Efficient Big Link Allocation Scheme in Virtualized Software-defined Networking

Wontae Jeong†, Gyeongsik Yang†, Seong-Mun Kim, Chuck Yoo
College of Informatics, Korea University, Seoul, South Korea
Email: wtjeong@os.korea.ac.kr, ksyang@os.korea.ac.kr, soulcrime@korea.ac.kr, chuckyoo@os.korea.ac.kr

***Abstract*—We propose an efficient resource allocation scheme for big links in virtualized software-defined networking. Network virtualization based on software-defined networking provides big link concept to facilitate simple network management – big link maps a set of switches and links into a single virtual link. However, this paper reports an issue of the big link in that there is a severe performance degradation in virtualized SDN environments. We find the cause: the existing network hypervisors do not consider the network traffic when allocating physical resources to a big link. To address this issue, we present big link allocation scheme (BAS) that considers network traffic when allocating and reallocating resources to a big link. A prototype implementation is done with OpenVirteX, and experiments demonstrate that the big link with BAS achieves four times greater throughput than that of the big link without BAS. Moreover, by including a timer in OpenVirteX, the BAS decreases unnecessary resource reallocations, which reduces overhead.**

I. Introduction

Network virtualization has become an important technique for providing various services over a single physical network. Many widely used virtualization techniques, such as VxLAN [1], GRE [2], MPLS [3], have been proposed; however, fine control and flexible programming of each virtual network remain challenging. Due to the distributed nature of physical networks, most previous studies relied on tunneling between heterogeneous networks. However, software-defined networking (SDN) presents new possibilities for network virtualization. SDN decouples the control and data planes and centralizes control functions in the controller. With this centralized entity, global resource management becomes possible, and many virtualization methods [4], [5], [6], [7], [8] with centralized control planes have been proposed. They provide abstractions such as virtual topology, address, and policy for each tenant.

Many network virtualization solutions provide big link and big switch for virtual topology. This paper focuses on the big link which is a topology abstraction for a set of physical resources, such as physical links and switches. With the big link, each tenant can create an arbitrary virtual network and its topology. Especially, by selecting only physical resources the tenant wants to use from the entire physical network, network virtualization can reduce the network size. In addition, a big link that abstracts multiple physical resources into one virtual resource further reduces the management complexity.

Previous study [6] has attempted to provide big links that exploit these advantages; however, such study focused on generating and providing the big links themselves. The big resource needs physical resource allocation, but the previous study calculates the allocable resources without consideration of physical network traffic. In Section III, we show that such study results in approximately 50% performance degradation compared to optimal allocations.

To overcome the link throughput degradation from the physical resource allocation, we propose a big link allocation scheme (BAS). The goal of the BAS is to provide efficient resource allocation to increase big link throughput. The proposed BAS consists of three phases: initialization (BAS-I), allocation of physical resources (BAS-II), and reallocation (BAS-III). Moreover, we develop a link throughput measurement and cost calculation process for the resource allocation in consideration of network traffic. With this process, BAS-II attempts to efficiently find optimal resources to be allocated to a big link at a given time. In addition, we propose BAS-III to increase throughput by reallocating physical resources when traffic load from other virtual networks are created after physical resources are allocated to the big link. In addition, to reduce reallocation overhead, we propose loss avoidance and synchronous hard reallocation technique that reduce packet loss and traffic fluctuations efficiently.

We implement BAS on the OpenVirteX (OVX) [6], which is an SDN-based network hypervisor and evaluate BAS using an emulated physical network [9]. The experimental results demonstrate that the proposed BAS improves performance degradations by four times compared to the original OVX and does not increase overhead significantly.

II. Background

SDN-based virtualization is implemented through a network hypervisor located between network controllers and switches. To switches, network hypervisor is viewed as a controller, and simultaneously as switches to the network controllers. Therefore, the primary functions of the network hypervisor are to provide physical network abstractions and translate management messages from controllers and vice versa. The

†The first two authors contributed equally to this work.

*This work was supported by Institute for Information & communications Technology Promotion(IITP) grant funded by the Korea government(MSIT) (2016-0-00124, Research of Network Virtualization Platform and Service for SDN 2.0 Realization). This research was also supported by the MSIT(Ministry of Science and ICT), Korea, under the SW Starlab support program(2015-0-00280) supervised by the IITP(Institute for Information & communications Technology Promotion).

main abstractions used in network hypervisors are virtual addresses and topologies.

