Received: 31 March 2018 | Revised: 19 April 2019 | Accepted: 23 May 2019

DOI: 10.1002/dac.4065

RESEARCH ARTICLE

WILEY

Proactive multipath routing with a predictive mechanism in software-defined networks

Ying-Dar Lin[1] | Te-Lung Liu[2] | Shun-Hsien Wang[1] | Yuan-Cheng Lai[3]

[1]Department of Computer Science, National Chiao Tung University, Hsinchu, Taiwan

[2]National Center for High Performance Computing, Tainan, Taiwan

[3]Department of Information Management, National Taiwan University of Science and Technology, Taipei, Taiwan

Correspondence
Te-Lung Liu, National Center for High Performance Computing, Tainan, Taiwan.
Email: tlliu@narlabs.org.tw

Summary

With the growth of network traffic volume, link congestion cannot be avoided efficiently with conventional routing protocols. By utilizing the single shortest-path routing algorithm from link state advertisement information, standard routing protocols lack of global awareness and are difficult to be modified in a traditional network environment. Recently, software-defined network (SDN) provided innovative architecture for researchers to program their own network protocols. With SDN, we can divert heavy traffic to multiple paths in order to resolve link congestion. Furthermore, certain network traffics come in periodic fashion such as peak hours at working days so that we can leverage forecasting for resource management to improve its performance. In this paper, we propose a proactive multipath routing with a predictive mechanism (PMRP) to achieve high-performance congestion resolution. PMRP has two main concepts: (a) a proactive mechanism where PMRP deploys M/M/1 queue and traffic statistics to simulate weighted delay for possible combinations of multipaths placement of all subnet pairs, and leverage genetic algorithm for accelerating selection of optimized solution, and (b) a predictive mechanism whereby PMRP uses exponential smoothing for demand traffic volumes and variance predictions. Experimental results show a 49% reduction in average delay as compared with single shortest routing, and a 16% reduction in average delay compared with utilization & topology-aware multipath routing (UTAMP). With the predictive mechanism, PMRP can decrease an additional 20% average delay. Furthermore, PMRP reduces 93% of flow table usage on average as compared with UTAMP.

KEYWORDS

congestion resolution, flow table usage, multipath routing, SDN

1 | INTRODUCTION

As an increasing variety of network applications are now available, the volume of traffic grows rapidly, and link congestions cannot be efficiently avoided with conventional interior gateway routing protocols. In the traditional network environment, routers operate in a distributed fashion, and each of them has only a partial view of network status. Consequently, problems such as unbalance of resource allocation and performance degradation arise. Standard routing protocols adopt single shortest-path algorithm for routing selection, and hence the network traffic may block the trunk

links. Furthermore, periodically updating the link weights provided by link state advertisement is insensitive for network congestions. In addition to congestion resolution, there are no forecasting functions for resource management. The network pattern also has locality phenomenon such as daily rush hours so that we can leverage it for bandwidth allocation in routing mechanism. However, it is difficult to modify standard routing protocols in conventional distributed network architecture, and it is impossible to replace every individual commodity router to provide better congestion control algorithms.

Recent work on software-defined network (SDN) architecture[1] separates the control plane from data plane and provides programmable Application Programming Interface (API) for provisioning and management. With a centralized controller, a global view of network topology with status can be obtained, and researchers could implement network protocols easily in the SDN environment. However, flow tables in SDN switches are limited resources, and care has to be taken with use of flow entries. If we design a routing management mechanism according to host pairs, the number of flow entries will increase dramatically and will rapidly exceed the limits of the hardware. One would consequently like to manage the flows based on subnet pairs for better flow table utilization. For example, for two subnet domains with 200 hosts in each domain, there will be up to 80 000 interdomain flow entries if the routing is managed by host pairs and only two flow entries by subnet pairs. Recently, several studies on SDN routing mechanisms for congestion resolution have used multipath routing to achieve load balancing for better network utilization. Nevertheless, out-of-order forwarding and flow table usage remain important issues to be resolved. There are also works that adopt predictive mechanism for future traffic loads or network performance for congestion avoidance. These may similarly suffer from the same scalability issues.

In this paper, we propose proactive multipath routing with a predictive mechanism (PMRP) in the SDN environment for congestion resolution. PMRP measures demand traffic volume of subnets from each egress switch and allocates multipaths for each subnet pair to attain high precision congestion resolution. Because this procedure will lead to exponential complexity, we use the genetic algorithm (GA) to optimize path computation. To deal with the issue of flow table management, PMRP pre-allocates flow entries to decrease RTT delay between control and data planes, where subnet-based routing can greatly decrease flow entry usage. As a predictive mechanism, we deploy exponential smoothing to predict demand traffic volume of each subnet pair, so that pre-allocate mechanism can predict feedback of congestion beforehand.

The remainder of this paper is organized as follows. First, we introduce the related backgrounds in Section 2. The problem statement is outlined in Section 3, and in Section 4 we propose PMRP with system architecture. We record the simulation results in Section 5, and our conclusions and future work are presented in Section 6.

2 | BACKGROUND

In this section, we survey and compare related proposals on congestion resolution using multipath routing and traffic prediction mechanisms, and describe how we utilize the OpenFlow protocol for the proposed PMRP mechanism.

2.1 | Works on congestion resolution using multipath routing

In order to avoid network congestion, some studies separate traffic flows into multiple simultaneous paths for lowering the load on trunk links. As shown in Table 1, we classify these works according to their flow separation concepts. The

TABLE 1 SDN multipath routing proposals

Flow Separation Concept	Name	Path Computation		Out-of-Order	Provisioning Time	Flow Table Usage
		Event Trigger	Proactive Allocation			
QoS argument	Multipath SDN[2]	Flow		No	More	High
	B4[3]	Flow		No	Less	Medium
Traffic rate	DUCE[4]	Congestion		Yes	Less	Low
Available resource	Grooming[5]	Flow		Yes	More	High
	UTAMP/M2SDN[6]	Flow		Yes	More	High
	PMRP		Routine	No	Less	Low

flow could be rerouted into multipaths depending on QoS arguments, traffic rate, or available resources. In these proposals, path computations are event-triggered either by flow or by congestion. Flow-triggered means that when a new flow request arrives, the SDN controller will detect the packet-in event and calculate and allocate path(s) for this connection by issuing flow entries to SDN switches. Consequently, as the request rate increases, the number of flow entries also increases dramatically. Moreover, it takes more end-to-end latency with response time from packet-in to flow modification message. For congestion-triggered solutions, paths are recomputed when network congestions occur so that the excessive traffics will be diverted into other paths. Because the flows will be rerouted into multiple paths, these proposals may be suffered from packet out-of-order problem. When packets of the same traffic flow are transmitted into diverse paths, these packets may not be received in the original order at the destination site due to different latencies of the transmission paths. If the destination site does not contain enough receive buffer, such packet out-of-order problem will cause transmission failures.

Multipath SDN[2] and B4 network[3] separate network flows according to their QoS arguments. A tunnel is created for each host pair with its QoS argument. The packets within the same flow are therefore not routed into different paths to avoid out-of-order problems, and the usage of flow table cannot be controlled by limiting the number of tunnels. Moreover, Pan and Zheng[2] focuses on path selection with regard to QoS arguments and current network status; Jain et al[3] attempts to fully utilize the links among a couple of data centers. Hence, Pan and Zheng[2] will take more provisioning time with higher flow table usage.

