Collaborative Route Management to Mitigate Congestion in Multi-Domain Networks Using SDN

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

Li-Chia Yen
Department of Computer Science and Engineering
National Sun Yat-sen University
Kaohsiung, Taiwan
m083040037@student.nsysu.edu.tw

***Abstract*—In *software-defined networking (SDN)*, a controller monitors switches and directs them to process packets, thereby making network management easier. In this paper, we consider an SDN-based network with multiple connected domains, where each domain contains a set of switches supervised by a controller. Moreover, flows can have different priorities. Then, we propose a *multi-domain collaborative route management (MCRM)* scheme to schedule routing paths of flows to improve network performance. Each controller adaptively adjusts the paths of some flows in the domain based on their priorities to avoid congestion. However, when congestion occurs, but no paths can be adjusted, the controller borrows links from another domain. To build such cross-domain paths, two controllers exchange information through the XPub/XSub framework. This link borrowing mechanism results in *heterogeneous flows*, where a link carries the interior flows in its domain and the link-borrowing flows from exterior domains. Then, MCRM handles heterogeneous flows on the congested links by adopting OpenFlow meter and group tables. Simulation results show that MCRM can efficiently increase network throughput and decrease packet loss.**

***Keywords*—congestion, heterogeneous flow, multi-domain network, route management, software-defined networking (SDN).**

I. Introduction

For conventional networks, both the control and data planes are tightly coupled in each switch, where the former decides how to process packets, while the latter takes charge of packet forwarding. Hence, the network management is performed in a distributed manner, where users need to configure every involved switch individually to apply new policies or algorithms to the network, resulting in a high management cost [1].

The *software-defined networking (SDN)* technique can efficiently solve the above problem by decoupling the control and data planes [2]. The control plane is centralized in one entity called the *controller* and the data plane leaves in each switch. Switches report their statuses to the controller and obey its instructions. In this way, users can easily monitor the network state and change the behavior of switches through the controller, thereby facilitating network management. There have been manifold SDN-based applications developed, from Wi-Fi security [3] to data center management [4], anonymous authentication [5], and video streaming [6].

This paper considers using SDN to manage a large network with multiple domains (known as the *multi-domain network*), as shown in Fig. 1. Each domain is an autonomic subnetwork, whose switches are governed by an SDN controller. Since all domains belong to the same network, the controllers can cooperate to build "*cross-domain (CRD)*" paths for flows to facilitate packet routing. Practical examples include campus and enterprise networks. Some departments in a university or company have their local area networks pertaining to the campus or enterprise network [7]. Using multiple controllers to manage a multi-domain network brings benefits. First, each department can manage its domain (e.g., applying department-specific policies) [8]. Second, because multiple controllers co-manage the network, the load of each controller can decrease. Third, if a controller becomes busy (e.g., being attacked [9]), only its domain will be affected.

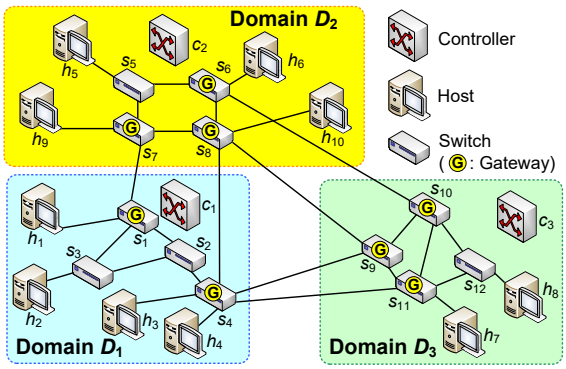


Fig. 1. Example of multi-domain network based on SDN.

In this paper, we propose the *multi-domain collaborative route management (MCRM)* scheme to efficiently schedule the routing paths of flows with different priorities. For each flow, the controller finds the shortest path with enough bandwidth in the domain. If some links are busy, the controller reroutes some flows on these links based on their priorities to mitigate congestion. However, once there is no substitute path available in the domain, the controller borrows links from a neighboring domain. To do so, the two controllers exchange information like gateway switches and flow priorities, which can be carried out by the Xpub/Xsub framework [10], to create a CRD path. Moreover, some links may carry *heterogeneous flows*, which include the interior flows in its domain and the link-borrowing

flows from other domains. If such links become congested, we use both meter and group tables in OpenFlow to deal with this situation. Through simulations by Mininet, we show that the MCRM scheme has higher network throughput and lower packet loss, as compared with existing solutions.

II. Related Work

Given a set of controllers, some studies assign a domain to each controller, which is known as the *controller placement* [11]. Hu et al. [12] develop a reliable placement method for controllers to maintain network stability and reduce packet loss. In [13], controllers are clustered into groups, where a master controller in each group decides switch assignment for the members to balance their loads. Wu et al. [14] use the deep reinforcement learning to allocate domains to controllers, so as to reduce data latency and achieve load balance. The work [15] chooses a low-load, high-throughput controller to be a leader to coordinate other controllers. Evidently, these studies have different objectives with this paper.

How to adjust the domain of each controller by switch migration is also discussed. Min et al. [16] transfer switches to different domains by Q-learning, which reduces the standard deviation of the loads of controllers. The work [17] picks the controller with the maximum response time, and then transfers its switch whose load is the heaviest to a controller with the minimum response time. A simulated-annealing approach is proposed in [18] to adjust domains to reduce the migration cost. To find target controllers for switch migration, the work [19] takes the memory size, CPU utilization, bandwidth, and hop count into account. Instead of changing each domain, which may make the network unstable and incur extra overheads, our work lets controllers cooperate to manage routes and mitigate congestion without modifying their domains.

A few studies address route management in a multi-domain network. The work [20] lets a controller share the topology of its domain (the entire topology or the connections between all gateways) with others. As the detailed topology of a domain is open, its autonomy may be broken. Also, controllers have to exchange many messages to depict the topologies of their domains. Assuming that a controller does not know the states of exterior domains, the study [21] proposes a *cooperative flow management (CFM)* method. If some links are congested and no substitute path is found in the local domain, the controller directly reroutes some flows to another domain. However, when congestion also occurs in the neighboring domain, these flows would encounter serious packet loss. In view of this, we develop the MCRM scheme to improve throughput of a multi-domain network, which allows controllers to exchange little information to build CRD paths and also uses group and meter tables to efficiently mitigate congestion.

