International Journal of Future Generation Communication and Networking
Vol. 9, No. 9 (2016), pp. 67-76
http://dx.doi.org/10.14257/ijfgcn.2016.9.9.06

End-to-end Dynamic Bandwidth Allocation Basedon User in Software-Defined Networks

GengZhang [1], Dahua Zhang [1], Liang Zhou [1] and Xi Liu [2]

[1]*China Electric Power Research Institute of information and Communications Research Institute，Beijing100192，China*
[1]*Sichuan Electric Power Company of State Grid, Chengdu,china*

Abstract

With the rapid development of Internet technology, more flexible applications appeared and had a higher demand on dynamic allocation of network resources. However, allocating network resources dynamically is difficult to achieve in current TCP/IP networks due to the lack of programming interface of network devices for unified and effective control and management. In recent years, a new network architecture -- software-defined networks (SDN) emerged and it provides services providers or network managers an opportunity to realize the goals. In this paper, a user-reservation-based end-to-end dynamic bandwidth allocation procedure in SDN with OpenFlow protocol is proposed. Users can reserve bandwidth and controller can allocation bandwidth dynamically to satisfy users' demand. Experiments are designed to verify the effectiveness of the allocation procedure. Results of the experiment show that the allocation procedure works well.

Keywords: *SDN, OpenFlow, dynamic bandwidth allocation*

1. Introduction

In the past few years, internet technology evolves rapidly and more flexible applications appeared.In this case, allocating network resources such as bandwidth dynamically is becoming more and more important. For example, some users may require more bandwidth in a short period of time for some special purposes, such as setting up a temporary video conference. If they rent enough bandwidth for their special needs all the time, it would cause waste of resources and increase in cost. Therefore, dynamic bandwidth allocation is of great significance. In current TCP/IP networks, bandwidth can be dynamically assigned theoretically, however, it doesn't work effectively or efficiently [1-2]. Devices used in current networks like routers and other middle-boxes are difficult to undertake unified and effective operation due to their proprietary. Therefore, dynamic bandwidth allocation is almost impossible to achieve because of the great scale of today's networks and the diversity of network devices. Besides, although some mechanisms such as Resource Reservation Protocol (RSVP) can be used to reserve bandwidth [3], however, they are not flexible enough to satisfy the demand. So it is quite complicated or even impossible for us to allocate bandwidth dynamically for specific users in current networks. Recently, the emergence of SD provides an opportunity to solve the problem.

SDN is a novel network architecture that is dynamic, manageable and programmable. These features make it ideal for the high-bandwidth, dynamic nature of today's applications. This architecture separate the control-plane and data-plane, enabling the network control to become directly pr[1]ogrammable and the underlying infrastructure to be abstracted for applications and network ser[2]vices[4-5].

1 This work is supported by Science and Technology Project of the StateGrid Corporation of China(Research on Global Energy Internet technology system)

2 This work is supported by Science and Technology Project of the StateGrid Corporation of China(Research on Global Energy Internet technology system)

ISSN: 2233-7857 IJFGCN

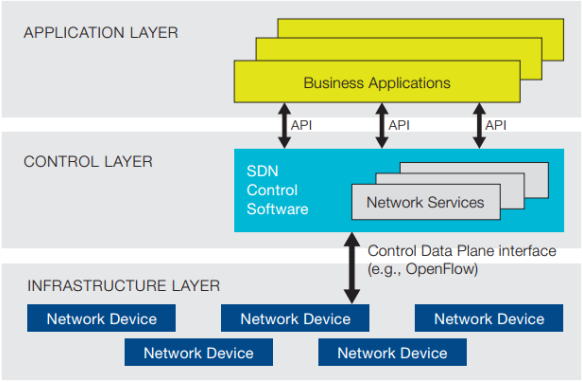

Figure 1. Software-defined Network Architecture

As illustrated in Figure 1, the architecture of SDN consists of application layer, control layer and infrastructure layer (data-plane). The physical separation of the control-plane and data-plane is the best known principle of SDN. It postulates an external control-plane entity universally called "controller" that carries out the control function extracted from traditional network devices. With the separation, network intelligence and state are logically centralized, and the underlying network infrastructure is abstracted from applications. In this case, SDN lets networks managers configure, manage, secure and optimize network resources very quickly via dynamic, automated SDN program which they can write themselves [6-8]. This is the key feature of SDN – Programmability.