DUCE[4] utilizes the OpenFlow meter table for congestion resolution caused by burst flows. When the transmission rate of the flow reaches a threshold, switches will divert the flow to other paths that appear to be congestion-free. Because the procedure is only involved in the data plane, the provisioning speed is very quick. However, the rerouted path is predefined by the control plane, which limits the flexibility of this solution.

Finally, the dynamic traffic grooming algorithm[5] and utilization & topology-aware multipath routing (UTAMP)[6] deploy multipath forwarding based on available network resources. They monitor network resource periodically and allocate paths according to the latest network status when a new packet triggers packet-in event. Both are flow-triggered, and the packets may be received out of order. Consequently, the receiving buffer may overflow which results in performance degradation. Moreover, both consume more flow entries and take more provisioning time because of the end-to-end latency noted above.

There are still related works that improve routing mechanism with different aspect. The CMBLB and CBLB in Hamed et al[7] focus on computing resource load balancing. They distribute the clients' requests among the servers that provide the same service with CPU and memory usage. Such resource management makes great improvement as compared with traditional round-robin and random-based approaches. Wang et al[8] focuses on routing consistency to prevent unexpected behavior like black holes and looping. Moreover, the authors state the coexistence of traditional network and SDN, which provides an alternative of network transformation.

Our proposed PMRP, by contrast, proactively computes multipaths for each subnet pair and periodically updates the routing paths according to available bandwidth. This approach not only decreases flow table usage but also reduces provisioning time. Furthermore, the flow for each host pair will remain in the same path so that there is no out-of-order problem.

2.2 | Works on predictive mechanisms

With the trend of using intercloud applications for big data, traffic flows between data centers are periodically generated. In order to adapt to such traffic environmental changes, current works leverage predictive mechanism for their network management functions, as shown in Table 2. TSDN[9] applies the tensor model for routing path recommendations based on different application types. Depending on current network status and the QoS parameters of the application type, TSDN allocates a corresponding path to improve QoS routing. According to Benson et al,[13] for interdata center communication, 80% of the flows are smaller than 10 KB in size, and most of the bytes are in the top 10% of the flows that we called elephant flows. Elephant Flow Detection[10] and CheetahFlow[11] both predict the elephant flows and reroute them to other paths to prevent congestion. In addition, CheetahFlow also utilizes the support vector machine (SVM) to construct frequent communication pairs and pre-establish their flow entries into switches to decrease provisioning latency. All of the above proposals have scalability issues. TSDN and CheetahFlow manage flows according to host pairs so that flow table usage grows rapidly with an increasing traffic demand. Elephant Flow Detection uses OpenFlow for traffic status monitoring, which will cause additional overheads to the control plane.

TABLE 2 Predictive mechanisms in SDN

Name	Predict Target	Technique	Detection Mechanism	Scalability
TSDN[9]	Application traffic volume	Tensor model	OpenFlow	Low
Elephant Flow Detection[10]	Detect elephant flow	Decision tree	External monitor	Low
CheetahFlow[11]	Detect elephant flow/frequent communication pairs	SVM	OpenFlow	Low
QoS Estimation[12]	Packet loss	Neural network	OpenFlow	High
PMRP	The traffic volume of subnet pairs	Exponential smoothing	OpenFlow	High

QoS Estimation[12] evaluates network performance as new flow enters and feeds back to the network state to improve the learning for lowering the packet loss rate. Our proposed PMRP aims to predict the traffic volume of subnet pairs by a simple exponential smoothing technique and runs faster without scalability problems.

2.3 | SDN and OpenFlow

SDN is an innovative network architecture that separates the control plane from the data plane for more flexibility. The control plane resides in a regular server called controller so that it is easy to deploy new network protocols or management functions with software coding. A Southbound interface is required for the communication between controller and SDN switches. OpenFlow[14] is the general southbound interface standard maintained by Open Networking Foundation (ONF). In OpenFlow, a flow table replaces the traditional routing and forwarding tables that manage network traffic. In addition to basic flow table operation, there are several types of messages for the controller to negotiate with SDN switches. For this proposed PMRP, we apply the multipart message and group table to archive dynamic multipath allocation routing mechanism in the data plane. The multipart message is used to acquire the status of an entire switch, including flow entries, traffic rates, and statistics of its interface. Hence, we can obtain the volume of egress traffic for each subnet pair. For the multipath forwarding function, we use the select function of the group table. First, we define several buckets in the group table. If a packet matches the field specified in the group table, the switch will choose one bucket to forward base on the weight of each bucket. However, the selection algorithm is not specified in OpenFlow and depends on switch implementations. In general, there are two types: packet-based and flow-based. Both distribute traffic to multiple paths based on bucket weights, but flows with the same host pairs will always be forwarded to the same bucket in the flow-based algorithm because of hashed MAC addresses. In our simulation, we deploy Open vSwitch[15] which implements flow-based forwarding obviating an out-of-order packet problem.

Based on the above, we propose the PMRP that anticipates the upcoming traffic load and proactively provisions multipath routing for better network performance with high scalability, based on SDN technology. The details of this system and its implementation are covered in the following sections.

3 | PROBLEM STATEMENT

In this section, we list the notations used in this paper and formulate the problem statement of our PMRP. A simple example is presented that eliminates link congestions by multipath routing.

3.1 | Notation descriptions

Table 3 lists the notations used in PMRP. $TR = [tr_{i,j}]_{n * n}$ is the matrix of traffic volume from i-th to j-th subnet. The number of paths from subnet i to subnet j is $npath_{i,j}$, $path_{i,j,k}$ denotes the k-th path from subnet i to subnet j, and $w_{i,j,k}$ is the weight of $path_{i,j,k}$. For average path delays, $LINK = \{link_{x,y}\}$ is the set of network links, $D^{LINK} = \left\{d^{link}_{x,y}\right\}$ is the set of link delays, $D^{SP} = \left\{d^{sp}_{i,j}\right\}$ denotes the set of average delay of subnet pair i and j, and $D^{P} = \left\{d^{p}_{i,j,k}\right\}$ is the set of delay of specific path k between two subnets i and j. Finally, $ct_{x,y}$ and $mb_{x,y}$ denote the current traffic volume and maximum bandwidth of $link_{x,y}$, respectively.