III. System Model

Let us consider an SDN-based network comprising multiple non-overlapping domains. Each domain D_k contains a number of switches and hosts, which is coordinated by one controller c_k. Domains are interconnected with each other, in the sense that some switches in each domain connect with some switches in a neighboring domain. These switches are called *gateways*. Each switch s_i is OpenFlow-compliant and belongs to a domain. Hosts are connected to switches. Fig. 1 gives an example, where the network is divided into three domains D_1, D_2, and D_3 supervised by controllers c_1, c_2, and c_3, respectively. The gateways of D_1, D_2, and D_3 are $\{s_1, s_4\}$, $\{s_6, s_7, s_8\}$, and $\{s_9, s_{10}, s_{11}\}$, respectively.

Let $f_{x,y}$ be a flow whose source and destination are hosts h_x and h_y, respectively. When h_x and h_y reside in a domain D_k, controller c_k finds a complete routing path (from h_x to h_y) for $f_{x,y}$. If h_x and h_y are in domains D_k and D_l, respectively, c_k finds a path from h_x to a gateway linking to D_l, and controller c_l decides the residual path. Flow $f_{x,y}$ is given a priority $w_{x,y}$, where $w_{\min} \leq w_{x,y} \leq w_{\max}$ and $w_{x,y}, w_{\min}, w_{\max} \in \mathbb{Z}^+$.

A controller can get the status of each switch (e.g., topology and load) in its domain via the *OFPPortStatsRequest* function. Two controllers will exchange messages to construct a CRD path. For security concern and saving the message overhead, the messages disclose merely necessary information such as gateways and priorities. Thus, each controller just has a local view of its domain. Then, our objective is to let each controller schedule the routing paths for its flows and cooperate with other controllers to build CRD paths if necessary, to resolve congestion and improve network performance.

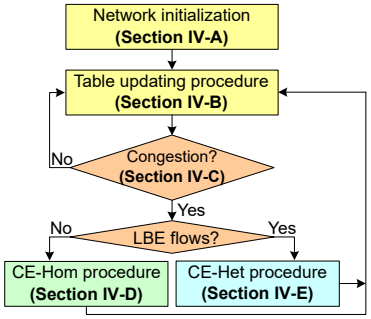


Fig. 2. Flowchart of the MCRM scheme.

IV. The Proposed MCRM Scheme

Fig. 2 shows MCRM's flowchart, which uses the procedure in Section IV-A to initialize the network. Every controller maintains three tables to obtain the latest status of its domain, which is carried out by the *table updating procedure* discussed in Section IV-B. By using the *congestion detecting procedure* mentioned in Section IV-C, the controller can check if some links become busy. If so, it further checks whether there exist link-borrowing flows from an exterior domain (below, we call such flows "*link-borrowing exterior (LBE)* flows") carried by the congested links. If not, the *congestion elimination with homogeneous flows (CE-Hom)* procedure in Section IV-D is used to reroute some flows on the congested links. Otherwise, the *congestion elimination with heterogeneous flows (CE-Het)* procedure in Section IV-E is employed to mitigate congestion.

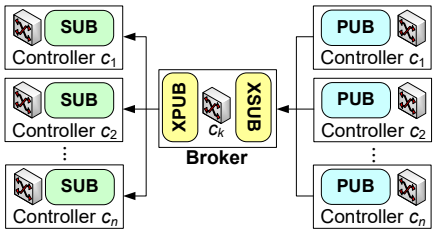


Fig. 3. Architecture of the Xpub/Xsub framework.

TABLE I
XPUB/XSUB MESSAGES TO BUILD AND MAINTAIN CRD PATHS.

message	publisher	parameters
help_request	c_k	domain, in_GW, out_GW, src_IP, dst_IP, flow_BW, priority
help_reply	c_l	domain, in_GW, out_GW, src_IP, dst_IP
adjustment_notice	c_l	domain, src_IP, dst_IP, share_BW

A. Network Initialization Procedure

Each switch s_i in a domain D_k recognizes the device linked to every port through the *link layer discovery protocol*, which can be (1) one host in D_k, (2) another switch in D_k, or (3) a gateway in an exterior domain (only if s_i is also a gateway). After that, s_i reports what it learns to controller c_k. In this way, c_k can know the complete topology in D_k and also the adjacent domain that each of its gateways connects to.

Controllers exchange messages by adopting the Xpub/Xsub framework, as shown in Fig. 3. Xpub/Xsub is based on the publish/subscribe (or simply pub/sub) model, which relies on one broker to relay the messages from each publisher to some subscribers. Specifically, the publisher will publish messages to a channel that a set of subscribers sign up to get them. Each controller can be both publisher and subscriber. Moreover, one controller serves as the broker to help relay messages.

Table I gives Xpub/Xsub messages related to CRD paths. If c_k wants to borrow links from a domain D_l to reroute flow $f_{x,y}$, it sends c_l (i.e., D_l's controller) a *help_request* message. For parameters, "domain" gives the domain's ID (i.e., D_k), "in_GW" / "out_GW" indicate D_k's incoming / outgoing gateways, "src_IP" / "dst_IP" denote $f_{x,y}$'s source / destination IP addresses, "flow_BW" is the bandwidth required by $f_{x,y}$, and "priority" is $w_{x,y}$. Then, c_l sends c_k a *help_reply* message, where "in_GW" and "out_GW" present D_l's incoming and outgoing gateways, respectively. The two messages are used in the CE-Hom procedure. However, if the borrowed links in D_l are busy, c_l then sends c_k an *adjustment_notice* message, where "share_BW" gives the amount of bandwidth that D_l offers to flow $f_{x,y}$ (where share_BW $<$ flow_BW). The detail will be discussed in Section IV-E.