Each virtual network can select arbitrary address for their host, which is called virtual address for the host. Therefore, the network hypervisor should translate virtual addresses to avoid address conflicts between different virtual networks. The translation is called address virtualization, and various methods [6], [10] have been proposed. The OVX uses one-to-one address mapping for IP address virtualization. OVX allocates one physical IP address for each virtual IP address. The upper 8 bits of the physical IP address signify the virtual network (tenant ID) and the remaining bits are used to identify the hosts in each virtual network (host ID).

In addition, a virtual topology can be created over the physical network topology. The components of a virtual topology are virtual switches, virtual links, and hosts. Virtual links can be divided into two groups: one-to-one and big link. A one-to-one link maps a single physical link to a virtual link. Network slicing is a representative example of a one-to-one link. A big link maps a path on the virtual link. Thus, the big link contains a set of physical switches and links. To realize the abstraction, the network hypervisor is tasked with finding and allocating an appropriate path to transfer data between two given physical ports. In specific, creating a big link in the network hypervisor is performed as follows: 1) initialize the big link abstractions, 2) calculate the allocable physical resources to the big link, and 3) install the flow rules for each traffic to the allocated switches reactively.

III. Big link Throughput Degradation

Here, we discuss the motivation for our study. In a multi-tenant network where various virtual networks co-exist within a single physical network, traffic from each virtual network can affect other virtual networks. However, the previous studies do not consider about it. The previous network hypervisor, OVX, uses the shortest path first routing algorithm to get allocable physical resources to the big link. However, the cost for each physical link of the routing algorithm is identical. Moreover, once the physical resources are allocated to the virtual link, no reallocation occurs even if other physical network resources become idle.

To observe the influence of the design, we experiment the link throughput of a big link as follows. We use Mininet [9] for physical network emulation and the OVX network hypervisor, which provides the richest topology abstraction. Fig. 1 shows the experimental topology. We create the first virtual network (VN1) in the linear form (H1–S1–S2–S3–S4–S5–S10–S11–H3). Then, Iperf3 is used to generate traffic flows between hosts H1 and H3. After the traffic flows on VN1, we create another virtual network (VN2) between hosts H2 and H4. This topology includes two switches and a single big link in the linear form (H2–S1–S11–H4). After creating VN2, Iperf3 is used to generate traffic flows between hosts H2 and H4, as well as to measure the maximum throughput between these hosts. To observe the effect of the physical network traffic,

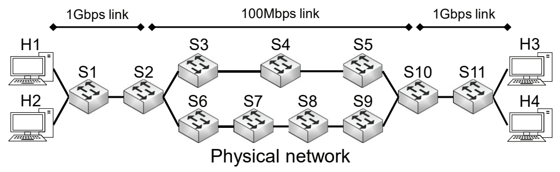

Fig. 1. Topology for big link throughput experiments

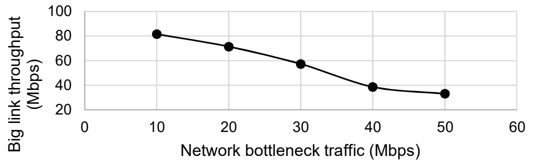

Fig. 2. Big link throughput degradation due to the traffic load on other virtual networks

we vary the load of the VN1. Fig. 2 shows the throughput of the big link in the VN2 for various traffic load of VN1.

Since the network hypervisor does not reflect the increased VN1's traffic to its routing algorithm, in the physical topology shown in Fig. 1, the big link is mapped to S2–S3–S4–S5–S10. When the traffic loads of VN1 increase, the big link throughput decreases, because the VN1 and VN2 both shares the same physical resources. The better throughput can be achieved if the big link is mapped to switches S2–S6–S7–S8–S9–S10, which are idle.

The reason for this throughput decline is that the network hypervisor does not consider existing traffic when allocating resources and does not alter the allocated resources. To address the issues, the proposed BAS involves the following.