TABLE 3 Notation descriptions

Category	Term	Description
Index	a,b,i,j	Index of subnets.
	k, l	Index of paths.
	x, y	Index of switches.
	t	Index of time
Entity	n	Number of subnets.
	m	Number of links.
Variable	$G = \{S,LINK\}$	The network topology.
	$S = \{s_x \mid x \geq 1\}$	The set of switches.
	$LINK = \{link_{x,y} \mid x,y \geq 1\}$	The set of links, where $link_{x,y}$ is the link between switch x and y
	$TR = [tr_{i,j} \mid i,j \geq 1]_{n*n}$	The set of traffic volume at current time, where $tr_{i,j}$ is the traffic volume from subnet i to subnet j
	$TR_t = [tr_{i,j,t} \mid i,j,t \geq 1]_{n*n}$	The set of traffic volume at time t, where $tr_{i,j,t}$ is the traffic volume from subnet i to subnet j at time t
	$path_{i,j,k} = \{link_{x,y} \mid x,y \geq 1\}$	The link set of k-th path from subnet i to subnet j.
	$W = \{w_{i,j,k} \mid i,j,k \geq 1\}$	The set of path weight, where $w_{i,j,k}$ is the weight of $path_{i,j,k}$.
	$NPATH = \{npath_{i,j} \mid i,j \geq 1\}$	The set of path number, where $npath_{i,j}$ is the number of paths from subnet i *to* subnet j.
	$DEP^{PATH} = \left\{dep^{path}_{i,j,k,l} \mid i,\ j,\ k,\ l \geq 1\right\}$	The set of path dependency, where $dep^{path}_{i,j,k,l}$ is the path dependency from $path_{i,j,k}$ to $path_{i,j,l}$.
	$DEP^{LOCAL} = \left\{dep^{local}_{i,j,k} \mid i,\ j,\ k \geq 1\right\}$	The set of local dependency, where $dep^{local}_{i,j,k}$ is the local dependency of $path_{i,j,k}$.
	$DEP^{GLOBAL} = \left\{dep^{global}_{i,j,k} \mid i,\ j,\ k \geq 1\right\}$	The set of global dependency, where $dep^{global}_{i,j,k}$ is the global dependency of $path_{i,j,k}$.
	nLD	The number of selected local dependency paths
	nGD	The number of selected global dependency paths
	$D^{SP} = \left\{d^{sp}_{i,j} \mid i,\ j \geq 1\right\}$	The set of average delay of subnet pair, where $d^{sp}_{i,j}$ is the average delay of from subnet i to subnet j.
	$D^{P} = \left\{d^{p}_{i,j,k} \mid i,\ j,\ k \geq 1\right\}$	The set of path delay, where $d^{sp}_{i,j,k}$ is the delay of $path_{i,j,k}$.
	$D^{LINK} = \left\{d^{link}_{x,y} \mid x,\ y \geq 1\right\}$	The set of link delay, where $d^{link}_{x,y}$ is the delay of $link_{x,y}$.
	$BOOL = \{bool_{i,j,k,x,y} \mid i,j,k,x,y \geq 1\}$	The set of boolean value that if specific path passes specific link, where $bool_{i,j,k,x,y}$ = true means $path_{i,j,k}$ passes $link_{x,y}$.
	$ct_{x,y}$	Current traffic volume of $link_{x,y}$.
	$mb_{x,y}$	Maximum bandwidth of $link_{x,y}$.
	pro_mu	Probability of mutation in genetic algorithm
	$budget$	The number of iterations that genetic algorithm runs.
	$tr^{predict}_{i,j,t}$	The prediction of traffic volume from subnet i to subnet j at time t.
	$trlower7pt\text{-}avg^{predict2}_{i,j,t}$	The prediction of average traffic volume from subnet i to subnet j at time t.
	$trlower7pt\text{-}\mathrm{var}^{predict1}_{i,j,t}$	The prediction of variance traffic volume from subnet i to subnet j at time t.
	α	The smoothing factor of exponential smoothing

3.2 | Problem statement

In our proposed PMRP, we proactively provision multiple paths for each subnet pair to minimize the total sum of the products of average transmission delays and traffic volumes of each subnet pair. Hence, the objective function is given as

$$\text{Objective: Min}\left(\sum_{i=1}^{n}\sum_{j=1}^{n} d^{sp}_{i,j} \times tr_{i,j}\right). \tag{1}$$

For subnet pair from i to j, we pre-provision $npath_{i,j}$ paths, and the average delay of subnet pair i, j, which is the sum of the multiplications of path weight and path delay, can be expressed as

$$\text{Delay of subnet pair:} d^{sp}_{i,j} = \sum_{k=1}^{npath_{i,j}} w_{i,j,k} d^{p}_{i,j,k}, \tag{2}$$

The path delay is total delay of links that this path passes through and is given as

$$\text{Path delay}: d^{p}_{i,j,k} = \sum_{x=1}^{m} \sum_{y=1}^{m} d^{link}_{x,y} bool_{i,j,k,x,y}. \tag{3}$$

We conduct M/M/1 model considering the maximum bandwidth and current traffic volume on this link for calculating the delay of links as

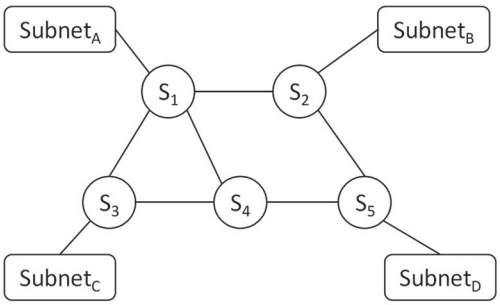
$$\text{Link delay}: d^{link}_{x,y} = \begin{cases} 1/\left(mb_{x,y} - \text{Min}\left(ct_{x,y}, mb_{x,y}\right)\right), & \text{if } ct_{x,y} < mb_{x,y} \\ 1 & , \textit{if } ct_{x,y} = mb_{x,y} \end{cases}. \tag{4}$$

Finally, the current traffic volume of each link is the total traffic volume of paths that pass this link and can be derived as

$$\text{Current traffic volume of } link_{x,y}: ct_{x,y} = \left(\sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{npath_{i,j}} tr_{i,j} w_{i,j,k} bool_{i,j,k,x,y} \right). \tag{5}$$

3.3 | Example

Figure 1A depicts a scenario topology. There are four subnets Subnet_A, Subnet_B, Subnet_C, and Subnet_D with traffic matrix *TR* shown in Figure 1B. We assume that the maximum bandwidth of each link $mb_{x,y}$ is 100. Without multipath routing, the traffic of each subnet pair traverses the topology using shortest path algorithm. We set the current traffic volume $ct_{x,y}$ by applying Equation (5) as illustrated in Figure 1C. For example, considering the traffic flow from Subnet_D to Subnet_A, it will follow the shortest path $\{link_{5,2}, link_{2,1}\}$. At $link_{5,2}$, there are traffic flows from subnet pairs (Subnet_D, Subnet_A) and (Subnet_D, Subnet_B), both with traffic volume of 50 and $ct_{5,2}$ of 100. On $link_{2,1}$, there are traffic flows from subnet pairs (Subnet_D, Subnet_A) and (Subnet_B, Subnet_A) with traffic volumes of 50 and 60, respectively. Although we can calculate $ct_{2,1}$ as 50 + 60 = 110, it is higher than the maximum bandwidth 100, and hence we set $ct_{2,1}$ to 100. Because both traffic volume of $link_{5,2}$ and $link_{2,1}$ are then congested, the delay of links $d^{link}_{5,2}$ and $d^{link}_{2,1}$ are both 1 according to)



(A) Scenario Topology

src \ dst	A	B	C	D
A		0	0	60
B	60		0	40
C	20	0		80
D	50	50	0	

(B) traffic volume matrix *TR*

src \ dst	1	2	3	4	5
1		60	0	0	0
2	***100***		0	0	100
3	20	0		80	0
4	0	0	0		80
5	0	100	0	0	

(C) Current traffic volume with single shortest path

src \ dst	1	2	3	4	5
1		60	0	0	0
2	85		0	0	100
3	20	0		80	0
4	25	0	0		80
5	0	75	0	25	

(D) Current traffic volume if reroute from D to A

FIGURE 1 Example of subnet pair delay calculation

Equation (4). As this is the only path from Subnet_D to Subnet_A, we can derive the delay of this subnet pair as

$$d^{sp}_{Subnet_D,Subnet_A} = d^{p}_{Subnet_D,Subnet_A,1} = d^{link}_{5,2} + d^{link}_{2,1} = 1 + 1 = 2.$$