In SDN, the interface between control-plane and data-plane is called "southbound interface". It is the enabler for the externalization of the control-plane and therefore key to the corresponding SDN principle [9]. Its realization is a standardized instruction set for the networking hardware. The most well-known southbound interface is the OpenFlow [10-11]. Also, SDN enables the exchange of information with applications running on top of the network. This information exchange is performed via an interface called "northbound interface". However, unlike southbound interface, standardized northbound interface does not exist.

In this paper, we propose a user-reservation-based end-to-end dynamic bandwidth allocation procedure in SDN. In the procedure, authorized users could send bandwidth allocation requests to the controller if necessary. The controller receives the requests, using an algorithm to calculate routes that can satisfy the demands of those users and then allocate the bandwidth for them. The remainder of this paper is organized as follows. Section 2 presents the allocation procedure in detail, describing the specific implementation of each step in the procedure. Section 3 evaluates and analyzes the effectiveness of the allocation procedure and Section 4 concludes the paper.

2. Dynamic Bandwidth Allocation

In this paper, we consider dynamic bandwidth allocation in a pure SDN environment, where all forwarding devices in data-plane are SDN-enabled. Besides, OpenFlow is used as the southbound interface. Moreover, we assume that the network system is a single domain SDN system, where all forwarding elements are controlled by one single controller.

2.1. Dynamic bandwidth allocation in SDN

The bandwidth allocation procedure is user-based in this paper. When a user needs certain bandwidth, it sends a bandwidth allocation request packet to controller. Request packet contains some requisite information such as the identification of the user itself, the destination it wants to reach, how much bandwidth it needs, when it needs the service and

so on. Since host connects to forwarding devices, and it could not connect to controller directly, one way for the request packet to arrive to controller is being in form of "Packet-in". Packet-in is one of the asynchronous message types in OpenFlow protocol [12]. Generally, when switch received a packet that has no matched flow-entry, it encapsulates the packet and sends it to the controller as a Packet-in message. We can construct a packet with special IP address or special MAC address that would never be actually used in the network to make sure that no matched flow-entry exists. In this case, this specific packet will be encapsulated as a Packet-in message and delivered to the controller. When receiving a packet-in message, controller check whether it is a bandwidth allocation request or not. If not, just ignore it. If it is, the controller handles it, using topology and bandwidth information to calculate a route for bandwidth allocation. Finally, controller replies the allocation result to the host. The procedure can be presented in Figure 2 below.

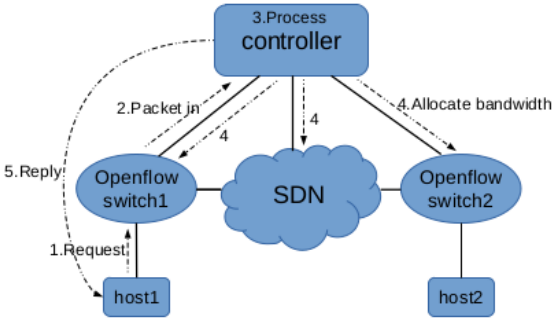

Figure 2. Bandwidth Allocation Procedure in SDN

2.2. Implementation of Dynamic Bandwidth Allocation

The implementation of dynamic bandwidth allocation can be divided into two parts, client-side module and controller-side module. Client-side module runs on host, sending allocation request to ask for bandwidth. Controller-side module is a little more complicated. It runs on the controller as a special application of it. It contains several functionality including requests parsing, residual bandwidth and routecalculation, bandwidth allocation and recovery.

2.2.1. Client-side module: User runs client-side module to send a bandwidth allocation request to controller when needed. With the above analysis, constructing a packet with a special IP or MAC address for asking bandwidth is required. Considering that when constructing an internet layer datagram or a transport layer segment, the host firstly sends an ARP (Address Resolution Protocol) request message to get the MAC address of the special IP address. However, it would never get the reply normally. Then the request packet would not be sent unless a special MAC is added to the ARP table in the host manually, which is not practical. Therefore, we should construct a packet with specific MAC address. Raw socket is able to realize such functionality. The structure of the request packet is as followed in Figure 3.