- Traffic-aware resource allocation schemes for a big link
- Reallocation of physical resources to overcome performance degradation caused by traffic flows in other virtual networks
- Overhead reduction in the resource reallocation process

IV. Design

The proposed BAS comprises three phases: BAS-I (initialization), BAS-II (allocation), and BAS-III (reallocation). We discuss each phase in the following subsections.

A. BAS-I: Initialization

The BAS-I phase is responsible for creating a big link when requested by the network manager. When the virtual network manager requests the creation of a big link between arbitrary physical ports (e.g., P_i and P_j), BAS-I creates new virtual ports VP_i and VP_j mapped to P_i and P_j. Then, the created big link BL_n is directly connected to VP_i and VP_j. Physical resources must be allocated between the virtual ports before any traffic can flow.

B. BAS-II: Allocation

In the BAS-II phase, the actual path between two virtual ports is allocated to the big link. The big link is created in the BAS-I phase; however, this does not mean that any flow

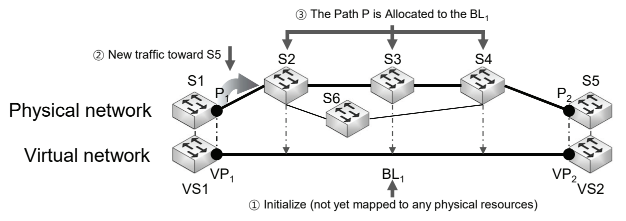
③ The Path P is Allocated to the BL1
② New traffic toward S5
S1
S2
S3
S4
S5
S6
P1
P2
Physical network
Virtual network
VP1
BL1
VP2
VS1
VS2
① Initialize (not yet mapped to any physical resources)

Fig. 3. Big link initialization and allocation

passes through the big link immediately. To allocate resources in consideration of the actual network traffic, resources must be calculated when the flow is created. Therefore, we allocate physical resources when a flow rule request message for the big link arrives.

To consider the network traffic in the resource allocating process, the network hypervisor should be aware of the link throughput of the entire physical network. In SDN, network statistics can be gathered using a southbound interface, such as OpenFlow [11]. Regarding the link throughput, we use RX and TX throughput values of each physical port.

Here, we consider the virtual network environment shown in Fig. 3. A big link BL_1 is not yet mapped to any physical resource. If new traffic flows from S1 to S5, S1 sends a flow rule request message to the network hypervisor. Then, the hypervisor looks up the virtual switch that corresponds to the physical switch that receives the message. Because S1 is mapped to VS1 and the packet belongs to the virtual network, the flow rule request message is delivered to the virtual network's controller. After receiving the flow rule request message, the controller sends a flow rule to the network hypervisor. Naturally, the flow rule dictates that traffic is sent to VP_1. The network hypervisor installs the rule in the physical network. Then, as packets are forwarded by S1, switch S2 requests a new flow rule for the network hypervisor. The network hypervisor can identify packets that belong to the virtual network because the allocated physical IP of the packet contains the tenant ID. However, S2 is not mapped to any other virtual switches in the virtual network. This is considered a signal to allocate a resource to the unallocated big link in the virtual network. Then, BAS-II is initiated and the hypervisor calculates link costs [12] at that point as follows.

$$link\ cost = \frac{1}{Link\ capacity - Link\ bitrate} \tag{1}$$

Using the link cost, the hypervisor calculates the shortest path between the given two arbitrary physical ports by Dijkstra's algorithm. As a result, the hypervisor can obtain the optimal path to be allocated to the requested big link at that point. Relative to the example shown in Fig. 3, assume that the optimal resource path P is {S2, S3, S4}. Then, the Path P is allocated to the big link BL_1.

After the allocation of physical resources, flow rules required to forward traffic must be installed to the allocated resources. To configure the packet-forwarding policy, we introduce an integrated inside rule (Section IV-C1).

C. BAS-III: Reallocation

In addition to allocating resources in consideration of network traffic, we propose the BAS-III resource reallocation scheme, which is similar to traffic load balancing. The goal of BAS-III is to enhance throughput of the big link by reallocating physical resources. However, the reallocation of the physical resources means that the path of the packet transfer changes, because the allocated path to the big link alters. Thus, such reallocation introduces new overheads. For example, changes to the packet transmission path cause packet-ordering issues, which results in significant performance degradation. Moreover, packets may be lost during the resource reallocation process. To resolve the issues, we implement the following.

