Big Flow Collision Avoidance Strategy for Load Balancing in SDN

Bin LUO[1,a,*],Jing SHEN[1,b], Li-Qian SUN[2,c]

[1]State Grid Henan electric power company information and telecommunication branch, [2]Beijing University of Posts and Telecommunications

[a]38362751@qq.com, [b]Shenjingpaper@163.com[c]liuyqpaper@163.com

Keywords: Software defined network, Huge flow, Equal-Cost Multi-Path (ECMP), A big flow collision avoidance strategy for load balancing (BFCAS), Fat-Tree.

Abstract: Software definition network (SDN) as a new model of the future network architecture system, which gains the academic and industry attention. Large traffic often leads to congestion and transmission of network links. Equal-Cost Multi-Path (ECMP) routing can play a significant role in balancing the load, but it cannot solve two or more large and long cycle of flow through the pseudo-random algorithm to select the same link forwarding collision. Then, we propose a big flow collision avoidance strategy for load balancing (BFCAS) to handle the collision of the huge flow, the default basic ECMP to handle the tiny flow. Finally, the SDN of the Fat-Tree topology is deployed, and the effect of BFCAS collision avoidance and the scalability are analyzed experimentally.

Introduction

Software defined network is a new network model[1], which separates the control layer and the forwarding layer in the network. SoftRouter[2] proposes a separate strategy to divide data pack forwarding components from routers. These forwarding components have standard interfaces and can be controlled by centralized control layer servers. The idea that the control layer is reconstructed as a publishing layer and a decision layer [3] has been raised. In other words, the above methods are proposed from two main components of SDN, SDN controller and SDN forwarding elements. SDN controller is a logically centralized function[4]. A network usually is controlled by one or more controllers. Controller decides every forwarding path of each flow. SDN forwarding elements constitute the data layer of the network. However, the logic of the forwarding packet is determined by the SDN controller[5] and loaded into the flow table of the SDN forwarding elements.This paper mainly considers the multipath routing of traffic from switch load balancing. Multi-path routing has been extensively studied for a long time. For the local area network, multi-path routing has a lot of advantages [6] which has also been experimentally proved.

Equal-Cost Multi-Path base on hash[7] is a load balancing scheme that uses flow Hash technology to allocate traffic to an available path. For a given sub-net, we can deploy a switch with multiple possible forwarding paths that support ECMP. In order to solve the limitation of ECMP, this paper presents the BFCAS (Big flow collision avoidance strategy for load balancing) in the SDN environment to equalize the flow on the available paths[8]. BFCAS algorithm is to obtain a backup path group in advance, the paths in the path group do not intersect each other, the purpose of which is to route the flow[9], so as to avoid the collision between the large flow; algorithm according to the order of nodes and nodes between the path group, through the alternate path group, the network load[10] can be further balanced.

The rest of paper is organized as follows: related work is elaborated in the second part; in the third part, the theory of BFCAS is introduced, and the model and formula of BFCAS are introduced; in the fourth part, experiments are used to evaluate algorithm of BFCAS, and the experimental topology the Fat-Tree topology is given in this part; the last part is a summary of the full text, and prospects for future work.

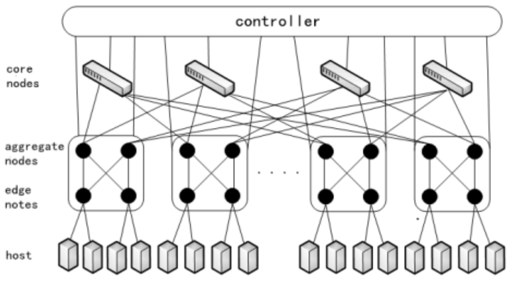
Related Works

In order to address the shortcomings of ECMP, the researchers proposed two load balancing solutions: Hedera[9] and Mahout[10] . Table 1 compares these two scenarios.