We see that the subnet pair from Subnet_D to Subnet_A suffers from considerable delay as a result of link congestion. If we reroute half of the flow to another path $\{link_{5,4}, link_{4,1}\}$, current traffic volume $ct_{x,y}$ is updated in Figure 1D. Now there are two paths from Subnet_D to Subnet_A: the first path is $\{link_{5,2}, link_{2,1}\}$,and the 2nd is $\{link_{5,4}, link_{4,1}\}$, both equally weighted, and hence $w_{Subnet_D,Subnet_A,1} = w_{Subnet_D,Subnet_A,2} = 0.5$. From Equation (4), we calculate the delay of the links according to M/M/1 model and then compute the delay of the two paths according to Equation (3) as

$$d^{p}_{Subnet_4,Subnet_1,1} = d^{link}_{5,2} + d^{link}_{2,1} = 1/(100-75) + 1/(100-85) = 0.1067,$$

$$d^{p}_{Subnet_4,Subnet_1,2} = d^{link}_{5,4} + d^{link}_{4,1} = 1/(100-25) + 1/(100-25) = 0.0267.$$

Finally, we can obtain the average delay of the subnet pair from Subnet_D to Subnet_A by Equation (2) as

$$d^{sp}_{Subnet_4,Subnet_1} = 0.5{*}d^{p}_{Subnet_4,Subnet_1,1} + 0.5{*}d^{p}_{Subnet_4,Subnet_1,2} = 0.0667.$$

Hence, we can avoid the possible network congestions by proactively multipath provisioning. *TR* and M/M/1 model are used for delay estimation. With the estimated delay, we can find the optimal combination of multipath and weight by the objective function.

4 | PROACTIVE MULTIPATH ROUTING WITH A PREDICTIVE MECHANISM (PMRP)

In this section, we give the details of the proposed multipath routing system by applying PMRP. Proactive multipath routing is designed to decide paths and weights in order to split the traffic flows into these paths so as to reach global optimization. To deal the variations in traffic volume, we developed a method to anticipate future traffic loads. Finally, we present the PMRP system architecture that contains both proactive and predictive operations.

4.1 | Proactive multipath routing

In proactive multipath routing operation, we wish to determine the paths to be allocated and the weights to be split traffic among these paths. First, we calculate all available path sets for each subnet pair with the Breadth-First Search (BFS) algorithm, which searches from the root node and explore all nodes at the present depth before moving on to the next depth. However, we cannot use all of them to split the traffic, because computing complexity is extreme high. As a result, our path selection depends on the dependencies of each path. Paths with lower dependency overlap less with other paths and are therefore selected for multipath routing.

For *k*-th path from subnet *i* to subnet *j* denoted as $path_{i,j,k}$, we compute the path dependency from $path_{i,j,k}$ to another path $path_{i,j,l}$ as $dep^{path}_{i,j,k,l}$, which is the number of overlapping links between $path_{i,j,k}$ and $path_{i,j,l}$ divided by the number of links of $path_{i,j,k}$. The path dependency $dep^{path}_{i,j,k,l}$ can be calculated as

$$dep^{path}_{i,j,k,l} = \begin{cases} \left(\sum_{x=1}^{m}\sum_{y=1}^{m} bool_{i,j,k,x,y} bool_{i,j,l,x,y}\right) / \sum_{x=1}^{m}\sum_{y=1}^{m} bool_{i,j,k,x,y}, & \text{if } k \neq l \\ 0, & \text{if } k = l \end{cases}. \tag{6}$$

Then, we derive the local dependency of $path_{i,j,k}$, which is the number of links that the path shares with the other paths in the same subnet pair, and then divided by $npath_{i,j}$, as

$$dep^{local}_{i,j,k} = \left(\sum_{l=1}^{npath_{i,j}} dep^{path}_{i,j,k,l}\right) / npath_{i,j}. \tag{7}$$

We then select at most the *nLD* paths with the lowest local dependency for each subnet pair. Moreover, for global awareness, we define global dependency $dep_{i,j,k}^{global}$ as

$$dep_{i,j,k}^{global} = \sum_{x=1}^{m}\sum_{y=1}^{m} bool_{i,j,k,x,y} * \left(\sum_{a=1}^{n}\sum_{b=1}^{n}\sum_{l=1}^{npath_{a,b}} bool_{a,b,l,x,y}\right), \tag{8}$$

which counts the overlapping links with paths from other subnet pairs. Finally, from the *nLD* paths selected above, we pick at most the *nGD* paths with the lowest global dependency.

The number of selected local dependency paths *nLD* represents the number of paths with the lowest degree of overlapping. As *nLD* increases, we could discover sparser paths based on network topology. However, these paths may have higher hop counts which we would like to avoid, and therefore the performance with larger *nLD* may not always be better. The number of selected global dependency paths *nGD* is a key factor that affects computation time. With larger *nGD*, we could determine rerouted paths more accurately but it will also exhaust more iterations. Hence, there is a trade-off between routing performance and computation time based on network scale and traffic volume.

For example, we have two subnets Subnet_A and Subnet_B with topology shown in Figure 2. The gateway switches for Subnet_A and Subnet_B are S_1 and S_8, respectively. There are three paths from subnet pair from Subnet_A to Subnet_B

$$path_{Subnet_A,Subnet_B,1} = \{Link_{1,2}, Link_{2,3}, Link_{3,8}\},$$

$$path_{Subnet_A,Subnet_B,2} = \{Link_{1,2}, Link_{2,4}, Link_{4,5}, Link_{5,8}\},$$

$$path_{Subnet_A,Subnet_B,3} = \{Link_{1,6}, Link_{6,7}, Link_{7,8}\}.$$

Considering the path $path_{Subnet_A,Subnet_B,1}$, we calculate its path dependency compared with $path_{Subnet_A,Subnet_B,2}$ (with one overlapped link) and $path_{Subnet_A,Subnet_B,3}$ (without overlapped link) from Equation (6) as

$$dep_{Subnet_A,Subnet_B,1,2}^{path} = 1/3,$$

$$dep_{Subnet_A,Subnet_B,1,3}^{path} = 0/3 = 0.$$

We can obtain the local dependency of $path_{Subnet_A,Subnet_B,1}$ by Equation (7) as

$$\text{dep}_{\text{Subnet}_\text{A},\text{Subnet}_\text{B},1}^{\text{local}} = \left(\sum_{\text{l}=1}^{\text{npath}_{\text{Subnet}_\text{A},\text{Subnet}_\text{B}}} \text{dep}_{\text{Subnet}_\text{A},\text{Subnet}_\text{B},1,\text{l}}^{\text{path}}\right) / \text{npath}_{\text{Subnet}_\text{A},\text{Subnet}_\text{B}} = (0 + 1/3 + 0)/3 = 1/9.$$

Similarly, local dependencies of other paths can be computed, and we then select the lowest ones for calculating global dependencies.