- Soft reallocation for reducing packet ordering issues
- Asynchronous hard reallocation for periodic path recalculation and Loss avoidance to eliminate packet loss
- Synchronous hard reallocation for reducing reallocation overhead

1) Soft reallocation: We introduce soft reallocation to avoid corruption of the packet transmission sequence. Soft reallocation reallocates physical resources within the big link when there is no traffic on the link for a certain period. The scheme assumes that data always flows on the same path at given time intervals. If a packet that matches the inside rule of the big link does not occur for a given period, the switch deletes the flow rule and notifies the event to the hypervisor. Since there is no flow in the big link, the proposed BAS returns to the BAS-II phase and reallocates resources when a new flow occurs in the big link.

However, flow rules are generated for each flow that enters the big link and installed individually in the existing hypervisor design. The resource we focus on is a link; thus, flow rules to be installed to the switches matches on various header fields but performs same things – forward the packet to the next switch. To solve this efficiently, we introduce the integrated inside rule. With this rule, the switch allocated to the big link only matches a packet with the switch's input port and the upper 8 bits of the source IP address, i.e., the tenant ID. If the packet matches these elements, it is transferred to the output port of the switch, i.e., the next switch of the allocated resources at the big link. If the integrated inside rule is already installed to the allocated physical resource of the big link, no additional configuration is required for the switch. Besides, the switch consumes much less memory for the flow table.

2) Asynchronous hard reallocation: Another concern is that, if traffic flows without interruption, soft reallocation will never occur. In other words, the reallocation process can suffer a sort of starvation. Thus, we introduce hard reallocation, which is a method for performing resource reallocation unconditionally after a specified time. Hard reallocation can be designed in two ways. The first is to have the switch remove the configuration installed by the network hypervisor after a certain period. The switch then requests a new resource allocation from the network hypervisor. This approach is referred to as asynchronous reallocation.

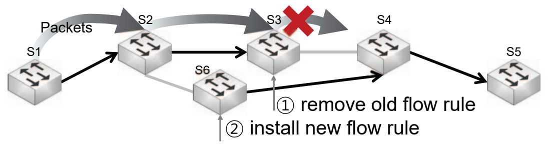

Fig. 4. Direction of packet transfer policy removal

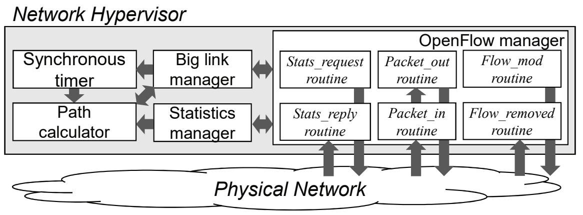

Fig. 5. BAS structure

However, packet transmission is not possible while sending the reallocation request, calculating the path for resources, and performing the allocation. The resulting packet loss and performance degradation are inevitable. To address this situation, we introduce loss avoidance, which buffers packets in the network hypervisor. When no packet forwarding policies are set at the physical switch, the switch sends all unmatched packets to the network hypervisor. This packet loss problem is easily solved by buffering and sending such packets to the last physical switch mapped to the big link.

3) Synchronous hard reallocation: Note that asynchronous hard reallocation always triggers a reallocation process for every determined period. In other words, even when the optimal physical resources have not changed relative to the resources allocated for the big link, the resources are removed for each period due to the asynchronous nature of the process. To address this problem, we propose a synchronous method. By implementing a timer that fires at a regular interval for each big link in the network hypervisor, reallocation is only performed when there is a change relative to previously allocated resources.

In addition, for the synchronous method, the order of the installation and deletion of a packet transfer policy affects performance. Specifically, if flow rule installation and network configuration for a newly allocated resource are performed prior to the deletion of previously allocated resources, network traffic can be transferred to the newly allocated resources without delayed configuration. Moreover, removing the previously allocated resources requires additional caution. If flow rules installed to the allocated path are deleted in reverse order of the packet forwarding direction, switches not yet configured will still transmit packets according to the old policy, thereby resulting in packet loss. For example, in Fig. 4, the packet that conforms to the previously allocated policy is lost in S3 because the policy is deleted in the middle of the resource reallocation process as the policy is removed in the reverse packet transmission direction.