Table 1 Comparisons of Hash - based ECMP Traffic Forwarding Scheme

Hash-based ECMP traffic forwarding scheme			
Proposed method	Huge flow investigation	Process overhead	Bandwidth overhead
Hedera	edge switches	between controller and switches	the switches of high demand
Mahout	endpoint hosts	between switches and hosts	the hosts of high demand

In the Hedera structure, there is a control loop, which consists of two basic steps: (1) when the edge switch detects a large flow ("huge" flow), as described in Figure 1, this path is used until traffic increases and reaches a specific threshold (such as 100Mbps in a Hedera deployment); (2) Hedera dynamically computes the appropriate path for the flow and loads it into the switch. Hedera uses periodic polling scheduling on the edge switch to collect traffic statistics and detect traffic every five minutes.



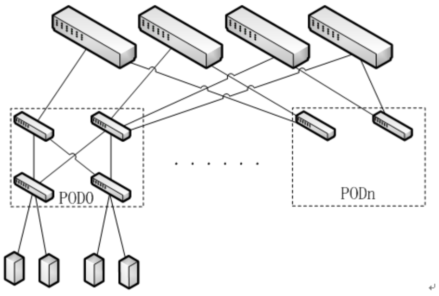


Figure1 the structure of Hedera Figure2 the topology of Fat-TreeSDN

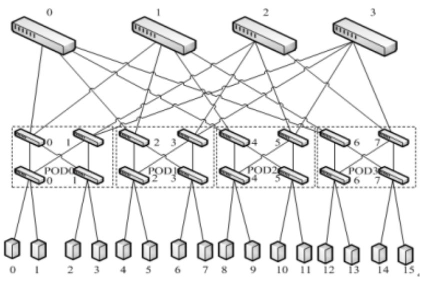


Figure3 the topology of Fat-Tree in SDN N=4

Mahout manages the flow by investigating a large flow[9]. Hedera uses periodic polling to detect the "huge" flow, which collects statistics for each flow from each edge switch. Mahout's key idea is to monitor and detect the "huge" flow on the terminal host through a shim layer in the operating system rather than directly detecting the switches in the network.

In this paper, the BFCAS algorithm is based on Hash to avoid the collision, in which the control center is not necessary, and it is conducive to SDN controller and OpenFlow switch expansion. Because the topology is Fat-Tree structure, the preferred sequence of the standby path groups is obtained in advance to facilitate distributed deployment. Special processing is performed for large flows, thereby reducing the probability of congestion in the network.

Proposed Algorithm

In this section we describe the BFCAS model construction and formula implementation, and proposes the method in the SDN environment to avoid large flow collision. As shown in Figure 2, according to the difference between source and destination addresses, the flow can be classified into three categories: connect interaction between the edge nodes in the same host; connect interaction between the edge nodes belong to the same slot (POD) but in different hosts; connect interaction between edge nodes not in the same slot (POD). So the following paper mainly describes the flow of the third category. In this paper, the pair of node (s, d) is used to represent the flow and the link, without causing ambiguity, we will not finger out s and d belongs to which node. In figure 2, the nodes from bottom to top are hosts/edge nodes/aggregate nodes/core nodes.

Models

Definition 1: Use a function w and a path system P to represent the forgetful routing policy, where function w is the weight of any path in system P. The characteristics of w are satisfied for the source-destination node pair (s, d). Definition 1 describes a random variable Y, where Y represents the probabilistic event of the node pair (s, d) in either of the alternate path groups, where the alternate path group is pre-computed.

As shown in Figure 3, the interaction between hosts under different PODs can be selected for four different paths. For example, the alternate path group (1, 10) has {(1,0,0,0,4,5,10), (1 , 0,0,1,4,5,10), (1,0,1,2,5,5,10), (1,0,1,3,5,5,10)}. If the path is chosen to be evenly distributed, the probability of collisions due to the selection of the same core link for more than two flows from the same POD can be calculated. It can be seen that if the large flow is issued from the same POD node, they have a very high collision probability at the convergence layer link.

Definition 2: In the slot routing mode of the fat-tree topology of the software-defined network, the path system of the node pair (s, d), where any path is randomly selected to route the node pair (s, d) The Thus defining a random selection function.