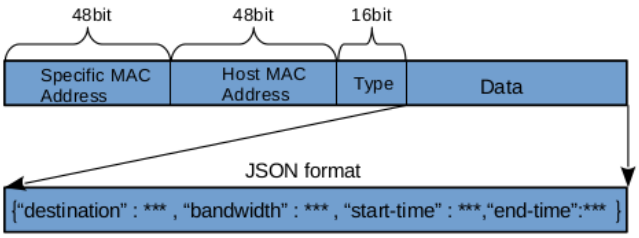

Figure 3. Request Packet Structure

The structure of the request packet is similar to Ethernet II Frame. In the first 48 bits (or 6 bytes), it is a peculiar MAC address. In fact, we can use some official reserved address [13]. The next 48 bits is the MAC address of the host. Then, the following 16 bits is Ether Type, which identifies as upper layer protocol encapsulating of the frame data, 0x8000 for IPv4 datagram, and 0x0806 indicates ARP frame for example. And here, we use a special type 0x0000 which is not actually used in internet to provide convenience for controller to verify the request packet. Data in the request packet is in JSON (JavaScript Object Notation) format. It contains several attributevalue pairs. The value of attribute "destination" means the destination that the host wants to transmit data to, then "bandwidth" corresponds to how much bandwidth the user needs, "start-time" and "end-time" present the time that the user needs to use the bandwidth resources. After end-time, controller would recycle bandwidth and wait for assigning to another user's request. Certainly, if need increases, we can add other attribute-value pairs to provide more information like identification if the network system requires further authentication. Furthermore, encryption can be undertaken for the data for privacy protection.

2.2.2. Controller-side module: Controller-side module works as follows in Figure 4.

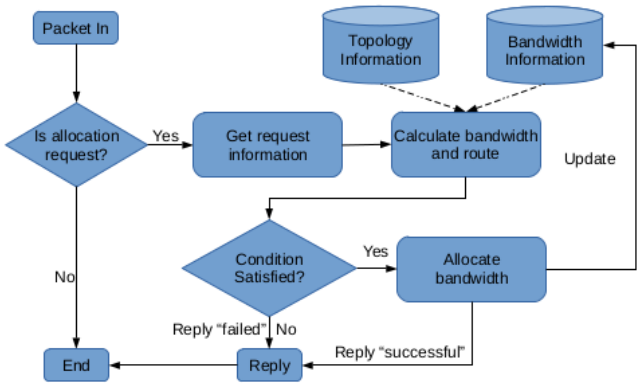

Figure 4. Request Allocation Procedure in Controller

When controller receives a packet-in message, it parses the packet and inspects the MAC address and Ether Type of the packet. If the packet is an allocation request, it gets request information, including destination, bandwidth, start time and end time of service and sends it to request handler waiting for processing. The request handler uses information of request, topology and bandwidth to process the allocation request, determining whether the request can be accepted or not due to the limitation of bandwidth resource. If the request can be satisfied and accepted, the module then controls switches via OpenFlow to allocate bandwidth and sends successful-reply to the user and the result is sent to the bandwidth information database for updating in the meantime. Otherwise, controller module replies "failed" to the host.

A. Acquisition of Topology Information

Since bandwidth allocation is end-to-end in this paper, we ought to calculate a route from the source to the destination. For this reason, topology information must be obtained. At present, all implementations of controller contain topology discovery functionality. However, the topology which their build-in function gets is partial since only switches' topology is gained, but no hosts. Without the connection information of hosts, route is unable to be got. Inspired by build-in topology discovery function in controller, we use Link Layer Discovery Protocol(LLDP，IEEE 802.1AB) to discover hosts and obtain global topology. In order to achieve this, we assume that hosts could transmit LLDP frame. Moreover, the LLDP frame transmitted by host takes its own IP address as a TLV. After the frame is sent, it goes to the switch that the host connects to. Then it would be delivered to controller as packet-in message. Note that the packet-in message contains information of the switch, so we can find out the switch and the port that the specific host connects to. In this way, with switches' topology provided by controller itself, global topology information can be gained. Besides, a periodical inspection is taken to check whether the connection between host and switch is alive or not. For example, if an LLDP frame is sent in every 30 seconds by hosts, then the inspection period can be set to 32 seconds. If controller has not received a LLDP frame from a host that was once discovered in 32 seconds, then we draw conclusion that the link between host and switch is down and host is disconnected.