V. Implementation

In this section, we explain the BAS implementation and discuss how each feature achieves the design goals described in Section III. We implement a prototype based on OVX, which uses OpenFlow as its southbound interface. The BAS does not depend on any particular OVX feature; thus, it can be implemented using any other SDN hypervisor. Fig. 5 shows the overall structure of the BAS. Its features can be divided into three main components: 1) big link management, 2) path calculation to be allocated to the big link, and 3) resource allocation. We describe each component as follows.

A. Big link management

When a virtual network manager asks the hypervisor to create a big link, the "Big link manager" is employed to collectively manage its related content, such as connected virtual ports. This is related to the BAS-I phase (Section IV-A). This component also manages information about the allocated resources in the BAS-II and BAS-III phases. With this information, further implementations, i.e., forwarding policy configuration and the reduction of reallocation overhead, are performed.

B. Path calculation

The "Path calculator" and "Statistics manager" are responsible for calculating the path allocated to big links in the BAS-II and BAS-III phases. To reflect the network traffic of the physical network, the "Statistics manager" periodically gathers the amount of data sent and received over physical ports using the "Stats_request routine" and the "Stats_reply routine." Note that the regular interval of statistics gathering follows the network hypervisor's policy as discussed in [13]. With the gathered data, the "Statistics manager" calculates the average link bitrate of each physical link. When the "Path calculator" requests the link bitrate at a specific point in time, the "Statistics Manager" returns the calculated value.

The "Path calculator" is called when the big link has not been mapped to any physical resource or reallocation is requested from other components. This component calculates the path to be allocated for the big link using the shortest path algorithm with costs that the "Statistics Manager" returns.

C. Resource allocation

As mentioned previously, after the BAS-I phase, physical resources are not allocated to the big link. When initial traffic occurs in the big link, the message from the switch goes to the "Packet_in routine," which calls the "Big link manager" and "Path calculator" in order. After the "Path calculator" returns the resources to be allocated to the big link, the "Big link manager" calls the "Flow_mod routine" to configure policies for packet transfer within the big link. The packet forwarding policy is realized as flow rules. To realize a soft reallocation design, we implement flow rules with an idle timeout value. When there is no matched traffic for the flow rule, the flow rule is removed after a designated idle timeout. The BAS installs only one rule to internal resources (integrated internal rule concept discussed in Section IV-C); thus, the allocated physical resource and its configuration are removed after the idle timeout value. Then, the switch sends a message to notify

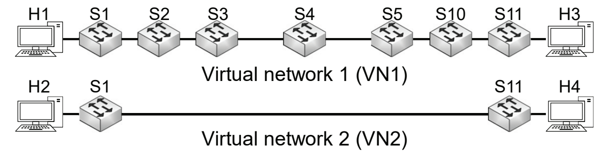

Fig. 6. Topology for experiments

the event, and the message is processed by the "Flow_removed routine."

In addition, we implement synchronous hard reallocation as the "Synchronous timer" initializes the corresponding big link's hard reallocation timer. When the hard reallocation timer of the big link fires, the timer calls the "Path calculator" component to calculate the optimal path at that point and resets timeout value. After the calculation, the routine compares the pre-allocated resources and performs resource reassignment only when there are changes to the newly computed resources and previously allocated resources. If there are changes, new resources are updated by the "Big link manager." Here, the order is important (Section IV-C3). The installation of the new policy proceeds before deletion of the existing policy. In addition, the direction of the flow rule is in reverse order of the packet forwarding direction in the big link; thus, existing packets in the big link can be transmitted according to the existing policy.

We additionally implement asynchronous hard reallocation in an optional mode to experimentally validate performance enhancement obtained by synchronous hard reallocation. In this manner, the flow rule is removed even if there is traffic. Then, the network hypervisor notifies the hard timeout and reallocates physical resources. During the flow rule reset process, all packets flowing in the big link are lost. To avoid this issue, we implement a loss avoidance design in the "Packet_out routine."

VI. Evaluation

In this section, we evaluate the performance of the proposed BAS and hard timeout overhead, i.e., the so-called reallocation overhead. We evaluate the BAS in three ways: 1) initial resource allocation performance, 2) resource reallocation performance for traffic load balancing, and 3) overhead for resource reallocation per design.