Definition 2 determines the probability distribution associated with the node pair, which is the construction node selects the probability distribution for the associated path. Because of the symmetric nature of Fat-Tree, we consider that the probability of a large-flow collision can be considered only from the source node, and this leads to another definition as follows.

Definition 3: In the case of slot-to-slot routing of the fat-tree topology of the software-defined network, the core-level node is the necessary node for all paths, and the set of such nodes is set to C. Define a random function, where is the set of all host nodes, N for the number of slots.

Definition 3 gives the probability distribution of a source node matching the corresponding path to the core node after a core node has been determined. From the definition 3, the same slot (POD) issued by the flow of nodes in the link along the core of the possibility of collision can be investigated.

Huge flow detection

Because the system proposed in this paper mainly avoids the collision of huge flows, there is a need for a mechanism to distinguish huge flows from all traffic.The literature [9] has proved that the detection of the terminal host is currently the most effective way to detect huge flows. So this system will use this method to detect and mark the huge flow. [9] uses the space reserved in the DSCP (Differentiated Service Code Point) AB to mark the huge flow as long as the data flow with the DSCP domain AB is the huge stream.

When the huge flow is detected, the proposed system will use the large flow collision avoidance algorithm to deal with the corresponding huge flow. As for the tiny flow, because of their large number and delayed allergy, it is unrealistic to calculate the best path for each tiny flow. Therefore, in order to make the system have a good scalability, this article uses the default mode (for example, using a hash-based multi-path routing ECMP) to handle unlabeled data flows, that is, tiny flows.

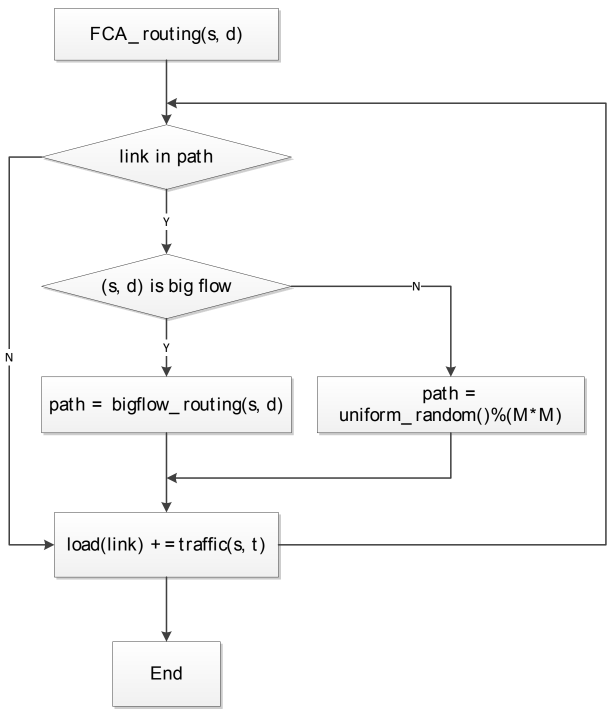
large flow collision avoidance routing

The random multi-path (RMP) routing algorithm based on the ECMP algorithm has made some improvements. RMP algorithm does not use ECMP pseudo-random characteristics. The characteristics of the same flow will be routed to the same path, which do not care about the details of the flow feature domain, while better reflect the algorithm's equalization. Randomness is an important feature of the RMP model, and the benefits are that the multi-pathing characteristics of the network are fully applied and the disadvantages are blindly random to control the collision of large flows.

From the definition 1, the probability distribution of RMP is uniformly distributed, where it is similar to the probability distribution of the optimization of single path routing, that is, the probability of selecting only one specific path.The topology discussed in this section is shown in Figure 3, limiting the source node in the same POD is the focus of our discussion. Use to denote a group of paths starting within the same POD, and each path will have a core node corresponding. At the same time, the probability distribution of the source node selection path is defined according to the definition1, which is the simplification of its representation.