After selecting at most the *nGD* paths with the lowest global dependencies, we assign a weight to each path and try every possible combination of these weights for better performance, with the constraint

$$\left(w_{i,j,1} + w_{i,j,2} + ... + w_{i,j,nGD}\right) = nGD. \tag{9}$$

However, testing every combination by brute force for determining the minimum average delay is impossible. Because the number of weight combinations is to the power of the number of subnet pairs, we know that the number of weight

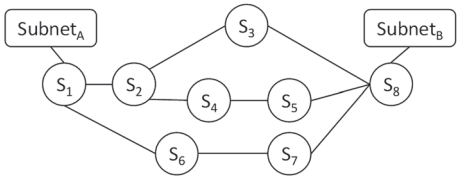


FIGURE 2 Example of path dependency and local dependency calculation

combinations grows exponentially. As a result, we leverage the GA to accelerate the computation for the solution. The GA[16] is an optimization method that finds good global solutions at high speed. The algorithm has four main components in each stage we call iteration: Gene, Chromosome, Fitness function, and Population. The flowchart of this algorithm is illustrated in Figure 3, which shows its four main functions: Evaluation, Selection, Crossover, and Mutation. The "mindset" of such GA is survival of the fittest. The possible solutions compete with each other and only the better ones survive. At first, we generate a number of chromosomes, and each chromosome is composed of random genes. The evaluation operation will apply fitness function for these chromosomes. After that, the selection function will copy and replace some chromosomes by probability according to its fitness, so the chromosomes with better fitness are more likely to survive. Both crossover function and mutation function update chromosomes by changing their portion of genes with different approaches. The crossover function randomly selects two chromosomes by probability and exchanges their random part of genes. The mutation function directly changes the genes of chromosome with random genes by probability. Because GA does not guarantee to find the optimal solution, it needs to run several iterations for training the solution to reach as optimal as possible. However, more iteration count results in more computation time. Hence, a threshold called *budget* is defined as the upper bound of number of iterations. In PMRP, we map the path weight of a subnet pair to one gene, and the path weight set of a subnet pair to one chromosome. For example, if the path weight set of a subnet pair is {4,3,2,1,0,0,0,0,0,0}, it means that 40% traffic is on the first path, 30% traffic on the second path, 20% traffic on the third path, 10% traffic on the fourth path, and no traffic on the fifth to 10th paths. We map each of the weights to a gene and the total set to a chromosome. The population is the number of chromosomes in each iteration, and the fitness function is mapped to our objective function in Equation (1) as the criteria of survival. The evaluation function calculates fitness function for the whole population of chromosomes, and then we perform the following operations of selection, crossover, and mutation with probabilities. Probability of selection is the ratio of chromosomes that would be replaced with better chromosome at each iteration. Probability of crossover is the ratio of chromosomes that will be involved in crossover operation. Probability of mutation is the probability of each gene being replaced at random with another gene. The GA runs repeatedly till the number of iterations reaches *budget*, and we can obtain the optimal solution from the final iteration.

4.2 | Predictive mechanism

When GA computes the objective function in Equation (1), link delay defined in Equation (4) should be determined. For calculating Equation (4), we need to update the current traffic volume which can be obtained by OpenFlow protocol. However, after we solve the objective function and submit the flow entries to SDN switches, the traffic volume would be altered according to user's activities. In order to adapt to the variance of real traffic volume, we deploy a predictive mechanism based on exponential smoothing[17] used in TCP retransmission timeout (RTO) to forecast the traffic volume of each subnet pair. According to the concept of RTO, we predict the value of traffic volume with double exponential smoothing and variance of traffic volume with single exponential, respectively. Therefore, at time t, we can predict

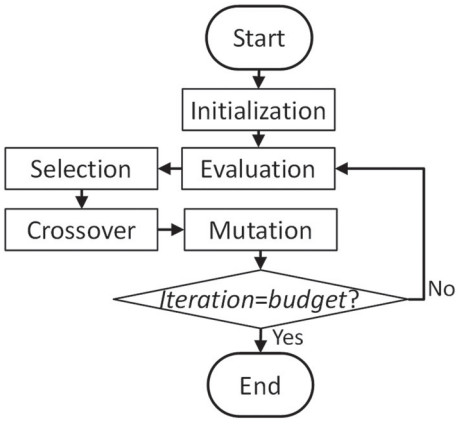


FIGURE 3 Flowchart of genetic algorithm

the traffic volume from subnet i to subnet j for time $t+1$ as

$$tr_{i,j,t+1}^{predict} = tr\_avg_{i,j,t}^{predict2} + tr\_\mathrm{var}_{i,j,t}^{predict1}. \tag{10}$$

$tr\_avg_{i,j,t}^{predict2}$ is the predicted average traffic volume from subnet i to subnet j at time t and is conducted by model of double exponential smoothing as

$$tr\_avg_{i,j,t}^{predict2} = \frac{2-\alpha}{1-\alpha} tr\_avg_{i,j,t}^{predict1} - \frac{1}{1-\alpha} tr\_avg_{i,j,t}^{predict2'}. \tag{11}$$

To obtain $tr\_avg_{i,j,t}^{predict2}$, we first determine average traffic volume from recent five detected results (from time *t-4* to *t*) as

$$tr\_avg_{i,j,t} = \begin{cases} \sum_{z=0}^{4} tr_{i,j,t-z}/5 & , \text{if } t \geq 5 \\ \sum_{z=0}^{t-1} tr_{i,j,t-z}/t & , \text{if } t < 5 \end{cases}. \tag{12}$$

$tr\_avg_{i,j,t}^{predict1}$ can be then calculated by single exponential smoothing from $tr\_avg_{i,j,t}$ as

$$tr\_avg_{i,j,t}^{predict1} = \begin{cases} tr\_avg_{i,j,t-1}^{predict1} + \alpha\left(tr\_avg_{i,j,t} - tr\_avg_{i,j,t-1}^{predict1}\right) & , if\ t > 1 \\ tr\_avg_{i,j,1} & , if\ t = 1 \end{cases}. \tag{13}$$

Similarly, $tr\_avg_{i,j,t}^{predict2'}$ can be computed by double exponential smoothing from $tr\_avg_{i,j,t}^{predict1}$ as

$$tr\_avg_{i,j,t}^{predict2'} = \begin{cases} tr\_avg_{i,j,t-1}^{predict2'} + \alpha\left(tr\_avg_{i,j,t}^{predict1} - tr\_avg_{i,j,t-1}^{predict2'}\right), & if\ t > 1 \\ tr\_avg_{i,j,1} & , if\ t = 1 \end{cases}. \tag{14}$$

$tr\_\mathrm{var}_{i,j,t}^{predict1}$ is the variance of predicted traffic volume from subnet i to subnet j at time t and can be conducted by single exponential smoothing as

$$tr\_\mathrm{var}_{i,j,t}^{predict1} = \begin{cases} tr\_\mathrm{var}_{i,j,t-1}^{predict1} + \alpha\left(tr\_\mathrm{var}_{i,j,t} - tr\_\mathrm{var}_{i,j,t-1}^{predict1}\right) & , \text{if } t > 1 \\ 0 & , if\ t = 1 \end{cases}. \tag{15}$$

We give an example of traffic prediction in Table 4. The observed real traffic volume at $t = 1$, 2, and 3 are 2, 6, and 7, respectively, and we assume $\alpha = 0.4$. At time $t = 1$, both $tr\_\mathrm{var}_{i,j,1}^{predict1}$ and $tr\_avg_{i,j,1}^{predict2'}$ are equal to $tr\_avg_{i,j,1} = 2$, and