B. Table Updating Procedure

Each controller c_k adopts three tables to keep track of the status in its domain D_k. Specifically, the *host table* helps c_k identify the hosts that it learns. For each host h_x in D_k, there is an entry $\langle h_x, a_x, s_i, p_j \rangle$ to record its information, where a_x is h_x's IP address and s_i is the switch that h_x attaches to (via port p_j). Fig. 1 gives an example, where we consider domain D_2. The entry for host h_5 is $\langle h_5, 10.0.2.5, s_5, 4 \rangle$, meaning that h_5 (whose IP address is 10.0.2.5) connects to switch s_5 via its port 4. When host h_x communicates with a host h_y in an exterior domain D_l, c_k also records h_y's information in the table. Here, c_k knows that h_y is an exterior host pertaining to D_l by using the subnet mask. In the entry $\langle h_y, a_y, s_i, p_j \rangle$ for h_y, s_i is D_k's gateway to contact h_y located in D_l (via its port p_j). Let us take host h_1 as an example, where its IP address is 10.0.1.1. Since h_1 is in domain D_1 (i.e., an exterior domain), its entry recorded by c_2 is $\langle h_1, 10.0.1.1, s_7, 3 \rangle$, which means that h_1 is contacted via gateway s_7 (using port 3).

TABLE II
EXAMPLE OF THE TRAFFIC TABLE (FOR DOMAIN D_2).

flow (src, dst)	priority	s_5	s_6	s_7	s_8
$f_{5,10}$ (h_5, h_{10})	1	2578	0	2403	1078
$f_{5,6}$ (h_5, h_6)	4	4585	4465	0	0
$f_{2,4}$ (h_2, h_4)	2	0	0	3500	2100

The *path table* records the path of each flow $f_{x,y}$, whose source and destination are hosts h_x and h_y, respectively. From c_k's viewpoint, there are four cases to be considered:

- **I.** *Both hosts are in the local domain D_k.* The entry records a complete path between h_x and h_y in D_k. For example, the entry for flow $f_{9,6}$ is $\langle s_7, s_5, s_6 \rangle$, meaning that the path is "$h_9 \rightarrow s_7 \rightarrow s_5 \rightarrow s_6 \rightarrow h_6$".
- **II.** *h_x is in D_k but h_y is in another domain D_l.* The entry records a path from h_x to g_v^k, where g_v^k is D_k's outgoing gateway to contact h_y in D_l. For example, the entry for flow $f_{10,7}$ is $\langle s_8 \rangle$, where host h_{10} attaches to switch s_8, which is also the outgoing gateway in domain D_2 to contact host h_7 in domain D_3.
- **III.** *h_x is in D_l and h_y is in D_k.* The entry records a path from g_u^l to h_y, where g_u^l is D_k's incoming gateway to contact h_x in D_l. For example, the entry for flow $f_{1,9}$ is $\langle s_7 \rangle$, as host h_9 links to switch s_7, which is also D_2's incoming switch to get host h_1's data from domain D_1.
- **IV.** *Both hosts are in D_l.* In this case, $f_{x,y}$ is an LBE flow of D_l, so c_k records the path from g_u^k to g_v^k, where g_u^k and g_v^k are D_k's incoming and outgoing gateways to contact h_x and h_y, respectively. Take flow $f_{1,4}$ in domain D_1 as an example. The entry recorded by c_2 is $\langle s_7, s_8 \rangle$, where s_7 and s_8 are D_2's incoming and outgoing gateways to contact h_1 and h_4 in D_1, respectively.

Notice that cases II and III are used to build an *inter-domain path* for a flow whose source and destination are in different domains, where the controller of each domain is responsible for finding the partial path between a host (either the source or the destination) and a gateway in its domain. Then, the inter-domain path will be a concatenation of these partial paths.

The *traffic table* records the number of packets of each flow $f_{x,y}$ successfully delivered by the switches in a domain. This table also indicates the flow's priority $w_{x,y}$. Table II gives an example, where the priority of flow $f_{5,10}$ is set to 1. Since

$f_{5,10}$'s path is "$h_5 \rightarrow s_5 \rightarrow s_7 \rightarrow s_8 \rightarrow h_{10}$", the traffic table shows that switches s_5, s_7, and s_8 delivered 2578, 2403, and 1078 packets for $f_{5,10}$ in the last period.

C. Congestion Detecting Procedure

Through the traffic table, each controller c_k can check if congestion occurs in its domain D_k. Suppose that the routing path of a flow $f_{x,y}$ passes m switches s_{α_1}, s_{α_2}, $\cdots$, and s_{α_m} in sequence. Moreover, the inter-switch *packet loss rate (PLR)* of $f_{x,y}$ between two switches s_i and s_j is defined by

$$\tilde{L}(f_{x,y}, s_i, s_j) = \frac{|\tilde{P}(f_{x,y}, s_i) - \tilde{P}(f_{x,y}, s_j)|}{\max\{\tilde{P}(f_{x,y}, s_i), \tilde{P}(f_{x,y}, s_j)\}}, \quad (1)$$

where $\tilde{P}(f_{x,y}, s_i)$ is the number of $f_{x,y}$'s packets successfully delivered by switch s_i in the previous period. Since the source h_x is the closest to s_{α_1} while the destination h_y is the closest to s_{α_m} in domain D_k, $\tilde{L}(f_{x,y}, s_{\alpha_1}, s_{\alpha_m})$ can be viewed as the overall PLR of $f_{x,y}$ in D_k. If $\tilde{L}(f_{x,y}, s_{\alpha_1}, s_{\alpha_m}) \geq \delta$, where δ is a predefined threshold and $0 < \delta < 0.5$, congestion occurs in D_k. In this case, c_k further finds out which links of $f_{x,y}$'s path are congested. To do so, for each link $(s_{\alpha_q}, s_{\alpha_{q+1}})$ between two adjacent switches s_{α_q} and $s_{\alpha_{q+1}}$ in $f_{x,y}$'s path, for $q = 1, \cdots, m-1$, if the following condition holds, c_k treats it as a congested link:

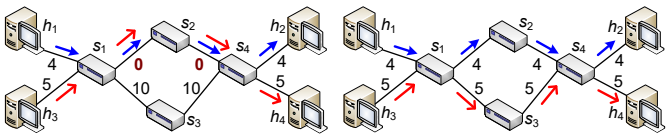
$$\tilde{L}(f_{x,y}, s_{\alpha_q}, s_{\alpha_{q+1}}) \geq \frac{\tilde{L}(f_{x,y}, s_{\alpha_1}, s_{\alpha_m})}{m-1}. \quad (2)$$

That is, the PLR caused by link $(s_{\alpha_q}, s_{\alpha_{q+1}})$ is above the average PLR of all links. Table II gives an example, where δ is set to 0.2. The overall PLR of flow $f_{5,10}$ in domain D_2 is $\frac{|2578-1078|}{\max\{2578,1078\}} > 0.2$, so congestion occurs in D_2. In addition, the inter-switch PLR between s_7 and s_8 is $\frac{|2403-1078|}{2403} > \frac{0.58}{3-1}$. Therefore, link (s_7, s_8) is considered as a congested link.

Depending on the type of flows carried by a congested link, the controller may change the paths of some flows to bypass that link, or limit the bandwidth usage of each flow on the link. Specifically, if the link carries homogeneous flows (i.e., cases I, II, and III), the controller adopts the CE-Hom procedure to eliminate congestion. Otherwise, there exist LBE flows (i.e., case IV), so the controller employs the CE-Het procedure to resolve congestion.

D. CE-Hom Procedure

Let $\hat{\mathcal{F}}$ be the set of flows carried by a congested link (s_i, s_j). As all flows in $\hat{\mathcal{F}}$ are homogeneous (in other words, the source or the destination of a flow is in the local domain D_k), controller c_k selects one flow from $\hat{\mathcal{F}}$ for rerouting, such that its new path does not include (s_i, s_j). The selection depends on the number of flows in $\hat{\mathcal{F}}$. If $\hat{\mathcal{F}}$ contains two flows, c_k selects the low-priority flow, so as to make the route of the high-priority flow stable. Otherwise, the flow with the highest priority, say, $f_{x,y}$ is selected for rerouting. In this case, since $\hat{\mathcal{F}}$ has three or more flows, c_k prefer finding a new path for $f_{x,y}$, so $f_{x,y}$ need not compete (s_i, s_j)'s bandwidth with other flows in $\hat{\mathcal{F}}$. In case of a tie, the flow with the largest size is selected, where the *size* of a flow is defined as the amount of bandwidth required by that flow.



(a) Original situation (b) Rerouting flow $f_{3,4}$

Fig. 4. Example of flow rerouting in the CE-Hom procedure.

Then, c_k finds an alternative path for $f_{x,y}$ based on two conditions: (1) As compared with $f_{x,y}$'s original path, the increase in hop counts of the alternative path is at most ξ, where ξ is a small, nonnegative integer (e.g., $\xi = 0$ or 1). (2) The *bottleneck bandwidth (BNB)* of the alternative path is no smaller than $f_{x,y}$'s size, where BNB is defined as the minimum residual bandwidth of all links not carrying $f_{x,y}$ in that path. In particular, condition (1) is to avoid significantly increasing the average packet latency of $f_{x,y}$, and condition (2) implies that the alternative path has enough bandwidth to support $f_{x,y}$. Fig. 4 gives an example, where the capacity of each link is 10 Mbps, and the sizes of flows $f_{1,2}$ and $f_{3,4}$ are 6 Mbps and 5 Mbps, respectively. Fig. 4(a) shows the original paths of both flows, which make links (s_1, s_2) and (s_2, s_4) congested. Supposing that $w_{1,2} > w_{3,4}$, the controller finds an alternative path "$h_3 \rightarrow s_1 \rightarrow s_3 \rightarrow s_4 \rightarrow h_4$" for $f_{3,4}$, where the BNB of this path (i.e., 10 Mpbs) is larger than $f_{3,4}$'s size (i.e., 5 Mbps). Fig. 4(b) presents the result after rerouting $f_{3,4}$.

However, when no alternative path can be found in the local domain D_k, c_k seeks to borrow some links from an adjacent domain D_l for rerouting flow $f_{x,y}$. More concretely, the link-borrowing mechanism involves the following steps:

[Step 1] Let g_u^k and g_v^k be D_k's gateways connecting with D_l and closest h_y and h_x, respectively. Then, c_k sends c_l message "[type=help_request, domain=D_k, in_GW=g_u^k, out_GW=g_v^k, src_IP=a_x, dst_IP=a_y, flow_BW=$s_{x,y}$, priority=$w_{x,y}$]", where $s_{x,y}$ is $f_{x,y}$'s size. Moreover, c_k finds a path $\mathcal{R}_1$ from h_x to g_v^k and also a path $\mathcal{R}_3$ from g_u^k to h_y.

[Step 2] Suppose that g_u^l and g_v^l are the two gateways in D_l connecting with g_v^k and g_u^k, respectively. If c_l can find a path $\mathcal{R}_2$ in D_l whose two endpoints are g_v^l and g_u^l, such that the path's BNB is larger than $s_{x,y}$ (i.e., $f_{x,y}$'s size), c_l lends c_k path $\mathcal{R}_2$ for rerouting flow $f_{x,y}$. In this case, c_l records $\mathcal{R}_2$ in its path table for $f_{x,y}$ (as discussed in Section IV-B), and sends c_k message "[type=help_reply, domain=D_l, in_GW=g_u^l, out_GW=g_v^l, src_IP=a_x, dst_IP=a_y]".

[Step 3] However, if c_l cannot find path $\mathcal{R}_2$, it sends message "[type=help_reply, domain=D_l, in_GW=null, out_GW=null, src_IP=a_x, dst_IP=a_y]" to c_k, meaning that c_l cannot help.