A. Experimental Setup

To evaluate the performance of the proposed BAS, we prepare three server machines connected by a 10-GbE switch. The first machine employs Mininet to emulate a physical network, and the second machine employs multiple ONOS [14] controllers for virtual network management. In addition, we run the network hypervisor with BAS implemented on another machine. To analyze the performance improvement obtained by the BAS, we use OVX for comparison.

We use the physical network in Fig. 1, and the topology has 11 physical switches and two hosts are connected to each edge switch. The network hypervisor creates two virtual networks as Fig. 6. We create a linear topology comprising seven one-to-one virtual switches connected by one-to-one virtual links and two physical hosts for virtual network 1 (VN1), and a linear topology comprising two one-to-one virtual switches, one big link, and two physical hosts for virtual network 2 (VN2). VN1 is used to add traffic load to the physical network, and the VN2 is used to evaluate the performance of the BAS relative to big link throughput. Here, Iperf3 is used to generate traffic and measure the results. In addition, the soft reallocation period and hard reallocation time are set to 1 s and 5 s, respectively.

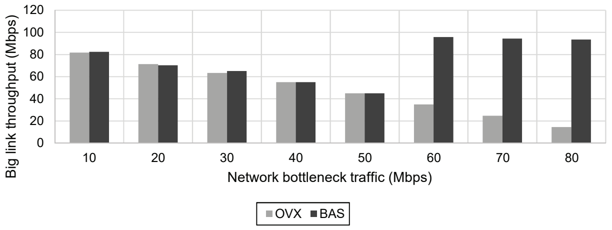

Fig. 7. Comparison of big link throughput to OVX

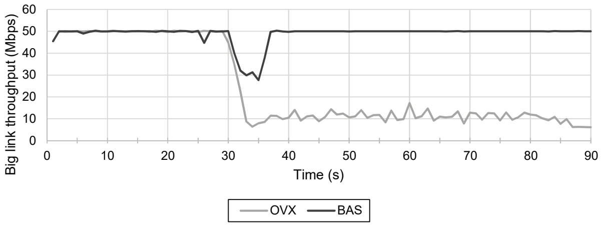

Fig. 8. Big link throughput evaluation for resource reallocation

B. Results

1) Initial resource allocation performance: Fig. 7 shows the resource allocation performance of the BAS. For the traffic load, prior to creating a big link in VN2, we generate TCP traffic between two hosts (10 to 80 Mbps) in VN1. The big link is created while VN1 traffic flows, and new traffic in VN2 (128 MB of TCP data) flows after the creation of the big link. After this, we measure the throughput of the big link during the file transfer.

The results show that the BAS successfully allocates physical resources better than OVX. Specifically, with bottleneck loads of 60 Mbps or greater, the proposed BAS shows improvements of 2.7 to 3.8 times to OVX. Between 10 and 50 Mbps, the total big link throughput is similar to existing methods. This is due to our calculated link cost. The link cost used by the routing algorithm is the reverse of the available bandwidth; thus, rather than the current link's bitrate, the results are affected by the number of hops. We plan to research a metric for this in future.

2) Reallocation performance: For resource reallocation, we demonstrate throughput enhancement when the traffic load is increased after creating a big link. For this experiment, we first create a big link in VN2 and generate 50 Mbps traffic between two hosts. Then, after 30 s, we generate 100 Mbps of UDP traffic in VN1. Here, we apply synchronous hard reallocation and loss avoidance. As shown in Fig. 8, the big

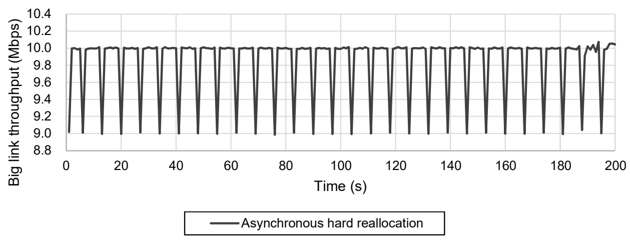

Fig. 9. Overhead measurement for asynchronous hard reallocation

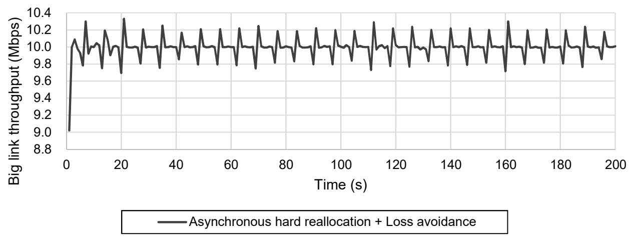

Fig. 10. Overhead measurement for asynchronous hard reallocation and loss avoidance