TABLE 4 Example for predictive mechanism

Time *t*	***t* = 1**	***t* = 2**	***t* = 3**	***t* = 4**
Observed traffic volume	2	6	7	
$tr\bar{}\ avg_{i,j,t}$	2	4	5	
$trlower7pt\bar{}avg_{i,j,t}^{predict1}$	2	2.8	3.68	
$trlower7pt\bar{}avg_{i,j,t}^{predict2'}$	2	2.32	2.864	
$trlower7pt\bar{}avg_{i,j,t}^{predict2}$			**5.04**	
$tr\bar{}\ var_{i,j,t}$	0	4	4.667	
$trlower7pt\bar{}\mathrm{var}_{i,j,t}^{predict1}$	0	1.6	**2.8267**	
$tr_{i,j,t}^{predict}$				**7.8667**

$tr\_\mathrm{var}_{i,j,1}^{predict1}$ is 0 because there is no traffic variance initially. At time $t = 2$, we have an average traffic volume $tr\_avg_{i,j,2} = (2+6)/2 = 4$ and derive $tr\_avg_{i,j,2}^{predict1}$ and $tr\_avg_{i,j,2}^{predict2'}$ according to Equations (13) and (14). The traffic variance between 2 and 6 $tr\_var_{i,j,2}$ is 4, and $tr\_\mathrm{var}_{i,j,2}^{predict1}$ is 1.6, derived from Equation (15). At time $t = 3$, the average traffic of 2, 6, and 7 $tr\_avg_{i,j,3}$ is 5, and we again derive $tr\_avg_{i,j,3}^{predict1}$ and $tr\_avg_{i,j,3}^{predict2'}$ according to Equations (13) and (14). The traffic variance between 2, 6, and 7 $tr\_var_{i,j,3}$ is 4.667, and we calculate $tr\_\mathrm{var}_{i,j,2}^{predict1}$ as 2.8267 according to Equation (15). In order to forecast the traffic volume at $t = 4$, we compute $tr\_avg_{i,j,3}^{predict2} = 5.04$ from Equation (11) and obtain $tr_{i,j,4}^{predict} = tr\_avg_{i,j,3}^{predict2} + tr\_\mathrm{var}_{i,j,3}^{predict1} = 5.04 + 2.8267 = 7.8667$. Hence, the predicted traffic volume for $t = 4$ from the past detected results 2, 6, and 7 is 7.8667.

4.3 | System architecture and implementation

The system architecture of the proposed PMRP is illustrated in Figure 4. We implemented PMRP mechanisms on Ryu[18] controller version 4.10. At first, Ryu controller collects the network topology from SDN switches and assesses all available paths of each subnet pair with Breadth-First Search (BFS) algorithm. Then, the proactive multipath routing procedure described in Section 4.1 selects 10 candidate paths with lowest global dependency for each subnet pair. Ryu controller then collects bandwidth statistic of all egress switches by *TR* collector and forecasts the traffic volume applying the predictive mechanism covered in Section 4.2. After that, we use the GA to search for an optimal solution from candidate paths with possible weight sets and predicted traffic volumes. Finally, Ryu controller issues the flow entries of the solution from GA to the underlying SDN switches. *TR* collector will collect the statistics every sampling cycle, and PMRP procedure is carried out repeatedly.

5 | NUMERICAL EVALUATION

In this section, we briefly describe our experiment environment and the analyses of the experimental results.

5.1 | Environment setup

In the simulation, we deployed two servers with Intel core i7-4790 CPU 4.00 GHz and 64 GB DRAM and installed a VMWare Workstation. The first server acted as an OpenFlow controller, running Ryu[18] and the proposed PMRP algorithm. In PMRP, we set $nLD = 30$ and $nGD = 10$. The parameters of GA are listed in Table 5. We set Probability of Selstion = 20%, Probability of Crossover = 80%, Probability of Mutation = 2%, and Budget = 3600. The second server

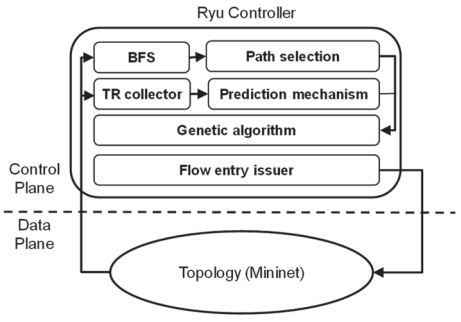


FIGURE 4 System architecture of PMRP

TABLE 5 Parameters of GA in simulation

Probability of Selection	Probability of Crossover	Probability of Mutation	Budget
20%	80%	2%	3600

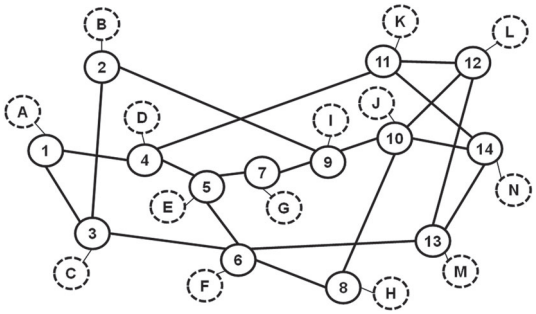


FIGURE 5 Experiment topology

simulated the network with the topology illustrated in Figure 5, which shows the use of Mininet[19] to simulate 14 core SDN switches (1-14) and 14 subnets (A-N). Each subnet is equipped with Tcpreplay as the traffic generator. For the replayed traffic, we utilized CAIDA UCSD Anonymized Internet Traces 2016.[20] which is a dataset that contains anonymized traces captured at CAIDA's monitor sites in PCAP format. To form the traffic matrix pattern among these subnets in our simulation, we selected 20 connections from the CAIDA dataset for each subnet pair, and hence there are total $14 \times 13 \times 20 = 3640$ flows per simulation run. When we replay the PCAP file, we control the replay speed with a parameter called traffic matrix scale to reflect various network loading situations. The traffic replay speed is the origin traffic speed stored in PCAP file multiplied by traffic matrix scale. Finally, we performed five runs for each simulation and calculated the average of these results.

5.2 | Experimental results

In this section, we provide the results of our observations of the performance on average delay, effectiveness of prediction, and flow table usage. For average delay results, we compared PMRP with single shortest path and UTAMP proposed in Peng et al.[5] As discussed in Section 2.1, UTAMP deploys multipath flows based on available network resources, which is similar to PMRP. UTAMP is a flow-triggered algorithm. It employs a sampling cycle time and updates network status at each cycle. The path decision will be made according to the latest network status when a new packet arrives. Because UTAMP performs host-based routing, it consumes more flow entries. In contrast, PMRP has a predictive mechanism and proactively reroutes packets at each sampling cycle. PMRP routes the traffic according to subnet pairs and hence saves flow table spaces. In the following results, we wanted to compare several performances between these algorithms.