[Step 4] If c_k receives a positive reply from c_l (by step 2), c_k adds paths $\mathcal{R}_1$ and $\mathcal{R}_3$ to its path table. Consequently, a CRD path "$\mathcal{R}_1 \rightarrow \mathcal{R}_2 \rightarrow \mathcal{R}_3$" is built for flow $f_{x,y}$.

[Step 5] If c_k gets a negative reply from c_l (by step 3), c_k discards $\mathcal{R}_1$ and $\mathcal{R}_3$, and asks help from another domain.

The above steps are repeated until a CRD path is found (by step 4) or no domains can help. In the former case, c_k reroutes $f_{x,y}$ by using the CRD path. In the latter case, c_k limits the bandwidth usage of each flow on the congested link by the meter table. The detail will be discussed in Section IV-E.

Fig. 1 presents an example, where c_1 asks domain D_2 to help reroute flow $f_{1,4}$, whose size is 4 Mbps and priority is 3. In step 1, gateways g_u^k and g_v^k in D_1 are s_4 and s_1, respectively. Thus, path $\mathcal{R}_1$ is "$h_1 \rightarrow s_1$" and path $\mathcal{R}_3$ is "$s_4 \rightarrow h_4$". After that, c_1 sends c_2 message "[type=help_request, domain=D_1, in_GW=s_4, out_GW=s_1, src_IP=10.0.1.1, dst_IP=10.0.1.4, flow_BW=4, priority=3]", where the IP addresses of both hosts h_1 and h_4 are 10.0.1.1 and 10.0.1.4, respectively. In step 2, gateways g_u^l and g_v^l in D_2 are s_7 and s_8, respectively. Suppose that the BNB of path "$s_7 \rightarrow s_8$" (i.e., $\mathcal{R}_2$) is 8 Mbps. Thus, c_2 can lend c_1 path $\mathcal{R}_2$, and c_2 sends c_1 message "[type=help_reply, domain=D_2, in_GW=s_7, out_GW=s_8, src_IP=10.0.1.1, dst_IP=10.0.1.4]". By step 4, a CRD path "$h_1 \rightarrow s_1 \rightarrow s_7 \rightarrow s_8 \rightarrow s_4 \rightarrow h_4$" is finally built for rerouting $f_{1,4}$.

E. CE-Het Procedure

After controller c_l lends links to controller c_k, some links in domain D_l may carry heterogeneous flows, which contain D_l's interior flows and also the LBE flows from domain D_k. When the bandwidth demands of some flows carried by these links increase, the links could become congested. In this case, c_l first finds alternative paths in D_l to reroute some flows on these links (as mentioned in Section IV-D). However, when c_l cannot find alternative paths, it is infeasible to construct new CRD paths for rerouting flows again. The reason is that such CRD paths would pass three or more domains (including D_k and D_l), which complicates route management and increases packet latency. Instead, we adopt both meter and group tables provided by OpenFlow to deal with this situation.

Consider that one link (s_i, s_j) in domain D_l carries a set $\hat{\mathcal{F}}$ of heterogeneous flows. When the link becomes congested, c_l installs a meter table in switch s_i to restrict the bandwidth consumption of flows in $\hat{\mathcal{F}}$. More concretely, the amount of bandwidth used by each flow $f_{x,y}$ in $\hat{\mathcal{F}}$ cannot overtake the following bound:

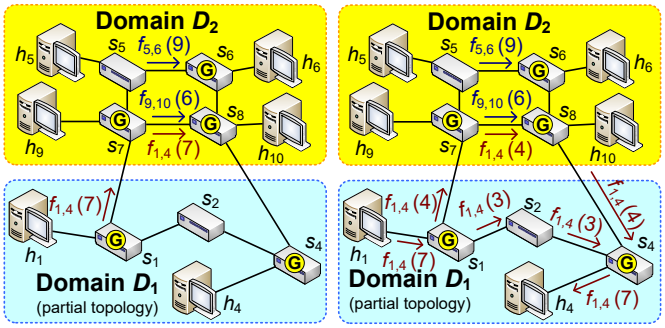
$$\tilde{B}(f_{x,y}) = \frac{w_{x,y}}{\sum_{f_{x',y'} \in \hat{\mathcal{F}}} w_{x',y'}} \times \lambda(s_i, s_j), \tag{3}$$

where $\lambda(s_i, s_j)$ is the capacity of link (s_i, s_j). In other words, the amount of link (s_i, s_j)'s bandwidth allocated to each flow $f_{x,y}$ is proportional to its priority $w_{x,y}$. Moreover, the priority of an LBE flow (from domain D_k) is updated to

$$w_{x,y} = \max\{w_{x,y} - \Delta_w, w_{\min}\}, \Delta_w \in \mathbb{Z}^+, \tag{4}$$

In this way, D_l's flows can be given more bandwidth of link (s_i, s_j). Then, c_l sends c_k message "[type=adjustment_notice, src_IP=a_x, dst_IP=a_y, share_BW=$\tilde{B}(f_{x,y})$]" to indicate that D_l offers at most $\tilde{B}(f_{x,y})$ bandwidth to D_k's LBE flow $f_{x,y}$.

As $\tilde{B}(f_{x,y}) < s_{x,y}$, where $s_{x,y}$ is $f_{x,y}$'s size, c_k has to find another path in the local domain D_k to offer $f_{x,y}$ the residual



(a) Original situation (b) Splitting LBE flow $f_{1,4}$

Fig. 5. Example of flow rerouting in the CE-Het procedure.