link throughput with the BAS increases after 35 s, which is 5 s after the traffic generation of other tenants. When the big link is initially created, physical switches {S2, S3, S4, S5, S10} are allocated to the big link. VN2's switches share the same physical switches; thus, traffic is concentrated to those physical resources, even though there are other idle resources (switches S6, S7, S8, and S9). In this situation, the BAS can allocate a new resource when there are better physical resources for the big link, unlike the existing technique. The time to recovering performance is up to the hard reallocation time. This parameter determines the interval of the network hypervisor to check the link throughput of the physical network and recalculates the optimal path before the actual policy installation occurs. The resource is reallocated for approximately 35 s since the hard reallocation time is 5 s in our case, and the throughput of BAS increases up to the previous performance of 50 Mbps.

3) Reallocation overheads: Here, we evaluate overhead for physical resource reallocation in our designs. In this experiment, we only create VN2 and generate 10 Mbps of UDP traffic between two hosts.

Fig. 9 shows the overhead in asynchronous hard reallocation scheme without applying any overhead degradation scheme. As can be seen, throughput degradation is approximately 10% (at an interval of 5 s). This happens because the policies are removed from the physical switches every 5 s, new settings are requested for the network hypervisor, and no packets are transmitted before the configuration is established. During the configuration, the packets sent from the hosts are lost.

Fig. 10 shows the results after applying the loss avoidance with asynchronous hard reallocation design. The loss of throughput is reduced significantly by the way in which the network hypervisor buffers packets that would be lost in the reallocation process and sends them to the last physical switch allocated to the big link.

Finally, we compare the BAS with both synchronous hard reallocation and loss avoidance to OVX to evaluate the

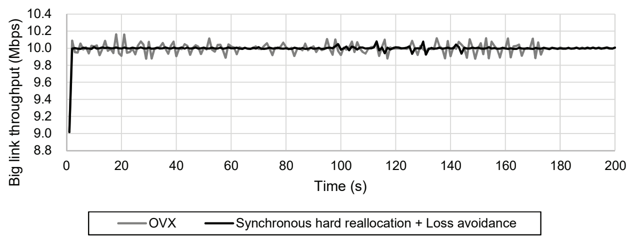

Fig. 11. Overhead measurement for synchronous hard reallocation and loss avoidance

throughput overhead for the reallocation process. The results are shown in Fig. 11. The throughput fluctuation is very small compared to the results shown in Fig. 10, and there is negligible load compared to the conventional technique which does not reallocate physical resources. This is because synchronous hard reallocation avoids unnecessary reallocation for overlapping resources by determining whether the best physical resources at that moment have changed.

VII. Related work

This work is related to SDN-based network virtualization. Compared to the previous work [6], BAS provides a big link with a higher throughput. On the other hand, BAS is also related to the network embedding algorithms in that BAS deals with creating a big link within a single physical topology. The main purpose of network embedding algorithms is to calculate capable physical resources to satisfy constraints, i.e., delay, bandwidth for the virtual network. Because the problem is NP-hard, they reduce the problem to sub-optimal. [15], [16], [17] However, most of them focus on resource calculation itself. Moreover, the performance evaluation is done with a simulation. In contrary, we practically designed BAS to cover not only allocation scheme but also actual implementation issues, such as overheads in reallocation process. Moreover, we implemented the actual prototype of BAS with the open source network hypervisor.

VIII. Conclusion

In this paper, we proposed BAS, a resource allocation scheme for a big link in virtualized SDN to enhance the throughput of big link. By regarding the network traffic and soft/hard reallocation method, BAS enhance the throughput of the big link. We implemented BAS with OpenVirteX and evaluated the performance on the emulated network. The evaluation results show that BAS successfully enhances the initial allocation and deal with traffic loads from other tenants by reallocation. Moreover, packet loss and the throughput fluctuation is decreased. Based on our work, we shall extend the framework to cover other big resources, such as big switches. Moreover, in order to cover practical use of the design, research for guaranteeing the specified amount of network bandwidth and transfer delay will be conducted.