B. Acquisition of Bandwidth Information

Only when there is enough bandwidth for reserved can the allocation request be accepted and the allocation procedure be executed, so bandwidth information should be acquired before allocation request is processed. We can easily obtain link capacity information by calling API provided by controller since this functionality is supplied by OpenFlow. Since bandwidth allocation is reservation-based, we need to calculate bandwidth that has been assigned or reserved in the specific time that the user asked for bandwidth allocation. Then we can get the amount of the residual bandwidth and send this information to route computation module described below to obtain a route for the request.

In order to achieve this, results of requests that have been processed should be stored. If a request can be accepted, the following information is store: start-time, end-time, amount of bandwidth and the route. For convenience, parameters are defined as follow to describe the procedure of calculating residual bandwidth:

Definition:

n	Number of requests that have been accepted
E	Set of links between switches
$i \leftrightarrow j$	Link between switch$i$ and switch j
$RH=\{rh_k\}$	Set of previous requests that have been accepted
st_k	Start-time of request rh_k
et_k	End-time of request rh_k
b_k	Bandwidth of request rh_iasked for
$route_k$	Route for allocating bandwidth for request rh_k.$route_k = (i_1^k \leftrightarrow j_1^k, i_2^k \leftrightarrow j_2^k \ldots i_l^k \leftrightarrow j_l^k)$, *means* $route_k$ *consists of link* $i_1^k \leftrightarrow j_1^k, i_2^k \leftrightarrow j_2^k \ldots i_l^k \leftrightarrow j_l^k$.
st_0	Start-time of current request
et_0	End-time of current request
b_0	Bandwidth of current request asked for
$R=\{r_{ij}\}_{n\times n}$	Residual bandwidth matrix. r_{ij} represents the residual bandwidth of link $i \leftrightarrow j$. Initially, r_{ij} is the link capacity of $i \leftrightarrow j$. And$r_{ij} = 0$ if$i \leftrightarrow j$not in E

Procedure of Calculating Residual Bandwidth

```
for rh_k in RH:
    if st_k ⩽ st_0 < et_k:
        for i↔j in route_k:
            r_ij = r_ij - b_k
        endfor
    endif
endfor
```

After the procedure, R is the residual bandwidth matrix that can be used for route computation.

C. Route Calculation

Since users asked for end-to-end bandwidth, a route should be obtained for bandwidth allocation. Note that centralized controlled is the most important superiority of SDN and this special characteristic makes it easy for network optimization, we can use this feature to find a "best" route for bandwidth allocation.In this session, a route calculation algorithm which can find a shortest route that has enough bandwidth for allocation is proposed.

In this paper, network system can be modeled as an undirected graph $G=<N, E>$. And here comes the definition of the model.

Definition:

N	Set of switches
n	Number of switches
E	Set of links between switches
$i \leftrightarrow j$	Link between switch$i$ and switch j
$R=\{r_{ij}\}_{n\times n}$	Residual bandwidth matrix. r_{ij} represents the residual bandwidth of the direct connection between switch$i$ and switch j.$r_{ij} = 0$ if$i \leftrightarrow j$ not in E
b	Bandwidth that the host request
$A=\{a_{ij}\}_{n\times n}$	Adjacency matrix. a_{ij} is the distance between switch i and switch j and $a_{ij} = infinity$ if$i \leftrightarrow j$ not in E
$RA=\{ra_{ij}\}_{n\times n}$	Improved adjacency matrix. $ra_{ij} = a_{ij}$ if$r_{ij} - b \geqslant 0$;$ra_{ij} = infinity$ if$r_{ij} - b < 0$

Route Computation Algorithm for Bandwidth Allocation

```
1. get R, r, A
2. compute RA:
    for i = 1:n
        for j = 1:n
            if r_ij - b ⩾0:
                ra_ij= a_ij
            else :
                ra_ij= Infinity
        endfor
    endfor
3. call Dijkstra(RA, source, destination)
```

Dijkstra(RA, source, destination) is Dijkstra algorithm[14] implementation. It returns the shortest path between source and destination in graph RA if paths exist, otherwise it returns NULL.

