Flow redirection based MPLS label algorithm in SDN

Lisheng Wang*, Peng Yi†, Yiming Jiang†, Yunjie Gu† and Yuehang Ding*

†National Digital Switching System Engineering & Technological Research Center, Zhengzhou 450002, China
*Corresponding author’s e-mail: 344800106@qq.com

Abstract. Aiming at the problem of congestion of flow table caused by limited performance of data plane TCAM in Software Defined Networking (SDN), an MPLS labeling algorithm based on flow redirection is proposed. The algorithm utilizes the source node switch flow redirection and the path switch to decentralize the MPLS label, occupies a part of the data link bandwidth to alleviate the problem that the TCAM update flow table rate is too slow, and can increase the system switch load. Compared to the existing method, the link load is reduced by up to 60% and the flow entry insertion delay of nearly 90% is increased, and the system capacity is increased by up to 200%.

1. Introduction

As one of the new network architectures that are most likely to replace traditional networks in the future, software-defined networks have been well studied in recent years. The software-defined network adopts a mechanism that separates the data plane from the control plane. Compared with the traditional network architecture, it has the advantages of centralized control logic, open programmable, and fine-grained flow management.

The control plane of the software-defined network grasps the global topology of the data plane through the OpenFlow [1] protocol. The switch of the data plane encapsulates the header packet of the newly arrived stream into a Packet-In message and transmits it to the control through the OpenFlow Agent (OFA). The control plane calculates the route and then installs the corresponding forwarding rule on the switch along the way through the Packet-Out message. The flow table that is released is installed in the TCAM (Ternary Content Addressable Memory) [2].

In the SDN network, the TCAM module is responsible for the update, storage and lookup of flow entries. However, because TCAM is not designed for OpenFlow switches, it has the disadvantages of high cost, high power consumption, small capacity and slow rate of flow table update operation. This results in a performance bottleneck in the downlink of the packet. Based on the Pica8 switch [3], the TCAM flowmeter has a mounting rate of only 200/sec. Compared to the early control plane, a single NOX [4] controller achieves a processing speed of 30,000/sec, using parallel Maestro control. The device [5] even reaches the request processing speed of 600 000 / sec. The 200/sec stream installation rate can't easily keep up with the controller processing rate, and the downlink control link is congested. The stream request cannot be transmitted in time, which seriously affects the communication quality.

In the past, people's research on TCAM mostly focused on the storage capacity of flow tables [6-8], but ignored the problem of flow rate update rate. As a result, when the controller sends a large number of flow tables to the switch in a short time, the storage space still exists in the switch TCAM. However, the flow table update rate is too slow, so that the flow control table cannot be quickly installed, and the downlink control link is congested. This causes a waste of TCAM resources and an increase in user
service delays. The literature [9,10] discussed this problem for the first time, and proposed the performance bottleneck of software-defined network control links under existing network hardware conditions. In [10], in order to solve this problem and propose to redirect the flow request to the virtual switch, the virtual switch has strong CPU processing capability for packet-in message uploading and flow table entry installation. However, the virtual switch has higher deployment cost which are not suitable for large-scale network deployment. In [11], an improved random routing-based routing (RRD) algorithm is proposed for the problem. The optimal path is selected from all possible routing paths by adding constraints. However, the problem is The NP is difficult to solve, only the approximate solution is obtained, and the calculation process is more complicated.

Aiming at the bottleneck problem of flow table update, this paper proposes a new flow redirection-based MPLS based on the method of packing Multi-Protocol Label Switching (MPLS) tags and Segment Routing (SR) proposed in [12]. The algorithm sets the trigger threshold. When the number of flow tables to be installed in the path exceeds the threshold, several suitable switches on the path are selected. The controller only delivers the MPLS flow table to these switch, and the subsequent switch only needs to select the corresponding port to forward according to the labeled label. The innovations of this paper are as follows:

- The threshold triggering mechanism is used. When the number of flow tables to be installed in the system exceeds the threshold, the subsequent flow table is sent to the MPLS mode to reduce the update pressure of the flow table.
- Only use the MPLS-based flow table for some flows, which will not excessively increase the packet size and will not generate large transmission pressure on the data link.
- The downlink control link load is the indicator to select the switch that delivers the MPLS flow table during the route.

2. Problem analysis and modeling

2.1. Problem analysis

For the congestion of the downlink control link, the flow entries that are congested can be classified into two types. The first type is the flow entry that is sent to the source node. Since the data is initially stored in the cache of the source node, both the traditional method and the MPLS algorithm intended for use in this paper need to send the flow table under the source node, so the load of such flow table always exists. The second type is the flow entry that is sent by the switch on the path after the controller calculates the flow path. Such a flow entry is a non-essential load because it can be replaced with a data prefix using the MPLS algorithm. However, it is also necessary to make a trade-off between the data link load and the linear control link load, and select some of the switches to send the flow table.

