. 2(c), we observe that the processing delay of RR significantly rises as the number of requests grows. That is because RR lets servers handle requests in turn without considering the amount of time consumed by a server to deal with each request. By contrast, our ERA scheme computes a grade for each server using both TOPSIS and PCC, which takes account of the resource utilization and processing delays of servers. In this way, ERA maintains a low processing delay for requests. More concretely, the ERA scheme reduces 42.25% and 59.22% of the processing delay as compared with the WRT and RR methods, respectively.

In Fig. 2(d) and (e), we give the average CPU and memory utilization of each server, where the number of total requests is 9000. Since $\varepsilon_1^{\text{CPU}} > \varepsilon_2^{\text{CPU}} > \varepsilon_3^{\text{CPU}}$, the order of servers in terms of CPU utilization is $s_1 < s_2 < s_3$. Because $\varepsilon_1^{\text{MEM}} < \varepsilon_2^{\text{MEM}} < \varepsilon_3^{\text{MEM}}$, the order of servers for memory utilization is $s_1 > s_2 > s_3$. Evidently, WRT makes s_1 handle most requests, as s_1 has a shorter response time due to the largest CPU capacity. This results in imbalanced loads on servers with diverse resource capacities and hurts performance. Table II lists *Jain's fairness index* of CPU and memory utilization in each method. Let u_i be the resource utilization of a server s_i. Given n servers, this index is calculated as follows [26]:

$$J = \frac{\left(\sum_{i=1}^{n} u_i\right)^2}{n \sum_{i=1}^{n} u_i^2}, \tag{7}$$

where $\frac{1}{n} \leq J \leq 1$. The load on servers will be more balanced when J is larger. As can be seen, both RR and ERA can assign requests to servers more fairly to achieve load balance.

VI. Conclusion and Future Work

This paper proposes the ERA scheme to assign requests to servers in an SDN-based DCN. For each request, the controller finds a candidate set of servers with enough resources to satisfy the request's demand. Then, it distinguishes HL servers using z-scores and calculates a grade for every server via TOPSIS and PCC. When calculating grades, we consider not only HL servers but also the relationship between processing delays, resource utilization, and response times of servers. Therefore, ERA can pick a suitable server to deal with the request. Using Mininet to simulate a DCN that contains servers with diverse resource capacities, we demonstrate that our ERA scheme can efficiently improve the connection rate, decrease the response time, and balance resource utilization on servers as compared with both WRT and RR methods.

In this work, we exploit the controller to efficiently assign requests to servers. Recently, the technique of P4, which stands for *programming protocol-independent packet processors*, has been proposed. Specifically, P4 is a domain-specific language developed for network devices such as switches that allows users to specify how these devices process each packet [27]. Regarding future work, we will consider using P4 switches to carry out request assignments. Since servers may connect with different switches, this issue is more challenging as requests may be assigned to servers in a distributed manner.

Acknowledgment

This work was supported by National Science and Technology Council, Taiwan under Grant 111-2221-E-110-023-MY2.

References

[1] K. Wang, Q. Zhou, S. Guo, and J. Luo, "Cluster frameworks for efficient scheduling and resource allocation in data center networks: A survey," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 4, pp. 3560–3580, 2018.

[2] Y. C. Wang and T. J. Hsiao, "URBM: User-rank-based management of flows in data center networks through SDN," in *IEEE International Conference on Computer Communication and the Internet*, 2022, pp. 142–149.

[3] H. Farhady, H. Y. Lee, and A. Nakao, "Software-defined networking: A survey," *Computer Networks*, vol. 81, pp. 79–95, 2015.

[4] Y. C. Wang and H. Hu, "An adaptive broadcast and multicast traffic cutting framework to improve Ethernet efficiency by SDN," *Journal of Information Science and Engineering*, vol. 35, no. 2, pp. 375–392, 2019.

[5] Y. L. Lan, K. Wang, and Y. H. Hsu, "Dynamic load-balanced path optimization in SDN-based data center networks," in *International Symposium on Communication Systems, Networks and Digital Signal Processing*, 2016, pp. 1–6.

[6] Y. C. Wang and S. Y. You, "An efficient route management framework for load balance and overhead reduction in SDN-based data center networks," *IEEE Transactions on Network and Service Management*, vol. 15, no. 4, pp. 1422–1434, 2018.