REFERENCES

[1] M. Mahalingam, D. Dutt, K. Duda, P. Agarwal, L. Kreeger, T. Sridhar, M. Bursell, and C. Wright, "Virtual extensible local area network (vxlan): A framework for overlaying virtualized layer 2 networks over layer 3 networks," Tech. Rep., 2014.

[2] D. Farinacci, T. Li, S. Hanks, D. Meyer, and P. Traina, "Generic routing encapsulation (gre)," Tech. Rep., 2000.

[3] E. Rosen, D. Tappan, G. Fedorkow, Y. Rekhter, D. Farinacci, T. Li, and A. Conta, "Mpls label stack encoding," Tech. Rep., 2000.

[4] D. Drutskoy, E. Keller, and J. Rexford, "Scalable network virtualization in software-defined networks," *IEEE Internet Computing*, vol. 17, no. 2, pp. 20–27, 2013.

[5] R. Sherwood, G. Gibb, K.-K. Yap, G. Appenzeller, M. Casado, N. McKeown, and G. Parulkar, "Flowvisor: A network virtualization layer," *OpenFlow Switch Consortium, Tech. Rep*, vol. 1, p. 132, 2009.

[6] A. Al-Shabibi, M. De Leenheer, M. Gerola, A. Koshibe, G. Parulkar, E. Salvadori, and B. Snow, "Openvirtex: Make your virtual sdns programmable," in *Proceedings of the third workshop on Hot topics in software defined networking*. ACM, 2014, pp. 25–30.

[7] T. Koponen, K. Amidon, P. Balland, M. Casado, A. Chanda, B. Fulton, I. Ganichev, J. Gross, P. Ingram, E. J. Jackson *et al.*, "Network virtualization in multi-tenant datacenters." in *NSDI*, vol. 14, 2014, pp. 203–216.

[8] X. Jin, J. Gossels, J. Rexford, and D. Walker, "Covisor: A compositional hypervisor for software-defined networks." in *NSDI*, vol. 15, 2015, pp. 87–101.

[9] M. Team, "Mininet: An instant virtual network on your laptop (or other pc)," 2012.

[10] B.-y. Yu, G. Yang, K. Lee, and C. Yoo, "Aggflow: Scalable and efficient network address virtualization on software defined networking," in *Proceedings of the 2016 ACM Workshop on Cloud-Assisted Networking*. ACM, 2016, pp. 1–6.

[11] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.

[12] S. Tomovic and I. Radusinovic, "Fast and efficient bandwidth-delay constrained routing algorithm for sdn networks," in *NetSoft Conference and Workshops (NetSoft), 2016 IEEE*. IEEE, 2016, pp. 303–311.

[13] G. Yang, K. Lee, W. Jeong, and C. Yoo, "Flo-v: Low overhead network monitoring framework in virtualized software defined networks," in *Proceedings of the 11th International Conference on Future Internet Technologies*. ACM, 2016, pp. 90–94.

[14] P. Berde, M. Gerola, J. Hart, Y. Higuchi, M. Kobayashi, T. Koide, B. Lantz, B. O'Connor, P. Radoslavov, W. Snow *et al.*, "Onos: towards an open, distributed sdn os," in *Proceedings of the third workshop on Hot topics in software defined networking*. ACM, 2014, pp. 1–6.

[15] M. Yu, Y. Yi, J. Rexford, and M. Chiang, "Rethinking virtual network embedding: substrate support for path splitting and migration," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 17–29, 2008.

[16] W.-H. Hsu, Y.-P. Shieh, C.-H. Wang, and S.-C. Yeh, "Virtual network mapping through path splitting and migration," in *Advanced Information Networking and Applications Workshops (WAINA), 2012 26th International Conference on*. IEEE, 2012, pp. 1095–1100.

[17] M. Capelle, S. Abdellatif, M.-J. Huguet, and P. Berthou, "Online virtual links resource allocation in software-defined networks," in *IFIP Networking Conference (IFIP Networking), 2015*. IEEE, 2015, pp. 1–9.