5.2.1 | Performance on average delay

The average delay between PMRP, UTAMP, and the single shortest path is illustrated in Figure 6. The X-axis is the scale of traffic matrix produced online.[20] We can see that average delay increased as the traffic increased, and our PMRP outperformed both UTAMP and the single shortest path. At traffic matrix scale 1.25, there was 49% reduction in average delay as compared with single shortest routing and a 16% reduction in average delay compared with UTAMP. Because

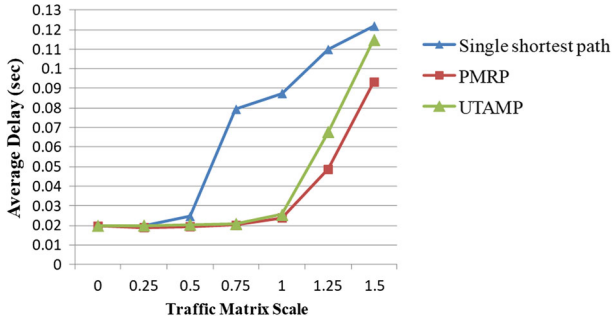


FIGURE 6 Average delay comparisons

UTAMP provisions flows based on host pairs, flow tables will become exhausted with higher traffic demand, and hence it becomes impossible to allocate extra multipaths. PMRP, by contrast, does not occupy much flow space and can provision multipaths for congestion alleviation under high traffic volumes. Table 6 lists the average number of paths created for a single pair from subnet A (switch 1) to subnet N (switch 14). According to Table 6, when the traffic matrix scale = 0.25, UTAMP can allocate on average six paths for a single host pair, but only three when traffic matrix scale = 1.5. PMRP can allocate on average four paths for a single subnet pair when the traffic matrix scale = 0.25 and allocate five paths when the traffic matrix scale = 1.5 Therefore, PMRP produces better flow table management for multipath provisioning with high traffic volumes.

Figure 7 shows the impact of probability of mutation *pro_mu* in the GA under traffic matrix scale = 1. With *pro_mu* = 0%-0.5%, the average delay decreases with the growing of *pro_mu*. Because when *pro_mu* is too low, it is difficult for genes to mutate into a better solution so that we cannot find an optimal solution efficiently. However, with *pro_mu* = 2%-60%, the average delay increases with the growth of *pro_mu*. The reason is that for higher *pro_mu*, the genes transform too fast, and better genes may not survive to the next iteration. We concluded then that the optimum value of *pro_mu* is between 0.5% and 2%.

5.2.2 | Effectiveness of prediction mechanism

Figure 8 depicts the performance of UTAMP and PMRP with and without the prediction mechanism. The X-axis has the sampling cycle time of *TR* collector. Apparently, PMRP with and without prediction outperforms UTAMP. UTAMP strongly relies on precise traffic statistics to make appropriate route decision when a new packet arrives. With lower sampling cycle time, UTAMP could have more accurate network status, and therefore the performance on average delay is better. However, PMRP makes path reroutes every sampling cycle while UTAMP only monitors network status.

TABLE 6 Average number of paths created for a single pair from Subnet A to Subnet N

Traffic Matrix Scale	0.25	0.5	0.75	1	1.25	1.5
UTAMP	6	4	4	3	3	3
PMRP	4	4	3	4	5	5

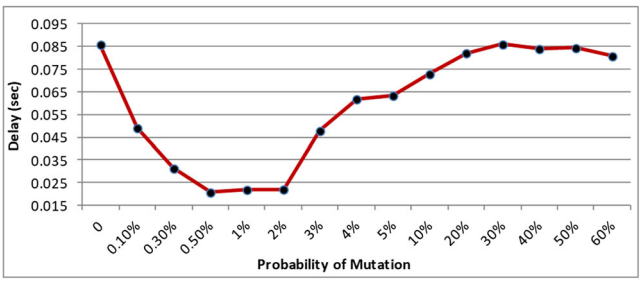


FIGURE 7 Probability of mutation

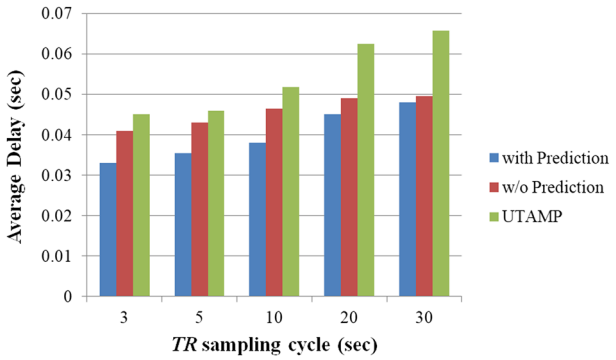


FIGURE 8 Performance on effectiveness of prediction

Hence, we assure that PMRP has better performance than UTAMP. We observed that the average delay of PMRP with prediction is significantly reduced when sampling cycle time = 3-10. With the predictive mechanism, PMRP can forecast the variance of the traffic volume and choose a suitable solution to meet our objective function. It could achieve reduction of 20% delay on average with sampling cycle time = 10. However, with an increasing sampling cycle time, the improvement for the prediction mechanism is limited because it is difficult to forecast accurately if the sampling cycle is lengthened. Although we can improve on the accuracy of prediction by increasing the sampling rate, frequent collection of network status would increase the control plane overhead, and such a trade-off should be carefully managed.

5.2.3 | Comparison of traffic splitting schemes

In Section 2.3, we discussed the flow-based and packet-based traffic splitting schemes for multipath traffic forwarding in OpenFlow. We deploy flow-based algorithm in order to prevent packet out-of-order problem in our experiments. In order to evaluate the performances between two traffic splitting schemes, Figures 9, 10, and 11 depict the results of out-of-order, traffic loss, and average delay of PMRP using flow-based and packet-based algorithms. For flow-based algorithm, there is no out-of-order packet under low traffic volume. As traffic matrix scale is larger than 2, there are traffic losses which cause out-of-order packets due to the missing components. In contrast, packet-based algorithm suffered from out-of-order problem even under low traffic volume. We also observed that the traffic matrix scale affects the loss rate of packet-based scheme more heavily than the flow-based scheme. It is because the out-of-order packets will invoke TCP DPU ACKs and cause packet retransmissions. With higher traffic matrix scale, such situation gets worse and therefore results in higher packet loss rate. At traffic matrix scale 2.25, we observed that out-of-order of the

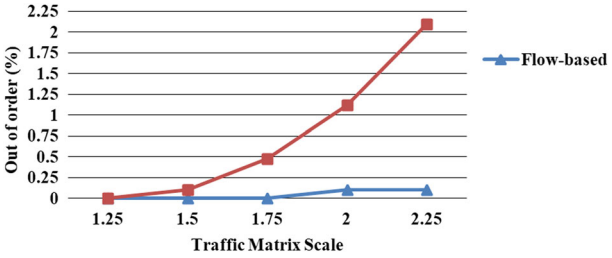


FIGURE 9 Out-of-order comparisons between flow-based and packet-based schemes

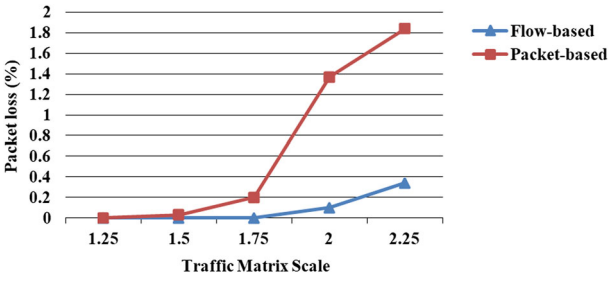


FIGURE 10 Packet loss comparisons between flow-based and packet-based schemes

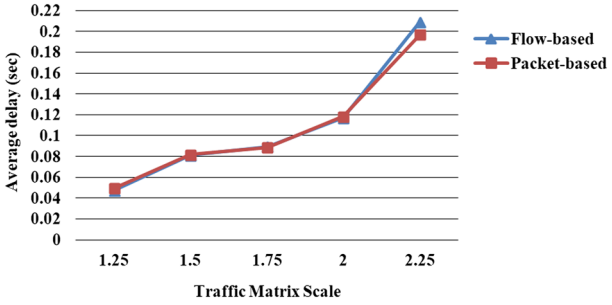


FIGURE 11 Average delay comparisons between flow-based and packet-based schemes

TABLE 7 Flow table usage

	Total Request Number	Average Flow Table Usage for Single Request	Average Flow Table Usage Per Switch
UTAMP	4243	11.42	3523
PMRP		N/A	233