We know four source nodes in a slot select four paths, because different paths are selected for different sources, and the single path routing algorithm is optimized so that the probability of collision avoidance can reach 100%.BFCAS algorithm, a source node of the main path is to optimize the path of single-path routing, the remaining path belongs to the path group, can be used as a backup. The main path and the alternate path are selected by the definiaton1 and 2. The difference between the main path and the alternate path is reflected in the probability that the selected path is higher than the alternate path.

For the four nodes to select the arrangement of four paths, we can see that the four nodes have different paths. After the probability distribution is given, the probability values for each case can be calculated, since the 24 cases cannot occur at the same time, so the probability that the path does not collide can be obtained by summation. Figure 4 is the algorithm flowchart of BFCAS routing algorithm proposed in this paper. For non-large-flow routing, it uses a uniform and random route; for large-flow routing it uses different probability routing.


FCA_routing(s, d)
link in path
Y
N
(s, d) is big flow
N
Y
path = bigflow_routing(s, d)
path = uniform_random()%(M*M)
load(link) += traffic(s, t)
End
Figure 4 Algorithm flowchart of BFCAS

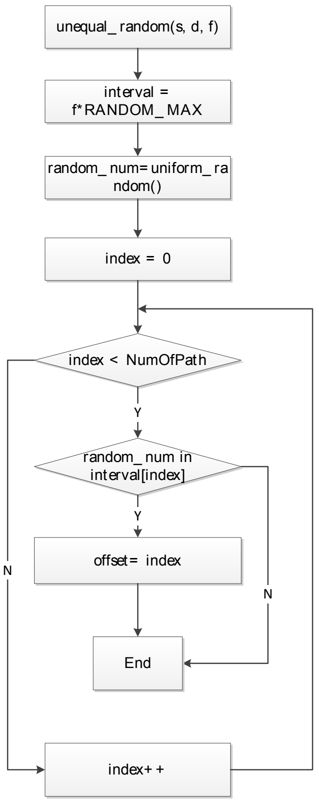


Figure 5 Algorithm flowchart of LQPGA

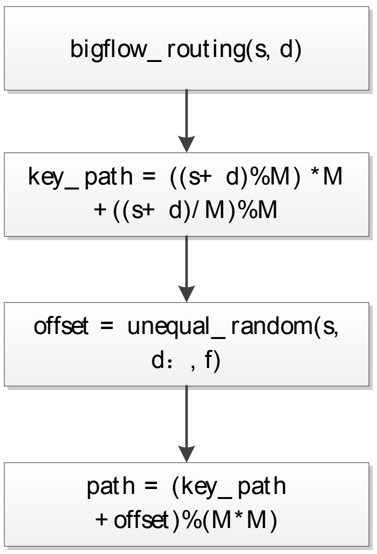

bigflow_routing(s, d)
key_path = ((s+ d)%M) *M +((s+ d)/M)%M
offset = unequal_random(s, d: , f)
path = (key_path +offset)%(M*M)

Figure 6 Algorithm flowchart of LFRA

Figure 5 is the flowchart of the above-mentioned inequality probability generation algorithm. It is based on the specified probability distribution, the current path of the random offset (compared to the main path), for the choice of large flow routing to avoid the collision of different paths. Algorithm 3 is the above-mentioned large flow routing algorithm as shown in figure 6, because the main path and alternate path is selected probability difference, the main path will be preferred. M = N / 2, where N is the number of switch ports, used to mark the path with the serial number.

Experimental Simulation

In this paper, the topology of the BFCAS algorithm is shown in Fig. 3. We used the experimental results to prove the effectiveness of the proposed algorithm in this paper. In the experiment, we investigated the loading of a single traffic request as input, comparing the two routing modes: random multipath routing (RMP) and a big flow collision avoidance strategy for load balancing (BFCAS) on a balanced core link.

The load on the core link is shown in Figure 7, and if a routing algorithm does not have a load that balances a core link, it is represented by a point with a load value of zero. Because the RMP algorithm blindly randomly selects the path, it limits its ability to reduce the maximum link load. The BFCAS algorithm calls the large flow routing algorithm, select the optimization of single path routing algorithm for the probability of the path is great, so as to avoid the collision of large flow.