The algorithm for bandwidth allocation returns the shortest path that has adequate bandwidth for allocation from the source to the destination, and returns NULL if the

residual bandwidth is inadequate for the request.

After the allocation request is processed, we store the result in the bandwidth information database. When dealing with the next request, previous allocation information is taken out to calculate residual bandwidth.

D. Bandwidth Allocation

OpenFlow-enable switch has the feature to assign bandwidth by setting Qos and Queue for a specific flow. Therefore, end-to-end bandwidth allocation can be achieved by setting the flow to transmit packets from a host to another with Qos and Queue.

Since bandwidth allocation is reservation-based, bandwidth should be allocation at the start-time the request asked and would be recycled after the end-time. In this case, controller has to maintain two timers, one for start-time and the other for end-time. When start-time of a request comes, timer for start-time sends out notification and controller calls the allocation function to accomplish bandwidth allocation.Besides, when end-time comes, timer for end-time would inform controller to delete Qos and Queue, cancelling the bandwidth allocation.

We suppose that the structure of the path returned by route calculation algorithm is showed as follows:

$PATH = [(switch_{a1}, port_{b1}), (switch_{a2}, port_{b2}) \ldots (switch_{ak}, port_{bk})]$

The Bandwidth Allocation Process
for (switch, port) in PATH:
1. Add Qos for port of switch
2. Add a Queue for Qos, set min_rate and max_rate of the Queue equal to b (b is the value of requested bandwidth)
3. Add a flowentry, specify in_port, eth_src, eth_dst in match field and OUTPUT, SET QUEUE in action field
endfor

3. Experimental Result

Experiment has been designed to check the effectiveness of the bandwidth allocation procedure proposed in this paper.

Experiment uses topology shown in Figure 5: one controller, two OpenFlow-enabled switches and two hosts.

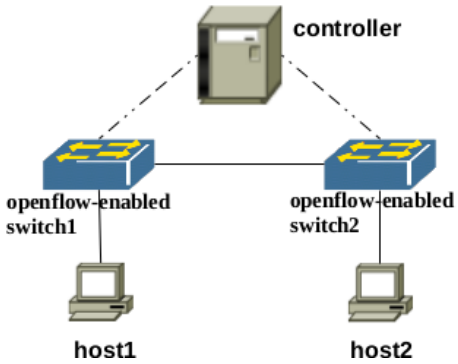

Figure 5. Topology of Experiment

There are several popular open source SDN controllers such as NOX [15], Floodlight[16], Ryu[17] and Opendaylight[18]. In consideration of continuous availability, functional completeness and simplicity of development, we choose Ryu as our experimental SDN controller in this paper. Openvswitch[19] (ovs) is used as OpenFlow-enabled switch. The IP address of host1 is 192.168.1.1 and host2 is 192.168.1.3.

To test the allocation result, we use iperf[20] to measure the bandwidth between host1 and host2.

In the testing process, we run client-side module in host1, asking for 10Mbps bandwidth to host2. After that, we run *iperf* program to test the bandwidth between host1 and host2. Then we change the value of bandwidth, asking for 100Mbps. The results are as follows:

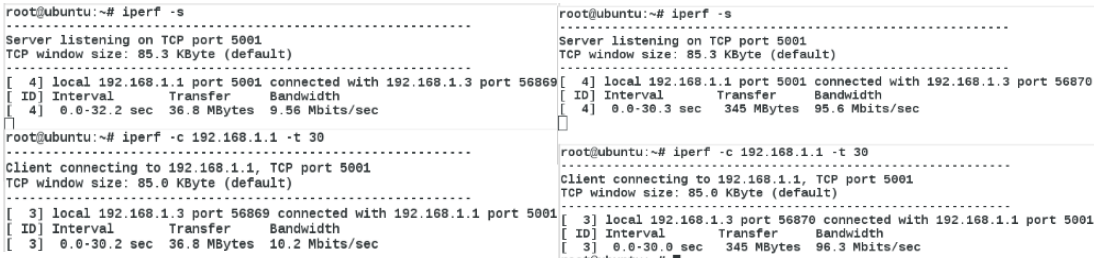

Figure 6. Result of Bandwidth Allocation

The left picture in Figure 6 is the result of asking for 10Mbps and the other one is the result of asking for 100Mbps. We can find out that the procedure works well.