$(s_{x,y} - \tilde{B}(f_{x,y}))$ bandwidth. To do so, we use the group table to split flow $f_{x,y}$, which can specify the ratio of $f_{x,y}$'s packets sent via different ports of a switch (that is, delivering packets to different paths). Let s_z be the switch that h_x attaches to, whose ports p_{z_1} and p_{z_2} forward $f_{x,y}$'s packets to the CRD path to domain D_l (as denoted by $\mathcal{R}_1$) and the interior path in domain D_k (as denoted by $\mathcal{R}_2$), respectively. We set the budget weights for ports p_{z_1} and p_{z_2} to $\tilde{B}(f_{x,y})/s_{x,y}$ and $(s_{x,y} - \tilde{B}(f_{x,y}))/s_{x,y}$, respectively. In this way, a ratio $\tilde{B}(f_{x,y})/s_{x,y}$ of $f_{x,y}$'s packets will be sent by path $\mathcal{R}_1$, while the residual packets will be sent by path $\mathcal{R}_2$.

Fig. 5 presents an example, where some devices and links in domain D_1 are omitted for simplification. The number in brackets gives the amount of bandwidth used by a flow. There are three flows $f_{5,6}$, $f_{9,10}$, and $f_{1,4}$ in domain D_2, whose sizes are 9 Mbps, 6 Mbps, and 7 Mbps, paths are "$h_5 \rightarrow s_5 \rightarrow s_6 \rightarrow h_6$", "$h_9 \rightarrow s_7 \rightarrow s_8 \rightarrow h_{10}$", and "$h_1 \rightarrow s_1 \rightarrow s_7 \rightarrow s_8 \rightarrow s_4 \rightarrow h_4$", and priorities are 3, 3, and 4, respectively. Since the capacity of each link is 10 Mbps, link (s_7, s_8) will be thus congested. Moreover, controller c_2 cannot find any alternative path in D_2 to reroute the flows on link (s_7, s_8), as shown in Fig. 5(a). In this case, c_2 will install a meter table in switch s_7 to limit the bandwidth usage of flows $f_{9,10}$ and $f_{1,4}$ according to their priorities. Suppose that $\Delta_w = 2$ and $w_{\min} = 1$. Since $f_{1,4}$ is an LBE flow, its priority is updated to $w_{1,4} = 4 - 2 = 2$. Therefore, the amount of link (s_7, s_8)'s bandwidth allocated to $f_{9,10}$ and $f_{1,4}$ will be $\frac{3}{3+2} \times 10$ and $\frac{2}{3+2} \times 10$ (i.e., 6 Mbps and 4 Mbps, respectively). Then, c_2 sends c_1 message "[type=adjustment_notice, src_IP=10.0.1.1, dst_IP=10.0.1.4, share_BW=4]". After getting the message, c_1 installs a group table in switch s_1 to split flow $f_{1,4}$, where $\frac{4}{7}$ and $\frac{3}{7}$ of $f_{1,4}$'s packets are sent by paths "$\mathcal{R}_1 : s_1 \rightarrow s_7 \rightarrow s_8 \rightarrow s_4$" and "$\mathcal{R}_2 : s_1 \rightarrow s_2 \rightarrow s_4$", respectively.

V. Performance Evaluation

To evaluate the network performance, we adopt the Mininet simulator [22], where switches and controllers are respectively implemented by the OVS (Open vSwitch) module [23] and the Ryu SDN framework [24]. Fig. 1 gives the network topology considered in the simulation, where the capacity of each link is set to 10 Mbps. In addition, we employ the iPerf tool [25]

TABLE III
FLOW GENERATION IN THE SIMULATION.

flow	domain	size	start time	duration	priority
$f_{4,2}$	D_1	9 Mbps	10th second	90 s	3
$f_{3,1}$	D_1	9 Mbps	25th second	90 s	3
$f_{10,6}$	D_2	7 Mbps	40th second	70 s	2
$f_{7,8}$	D_3	9 Mbps	40th second	120 s	3
$f_{6,5}$	D_2	9 Mbps	115th second	225 s	1
$f_{10,9}$	D_2	9 Mbps	135th second	210 s	4
$f_{4,1}$	D_1	9 Mbps	150th second	195 s	5
$f_{3,2}$	D_1	9 Mbps	150th second	225 s	2
$f_{5,7}$	D_2, D_3	6 Mbps	315th second	225 s	3
$f_{10,8}$	D_2, D_3	6 Mbps	345th second	210 s	2
$f_{1,2}$	D_1	5 Mbps	375th second	180 s	3
$f_{7,4}$	D_3, D_1	9 Mbps	360th second	225 s	5
$f_{1,3}$	D_1	9 Mbps	555th second	300 s	4
$f_{9,10}$	D_2	6 Mbps	555th second	300 s	5
$f_{2,4}$	D_1	6 Mbps	630th second	300 s	3
$f_{5,6}$	D_2	9 Mbps	660th second	300 s	3

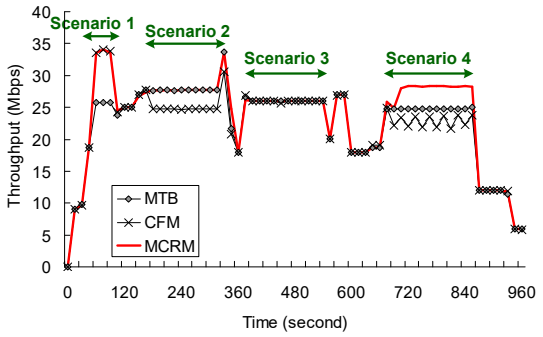


Fig. 6. Comparison of network throughput by different methods.

to generate UDP flows in the network, where the packet size is 1470 bytes. Table III presents the information of each flow. The total simulation time is 960 seconds.

We compare our MCRM scheme with two methods. One is the *meter table-based (MTB)* method [26], where a controller installs one meter table in each switch for route management in its domain. However, the controller will not find any path outside its domain for interior flows (i.e, without link borrowing). The other is the CFM method discussed in Section II. When there is no substitute path inside the local domain, CFM allows a controller to borrow some links from a neighboring domain. However, the controller simply directs the flow to the CRD path, without checking whether the neighboring domain has enough link bandwidth to do the help. In MCRM, we set the parameter Δ_m in Eq. (4) to 2.