[7] Y. Liu, H. Gu, Z. Zhou, and N. Wang, "RSLB: Robust and scalable load balancing in software-defined data center networks," *IEEE Transactions on Network and Service Management*, vol. 19, no. 4, pp. 4706–4720, 2022.

[8] K. A. Jadhav, M. M. Mulla, and D. G. Narayan, "An efficient load balancing mechanism in software defined networks," in *International Conference on Computational Intelligence and Communication Networks*, 2020, pp. 116–122.

[9] Y. A. H. Omer, M. A. Mohammedel-Amin, and A. B. A. Mustafa, "Load balance in cloud computing using software defined networking," in *International Conference on Computer, Control, Electrical, and Electronics Engineering*, 2021, pp. 1–6.

[10] V. Kumar, S. Jangir, and D. G. Patanvariya, “Traffic load balancing in SDN using round-robin and Dijkstra based methodology,” in *International Conference for Advancement in Technology*, 2022, pp. 1–4.

[11] I. T. Singh, T. R. Singh, and T. Sinam, “Server load balancing with round robin technique in SDN,” in *International Conference on Decision Aid Sciences and Applications*, 2022, pp. 503–505.

[12] A. K. Arahunashi, G. G. Vaidya, S. Neethu, and K. V. Reddy, “Implementation of server load balancing techniques using software-defined networking,” in *International Conference on Computational Systems and Information Technology for Sustainable Solutions*, 2018, pp. 87–90.

[13] B. R. Bhat, N. S. Sneha, K. Bhat, C. C. Kamath, and C. Naik, “Improving the efficiency of software defined network through load balancing algorithms,” in *International Conference on Intelligent Communication Technologies and Virtual Mobile Networks*, 2021, pp. 124–131.

[14] I. A. Prakoso, S. N. Hertiana, and F. Dewanta, “Analysis of fuzzy logic algorithm for load balancing in SDN,” in *International Seminar on Research of Information Technology and Intelligent Systems*, 2021, pp. 401–406.

[15] T. G. Thajeel and A. Abdulhassan, “A hybrid load balancing scheme for software defined networking,” in *Information Technology To Enhance e-learning and Other Application*, 2021, pp. 106–112.

[16] N. S. Prodanov, K. S. Nikolova, and D. K. Atamian, “Load balancing implementation in software defined networks,” in *International Scientific Conference on Information, Communication and Energy Systems and Technologies*, 2022, pp. 1–4.

[17] H. Nurwasito and R. Rahmawati, “Weighted response time algorithm for web server load balancing in software defined network,” in *International Conference on Electronics Representation and Algorithm*, 2021, pp. 143–148.

[18] Y. C. Wang and S. H. Wu, “Efficient deployment of virtual network functions to achieve load balance in cloud networks,” in *Asia-Pacific Network Operations and Management Symposium*, 2022, pp. 1–6.

[19] H. Zhu, “Social development paradox: An E-CARGO perspective on the formation of the Pareto 80/20 distribution,” *IEEE Transactions on Computational Social Systems*, vol. 9, no. 5, pp. 1297–1306, 2022.

[20] H. Yin, X. R. Li, and Y. Gao, “Relative Euclidean distance with application to TOPSIS and estimation performance ranking,” *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 52, no. 2, pp. 1052–1064, 2022.

[21] G. Li, A. Zhang, Q. Zhang, D. Wu, and C. Zhan, “Pearson correlation coefficient-based performance enhancement of broad learning system for stock price prediction,” *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 69, no. 5, pp. 2413–2417, 2022.

[22] Mininet. [Online]. Available: http://mininet.org

[23] Ryu. [Online]. Available: https://ryu-sdn.org

[24] Open vSwitch (OVS). [Online]. Available: https://www.openvswitch.org

[25] httperf. [Online]. Available: https://github.com/httperf/httperf

[26] Y. C. Wang and D. R. Jhong, “Efficient allocation of LTE downlink spectral resource to improve fairness and throughput,” *International Journal of Communication Systems*, vol. 30, no. 14, pp. 1–13, 2017.

[27] Y. C. Wang and P. Y. Su, “Collaborative defense against hybrid network attacks by SDN controllers and P4 switches,” *IEEE Transactions on Network Science and Engineering*, early access, Oct. 2023, doi: 10.1109/TNSE.2023.3324329.