4. Conclusion

In this paper, we have shown how to achieve end-to-end dynamic bandwidth allocation based on user in software-defined networks with OpenFlow protocol. After having described the implementation steps and scheduling model in detail, we have designed an experiment to verify effectiveness of the procedure. Our experimental results have shown that the allocation procedure is effective.

References

[1] Z. Chunyue, H. Jiuchuan and Z. Hongke, "A dynamic measurement-based bandwidth allocation scheme with QoS guarantee", Signal Processing, 2004, Proceedings. ICSP '04. 2004 7th International Conference on IEEE, vol. 3, **(2004)**, pp. 2640-2642.
[2] L. Enhui, "Method, apparatus, edge router and system for providing QoS guarantee", US, US7903553 B2, **(2011)**.
[3] P. Thulasiraman and Y. Sagir, "Dynamic bandwidth provisioning using Markov chain based RSVP for unmanned ground networks", Cognitive Methods in Situation Awareness and Decision Support (CogSIMA), 2014 IEEE International Inter-Disciplinary Conference on IEEE, **(2014)**, pp. 130-136.
[4] Open Networking Foundation, Software-Defined Networking (SDN) Definition, https://www.opennetworking.org/sdn-resources/sdn-definition
[5] M. Jarsche, T. Zinner, T. Hossfeld, P. T. Gia and W. Kellerer, "Interfaces, attributes, and use cases: A compass for SDN Communications Magazine", IEEE, vol. 52, no. 6, **(2014)**, pp. 210-217.
[6] I. F. Akyildiz, A. Lee, W. Pu, L. Min and C. Wu, "A roadmap for traffic engineering in SDN-OpenFlow Networks", Computer Networks, vol. 71, no. 3, **(2014)**, pp. 1–30.
[7] S. Tomovic, N. Prasad and I. Radusinovic, "SDN control framework for QoS provisioning", Telecommunications Forum Telfor (TELFOR), **(2014)**.
[8] J. Michael, H. Tobias, D. Franco, B. Raffaele, B. Roberto and C. Alessro, "SDN-Enabled Energy-Efficient Network Management", Green Communications: Principles, Concepts and Practice. John Wiley & Sons, Ltd, **(2015)**, pp. 323-338.
[9] T. Nadeau and K. Gray, "SDN: Software Defined Networks", O'Reilly Media, **(2013)**.
[10] N. Mckeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson and J. Rexford, "OpenFlow: Enabling Innovation in Campus Networks", AcmSigcomm Computer Communication Review, vol. 38, no. 2, **(2008)**, pp. 69-74.
[11] Open Networking Foundation, OpenFlow Switch Specification https://www.opennetworking.org/sdn-resources/onf-specifications/OpenFlow
[12] Open Networking Foundation, OpenFlow Switch Specification Version 1.3.3 (Protocol version 0x04) October 18, 2013 page: 29
[13] Group MAC address assignments for standards use, "http://www.ieee802.org/1/files/public/docs2007/admin-jeffree-standard-group-mac-address-assignmets-0307.pdf

[14] E.W.Dijkstra. A note on two problems in connexion with graphs.Numerischemathematik 1.1 (1959): 269-271.
[15] NOX“http://www.noxrepo.org/nox/about-nox/”
[16] Floodlight “http://www.projectfloodlight.org/floodlight/”
[17] Ryu “http://osrg.github.io/ryu/”
[18] Opendaylight“http://www.opendaylight.org/”
[19] Openvswitch “http://openvswitch.org/”
[20] Iperf “https://iperf.fr/”

Authors

GengZhang, he is a Senior Engineer at China Electric Power Research Institute. He lives in No.15, Xiaoying East Road, Qinghe, Haidian District, Beijing, China, 100192. His Research Interests: SDN, Energy Internet, Video Communication.

Dahua Zhang, she is a Senior Engineer at the China Electric Power Research Institute. She lives in No.15, Xiaoying East Road, Qinghe, HaidianDistrict, Beijing, China, 100192. Her Research Interests: Information technology of power, Informatization planning.