Fig. 6 shows the amount of network throughput and Table IV lists the PLR of each flow. In particular, there are four different scenarios presented in the simulation:

Scenario 1 (25s–100s): Domain D_1 has two 9 Mbps flows $f_{4,2}$ and $f_{3,1}$, which make link (s_2, s_4) congested. In MTB, since no substitute path can be found in D_1 for rerouting $f_{4,2}$ or $f_{3,1}$ (to bypass the congested link), these two flows have pretty high PLRs (i.e., 35.1% and 43.8%). On the contrary, CFM and MCRM allow controller c_1 to borrow some links from D_2 (which has enough link bandwidth for help). Thus, they result in higher throughput and lower PLRs than MTB.

TABLE IV
PLR OF EACH FLOW (UNIT: %).

flow	MTB	CFM	MCRM	flow	MTB	CFM	MCRM
$f_{4,2}$	35.1	8.4	15.7	$f_{5,7}$	0.0	0.0	0.0
$f_{3,1}$	43.8	24.7	17.5	$f_{10,8}$	0.0	0.0	0.0
$f_{10,6}$	0.0	0.0	0.0	$f_{1,2}$	0.0	0.0	0.0
$f_{7,8}$	0.0	0.0	0.0	$f_{7,4}$	0.0	0.0	0.0
$f_{6,5}$	0.0	7.1	0.0	$f_{1,3}$	22.1	39.7	12.7
$f_{10,9}$	2.3	21.0	0.1	$f_{9,10}$	0.0	8.1	0.0
$f_{4,1}$	44.3	56.4	24.5	$f_{2,4}$	33.0	5.6	11.9
$f_{3,2}$	19.2	3.6	39.1	$f_{5,6}$	0.0	7.0	1.3

Since $f_{4,2}$ and $f_{3,1}$ have the same priority (i.e., 3), MCRM allocates equal link bandwidth to them based on Eq. (3). That is why these two flows have similar PLRs in MCRM.

Scenario 2 (150s–345s): There are two 9 Mbps flows $f_{4,1}$ and $f_{3,2}$ in D_1, whose paths have common links. Evidently, these links must be congested. Based on the topology in Fig. 1, only D_2 can lend links to $f_{4,1}$ and $f_{3,2}$. Unfortunately, the two 9 Mbps flows $f_{6,5}$ and $f_{10,9}$ consume most of the link bandwidth in D_2 (but they do not cause congestion). For the CFM method, controller c_1 directs $f_{4,1}$ to D_2. Doing so can greatly reduce $f_{3,2}$'s PLR (i.e., 3.6%). However, $f_{4,1}$ competes with $f_{10,9}$ for the bandwidth of link (s_7, s_8), thereby degrading throughput and rising their PLRs (i.e., 56.4% and 21.0%). In our MCRM scheme, controller c_2 sends a negative reply to c_1 (i.e., step 3 in the CE-Hom procedure). Thus, c_1 will not build a CRD path for $f_{4,1}$. Instead, c_1 installs meter tables in some switches to allocate the bandwidth of common links to $f_{4,1}$ and $f_{3,2}$ according to Eq. (3). Thus, MCRM has similar throughput with MTB. Since $f_{4,1}$ has a higher priority than $f_{3,2}$, MCRM gives $f_{4,1}$ more link bandwidth than $f_{3,2}$. Thus, $f_{4,1}$ has a smaller PLR than $f_{3,2}$ in MCRM, which reflects their priorities.

Scenario 3 (375s–555s): Three inter-domain flows (including $f_{5,7}$, $f_{10,8}$, and $f_{7,4}$) and one local flow (i.e., $f_{1,2}$ in domain D_1) coexist in the network. Because there could be multiple choices of paths, each method prefers finding non-overlapping paths for these flows. In this way, they will not compete for the bandwidth of any link. The result shows that MTB, CFM, and MCRM work well on finding paths for inter-domain flows in this case, where they can achieve the highest throughput and no flow will encounter packet loss.

Scenario 4 (630s–855s): At the 555th second, one 9 Mbps flow $f_{1,3}$ and a 6 Mbps flow $f_{9,10}$ are generated in domains D_1 and D_2, respectively. Then, a 6 Mbps flow $f_{2,4}$ is added to D_1, which competes with $f_{1,3}$ for the bandwidth of link (s_2, s_4). The MTB method uses a meter table to let $f_{1,3}$ and $f_{2,4}$ share link (s_2, s_4)'s bandwidth. In this case, these two flows have higher PLRs. In the CFM method, controller c_1 seeks help from domain D_2, and a CRD path $\mathcal{R}_1 : h_1 \rightarrow s_1 \rightarrow s_7 \rightarrow s_5 \rightarrow s_6 \rightarrow s_8 \rightarrow s_4 \rightarrow h_3$ is built for $f_{1,3}$. At the 660th second, a 9 Mbps flow $f_{5,6}$ is added to D_2. Since CFM treats local and LBE flows in the same way, controller c_2 changes $f_{1,3}$'s path to $\mathcal{R}_2 : h_1 \rightarrow s_1 \rightarrow s_7 \rightarrow s_8 \rightarrow s_4 \rightarrow h_3$. However, doing so makes $f_{1,3}$ and $f_{9,10}$ compete

for link (s_7, s_8)'s bandwidth. Unavoidably, $f_{1,3}$'s path will be switched back and forth between $\mathcal{R}_1$ and $\mathcal{R}_2$. That explains why the throughput in CFM swings. On the other hand, our MCRM scheme differentiates between local flows and LBE flows, so c_2 will not change $f_{1,3}$'s path. Instead, the CE-Het procedure in MCRM splits $f_{1,3}$ and sends its packets via different paths to mitigate congestion. In this way, MCRM can significantly improve throughput and lower PLRs of flows, as compared with the MTB and CFM methods.

According to Table IV, the average PLR of flows is 12.49%, 11.35%, and 7.68% in MTB, CFM, and MCRM, respectively. This result shows that MCRM can mitigate congestion more efficiently than both MTB and CFM.