packet-based scheme is 2.1%, while that of the flow-based scheme is only 0.1% in Figure 9. From Figure 10, we could also see that the packet loss of the packet-based scheme is 1.84%, while that of the flow-based scheme is only 0.34%. Therefore, the flow-based scheme has a 95% reduction in out-of-order and an 81% reduction in packet loss compared with the packet-based scheme. As a trade-off, the flow-based scheme may cause higher packet delay due to unbalanced paths of traffic splitting. However, we learn that the packet delay performances of both algorithms are almost the same when traffic matrix scale ranges from 1.25 to 2. At traffic matrix scale 2.25, the packet delay of the packet-based scheme is 0.1967s, and that of the flow-based scheme is 0.2093 in Figure 11. Hence, the packet delay of the flow-based scheme is only 6% higher than the packet-based scheme. Hence, the flow-based scheme outperforms the packet-based scheme in both out-of-order and packet loss with acceptable performance reduction in average delay.

5.2.4 | Flow table usage

Table 7 gives the comparisons between UTAMP and PMRP on average flow table usage. With the traffic data online,[20] we have a total of 4243 requests. For UTAMP, multipaths are calculated and provisioned for each host-to-host connection request. Our results show an average of 11.42 flow entries generated for each request and that an average of 3523 flow entries were installed at each SDN switch. PMRP, by contrast, proactively and periodically computes the multipaths among the subnet pairs with predicted traffic volumes, and the number of flow entries is thus not affected by a single connection request. The average flow entries at each SDN switch taken by PRMP are 233, which outperforms UTAMP by 93%.

6 | CONCLUSIONS

In this paper, we propose the application of PMRP to resolve congestion resulting from multipath provisioning. With SDN technology, PMRP can monitor and manage the network within a centralized controller and utilize the global view of information for multipath decisions and traffic predictions. PMRP proactively calculates the multipaths for each subnet pair using the GA. In contrast with similar, related work that manages multipath flows for each host pair, PMRP can significantly save the usage of the flow table. For each end-to-end connection, PMRP hashes their MAC addresses and ensures that the flow of each host pair will always traverse the same path so that there will be no out-of-order packet problems. In addition, PMRP adopts a predictive mechanism with exponential smoothing to forecast the demand traffic volume of each subnet pair to avoid congestion beforehand. The simulation results show that PMRP reduces average delay by 49% compared with single shortest routing, and by 16% as compared with UTAMP. On the other hand, the probability of mutation in the GA should be set between 0.5% and 2% for optimum performance. The prediction mechanism of PMRP outperforms PMRP without the prediction mechanism by 20% on average delay. We also observed that the flow table of PMRP saves 93% more than UTAMP. Hence, PMRP could manage flow table resources better than UTAMP for allocating more multipaths in order to avoid traffic congestions.

ORCID

Te-Lung Liu https://orcid.org/0000-0002-3142-0302

REFERENCES

1. ONF SDN Architecture Issue 1.1, [Online]. Available: https://www.opennetworking.org/images/stories/downloads/sdn-resources/technical-reports/TR-521_SDN_Architecture_issue_1.1.pdf.
2. Pan Q, and Zheng X. Multi-path SDN route selection subject to multi-constraints, IEEE Third International Conference on Cyberspace Technology (CCT 2015), Beijing, October 2015.

3. Jain S, Kumar A, Mandal S, et al. B4: experience with a globally-deployed software defined WAN. *ACM SIGCOMM Comput Commun Rev*, vo. October 2013;43, issue(4):3-14.
4. Liu W, Zhou W, Wang Y, Duan Y, and Gao Z, Flexible multi-path routing for global optimization in software-defined datacenters, 2016 IEEE Symposium on Computers and Communication (ISCC), Italy, June 2016.
5. Peng Y, Deng Q, Guo L, Ning Z, and Zhang L, Design of dynamic traffic grooming algorithm in software-defined wireless mesh networks, 2015 IEEE 17th International Conference on High Performance Computing and Communications (HPCC), New York, August 2015.
6. Wang W, He W, and Su J, M2SDN: achieving multipath and multihoming in data centers with software defined networking, 2015 *IEEE 23rd International Symposium on Quality of Service (IWQoS)*, Portland, June 2015.
7. Hamed MI, El Halawany BM, Fouda MM, and Eldien AST, A novel approach for resource utilization and management in SDN, 2017 *13th International Computer Engineering Conference (ICENCO)*, Egypt, February 2017.
8. Wang W, Enhancing control consistency of software-defined networking, *Doctoral dissertation*, McGill University, August 2017.
9. Kuang L, Yang LT, Wang X, Wang P, Zhao Y. A tensor-based big data model for QoS improvement in software defined networks. *IEEE Netw*. January 2016;30(1):30-35.
10. Xiao P., Qu W., Qi H., Xu Y., Li Z., An efficient elephant flow detection with cost-sensitive in SDN, *1st International Conference on Industrial Networks and Intelligent Systems (INISCom)*, Japan, March 2015.
11. Su Z, Wang T, Xia Y, and Hamdi M, CheetahFlow: towards low latency software-defined network, 2014 IEEE International Conference on Communications (ICC), Sydney, 2014.
12. Cello M, Marchese M, Mongelli M. On the QoS estimation in an OpenFlow network: the packet loss case. *IEEE Commun Lett*. March 2016;20(3):554-557.
13. Benson T, Akella A, and Maltz DA, Network traffic characteristics of data centers in the wild, Proceedings of the 10th ACM SIGCOMM conference on Internet measurement, Australia, November 2010.
14. "OpenFlow Switch Specification version 1.0.0," [Online]. Available: https://www.opennetworking.org/images/stories/downloads/sdn-resourcspecifications/openflow/openflow-spec-v1.0.0.pdf
15. "Open vSwitch," [Online]. Available: https://www.openvswitch.org/
16. Haupt RL, Haupt SE. *Practical Genetic Algorithms*. 2nd ed. Wiley-Interscience; 2004.
17. Kalekar PS. Time series forecasting using holt-winters exponential smoothing. *Kanwal Rekhi Sch Inf Technol*. 2004;4329008.
18. Ryu, [Online]. Available: https://osrg.github.io/ryu/.
19. Mininet, [Online]. Available: http://www.mininet.org
20. The CAIDA UCSD Anonymized Internet Traces 2016, [Online]. Available: http://www.caida.org/data/passive/passive_2016_dataset.xml

How to cite this article: Lin Y-D, Liu T-L, Wang S-H, Lai Y-C. Proactive multipath routing with a predictive mechanism in software-defined networks. *Int J Commun Syst*. 2019;32:e4065. https://doi.org/10.1002/dac.4065