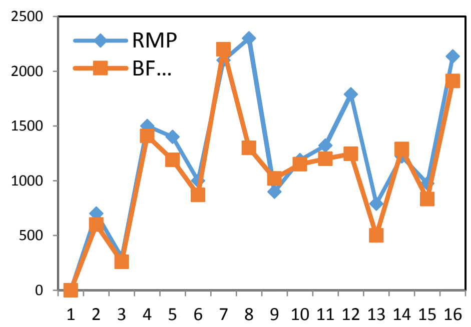


Figure 7 Single traffic request under core link load

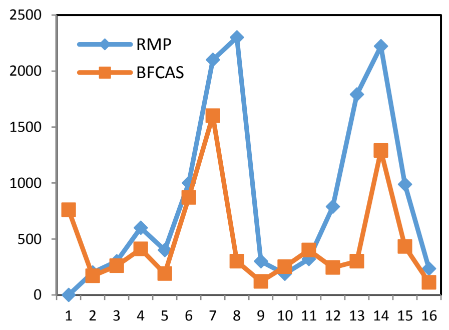


Figure 8 the distribution effect of each algorithm after topology increased

The number of host nodes in the topology is expanded by 8 times, the single traffic request is tested as the input effect, and Figure 8 is a partial view of the data. As can be seen from the figure 8, when the topology becomes larger, BFCAS still has a better effect of decentralized flow, which also proves that BFCAS has good expansibility. In figure 8 the x-axis is core link, the y-axis is link load, the unit of y-axis is 1 byte.

Conclusions and Future Work

This paper mainly deals with the problem of large flow collision in SDN environment. Only the huge flow is processed in the proposed system, and the basic ECMP is used as the default for the tiny flow. For the collision of elephant flows, we propose a BFCAS algorithm to avoid collisions. And compared with the RMP model algorithm, it can be concluded that BFCAS has better effect in the large flow collision and has stronger expansion than RMP. Since the SDN deployment in the industry is still in its start stage, researches on huge flows is also a specific application in intra-domain and data center networks. The future can be achieved between domains and testing. Our proposed algorithm enables load balancing to avoid large traffic collision.

Reference

[1] Blenk A, Basta A, Reisslein M, et al. Survey on Network Virtualization Hypervisors for Software Defined Networking[J]. IEEE Communications Surveys & Tutorials, 2015, 18(1):655-685.

[2] Yazici V, Sunay M O, Ercan A O. Controlling a Software-Defined Network via Distributed Controllers[J]. Eprint Arxiv, 2014.

[3] Hu F, Hao Q, Bao K. A Survey on Software-Defined Network and OpenFlow: From Concept to Implementation[J]. IEEE Communications Surveys & Tutorials, 2014, 16(4):2181-2206.

[4] Gringeri S, Bitar N, Xia T J. Extending software defined network principles to include optical transport[J]. IEEE Communications Magazine, 2013, 51(3):32-40.

[5] Huang D Y, Yocum K, Snoeren A C. High-fidelity switch models for software-defined network emulation[C]// ACM SIGCOMM Workshop on Hot Topics in Software Defined NETWORKING. ACM, 2013:43-48..

[6] Voellmy A, Wang J. Scalable software defined network controllers[C]// ACM SIGCOMM 2012 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication. ACM, 2012:289-290.

[7] Hopps C. Analysis of an Equal-Cost Multi-Path Algorithm[M]. RFC Editor, 2000.

[8] Wood T, Ramakrishnan K K, Hwang J, et al. Toward a software-based network: integrating software defined networking and network function virtualization[J]. IEEE Network, 2015, 29(3):36-41.

[9] Jeong K, Kim J, Kim Y T. QoS-aware Network Operating System for software defined networking with Generalized OpenFlows[C]// Network Operations and Management Symposium. IEEE, 2012:1167-1174.

[10] Curtis A R, Kim W, Yalagandula P. Mahout: Low-overhead datacenter traffic management using end-host-based elephant detection[C]// INFOCOM, 2011 Proceedings IEEE. IEEE Xplore, 2011:1629-1637.