VI. Conclusion

In this paper, we propose the MCRM scheme to efficiently schedule the routing paths of flows in a multi-domain network based on SDN. Each controller can keep an updated view of its domain by maintaining the host, path, and traffic tables. In case of congestion, the controller will modify the paths of some flows according to their priorities. However, once there is no alternative path available in the domain, the controller borrows links from an adjacent domain. In this case, the two controllers negotiate with each other through the XPub/XSub framework to create a CRD path. If a congested link carries heterogeneous flows, the switch uses a meter table to allot the link's bandwidth to these flows by referring to their priorities. With the group table, an LBE flow can use different paths for packet delivery. Through simulations, we show that MCRM achieves higher throughput and lower PLRs, as compared with both MTB and CFM. For future work, we will consider how to make controllers collaborate to resist cyberattacks such as the HTTP slow attack [27] in a multi-domain network.

Acknowledgment

This work was supported by the Ministry of Science and Technology, Taiwan under Grant 108-2221-E-110-016-MY3 and the Information Security Research Center, National Sun Yat-sen University, Taiwan.

References

[1] N. Anerousis, P. Chemouil, A. A. Lazar, N. Mihai, and S. B. Weinstein, "The origin and evolution of open programmable networks and SDN," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 3, pp. 1956–1971, 2021.

[2] 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.

[3] P. Shrivastava, M. S. Jamal, and K. Kataoka, "EvilScout: Detection and mitigation of evil twin attack in SDN enabled WiFi," *IEEE Transactions on Network and Service Management*, vol. 17, no. 1, pp. 89–102, 2020.

[4] 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.

[5] W. Iqbal, H. Abbas, P. Deng, J. Wan, B. Rauf, Y. Abbas, and I. Rashid, "ALAM: Anonymous lightweight authentication mechanism for SDN-enabled smart homes," *IEEE Internet of Things Journal*, vol. 8, no. 12, pp. 9622–9633, 2021.

[6] A. Ahmad, A. Floris, and L. Atzori, "Timber: An SDN-based emulation platform for experimental research on video streaming," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 7, pp. 1374–1387, 2020.

[7] Y. W. E. Sung, X. Sun, S. G. Rao, G. G. Xie, and D. A. Maltz, "Towards systematic design of enterprise networks," *IEEE/ACM Transactions on Networking*, vol. 19, no. 3, pp. 695–708, 2011.

[8] Y. C. Wang and H. Hu, "A Low-cost, high-efficiency SDN framework to diminish redundant ARP and IGMP traffics in large-scale LANs," in *IEEE Computer Software and Applications Conference*, 2018, pp. 894–903.

[9] Y. C. Wang and Y. C. Wang, "Efficient and low-cost defense against distributed denial-of-service attacks in SDN-based networks," *International Journal of Communication Systems*, vol. 33, no. 14, pp. 1–24, 2020.

[10] ZeroMQ Guide. [Online]. Available: https://zguide.zeromq.org/

[11] J. Lu, Z. Zhang, T. Hu, P. Yi, and J. Lan, "A survey of controller placement problem in software-defined networking," *IEEE Access*, vol. 7, pp. 24 290–24 307, 2019.

[12] T. Hu, J. Zhang, L. Cao, and J. Gao, "A reliable controller deployment strategy based on network condition evaluation in SDN," in *IEEE International Conference on Software Engineering and Service Science*, 2017, pp. 367–370.

[13] H. Sufiev, Y. Haddad, L. Barenboim, and J. Soler, "Dynamic SDN controller load balancing," *Future Internet*, vol. 11, no. 3, pp. 1–21, 2019.

[14] Y. Wu, S. Zhou, Y. Wei, and S. Leng, "Deep reinforcement learning for controller placement in software defined network," in *IEEE INFOCOM Workshop*, 2020, pp. 1254–1259.

[15] W. H. F. Aly, "Controller adaptive load balancing for SDN networks," in *International Conference on Ubiquitous and Future Networks*, 2019, pp. 514–519.

[16] Z. Min, Q. Hua, and Z. Jihong, "Dynamic switch migration algorithm with Q-learning towards scalable SDN control plane," in *International Conference on Wireless Communications and Signal Processing*, 2017, pp. 1–4.

[17] J. Cui, Q. Lu, H. Zhong, M. Tian, and L. Liu, "SMCLBRT: A novel load-balancing strategy of multiple SDN controllers based on response time," in *IEEE International Conference on High Performance Computing and Communications*, 2018, pp. 541–546.

[18] T. Hu, J. Lan, J. Zhang, and W. Zhao, "EASM: Efficiency-aware switch migration for balancing controller loads in software-defined networking," *Peer-to-Peer Networking and Applications*, vol. 12, pp. 452–464, 2019.

[19] K. S. Sahoo, D. Puthal, M. Tiwary, M. Usman, B. Sahoo, Z. Wen, B. P. S. Sahoo, and R. Ranjan, "ESMLB: Efficient switch migration-based load balancing for multicontroller SDN in IoT," *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 5852–5860, 2020.

[20] S. Civanlar, E. Lokman, B. Kaytaz, and A. M. Tekalp, "Distributed management of service-enabled flow-paths across multiple SDN domains," in *European Conference on Networks and Communications*, 2015, pp. 360–364.

[21] Y. C. Wang and E. J. Chang, "Cooperative flow management in multi-domain SDN-based networks with multiple controllers," in *IEEE International Conference on Smart Communities: Improving Quality of Life Using ICT, IoT and AI*, 2020, pp. 82–86.

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

[23] OVS. [Online]. Available: https://www.openvswitch.org

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

[25] iPerf. [Online]. Available: https://iperf.fr/

[26] H. Krishna, N. L. M. van Adrichem, and F. A. Kuipers, "Providing bandwidth guarantees with OpenFlow," in *IEEE Symposium on Communications and Vehicular Technologies*, 2016, pp. 1–6.

[27] Y. C. Wang and R. X. Ye, "Credibility-based countermeasure against slow HTTP DoS attacks by using SDN," in *IEEE Annual Computing and Communication Workshop and Conference*, 2021, pp. 890–895.