2.2. Problem modeling

In this paper, consider a software-defined network structure represented by a triple, $S(U, V, E)$. One part is $U = \{u_1, u_2, \ldots, u_m\}$ for all controllers and the other is $V = \{v_1, v_2, \ldots, v_n\}$ for all switches. Where $m = |U|$ is the number of controllers and $n = |V|$ is the number of switches. In a multi-controller SDN network, there is $m < n$, and for each switch $v_i \in V$, there is a controller $u_j \in U$ connected to it. For the switch network topology, we abstract it into $G \in \mathbb{S}$ and $G = (V, E)$, where $E$ is the set of all data links, and for each link $e_i \in E$, we use $c(e_i)$ to indicate its link capacity. For each switch $v_j$, there is a downlink capacity, denoted by $c_d(v_j)$, and a flow entry capacity, denoted by $c(v_j)$.

For the downlink control link congestion problem, this paper uses an improved MPLS algorithm to solve the problem. (1) Periodically collect the flow table installation distribution and interface status based on the OpenFlow protocol. (2) Predict the switch flow table installation for the next cycle. The load redirects the traffic of the source node switch that may be overloaded in the next cycle to the
adjacent lower-load switch as the new source node for path calculation. (3) If there is a downlink control link load exceeding the threshold in the calculation path in the case of the flow, the flow table is sent in the MPLS mode, and the switch with the lower load in the path is selected to send the flow table, and the bandwidth of a part of the data link is borrowed to alleviate the problem that the downlink control link is overloaded.

For the switch \( v_i \), use \( F(v_i) \) to indicate the number of flow tables to be delivered, and define its threshold, as shown in formula (1).

\[
R_i = c_d(v_i) \times B
\]  

(1)

\( B \) is the threshold factor for which it is redirected. For the source node, when the number of outgoing flow tables exceeds the threshold, the partial flow is first redirected to the switch with lower neighbor load by using a wildcard, and \( V_i \) is used to represent all switch sets with only one hop from the \( v_i \). \( p_i = |V_i| \). We specify that all switches that exceed the threshold are set to \( V' \). When the threshold is exceeded, the \( f_i(v_i, v_j) \) strip flow is redirected from \( v_i \) to \( v_j \), \( j = 1, 2, ..., p_i \), \( v_j \in V_j \). For the path switch, if the switch load exceeds the threshold on the path of a certain flow, enable the MPLS mode to deliver the flow table, and select the switch with the lowest load on the path switch to send the flow table. For the sake of simplicity, we assume that all downlinks have the same link capacity, that is, all switch link capacities are \( c_d(v) \times B \), and the threshold can be uniformly expressed as formula (2).

\[
R = c_d(v) \times B
\]  

(2)

For the source node redirection process, to increase system resiliency, set the redirection lower limit \( T \), where \( T < R \). That is, the final implementation reduces the overload switch \( V_i \) load below the threshold or the lowest load among all neighbors.

As shown in Figure 1, the MPLS algorithm encapsulates the path information as an additional label in the original data packet, so that it only needs to be delivered at the source node and some switch nodes on the path. The flow entry encapsulates the label information. Other switches only need to read the port information in the MPLS and send the data packet from the corresponding port. This method uses the bandwidth of the data link to alleviate the tension of the flow entries in the switch.

Figure 1. MPLS algorithm

Since each MPLS label has a size of 4 bytes, its additional link overhead is defined as shown in formula (3).

\[
Inf(l_f) = 4 \times \sum_{z=0}^{e-a-z} (e - a - 1 - z)
\]  

(3)
Where \( l_f \) represents the path vector of flow \( f \),
\[
    l_f = [x'_1, ..., x'_{a}, ..., x'_{b}, ..., x'_{c}, ...]
\]
\( x'_f \) indicates whether the flow table is installed on switch \( i \), if it is, \( x'_f = 1 \), otherwise \( x'_f = 0 \). And for all \( b \in (a,c) \), there is
\[
    l_f = [x'_1, ..., x'_{a}, ..., x'_{b}, ..., x'_{c}, ...]
\]
\( y^m_f \) indicates whether to use the MPLS label method for routing. The optimization problem optimization target is as shown in equation (7), and the constraint conditions are as shown in formula (8)-(11):
\[
    \min \mu, \forall v_i \in V
\]
\[
    \frac{1}{c_d(v) - \delta_i} < \tau
\]
s.t.
\[
    \delta_i = \sum_j (y^m_j \cdot x'_j + (1 - y^m_j)) < c(v)
\]
\[
    Inf(l_f) < C(f)
\]
\[
    y^m_f \in \{0, 1\}
\]

The constraint (8) indicates that the entry insertion delay is less than the threshold \( \tau \), the constraint (9) indicates that the number of flow tables in the switch cannot exceed the switch flow table capacity, the constraint (10) indicates that the MPLS label overhead is less than the threshold, and the constraint (11) indicates the \( y^m_f \) value.

3. Algorithm design

For the constraints and optimization goals proposed above, the workflow is first explained by an simple example. For the topology shown in Figure 2

3.1. Source node redirection algorithm

According to the prediction issued by the system to the next unit time flow table, when the source node may be in the downlink control link congestion, a wildcard is installed in the switch in advance, and a part of the flow is redirected to the neighbor switch, and the idle neighbor node is used as the new one. The source node performs the reporting of the Packet-In message and the sending of the flow table. Since there are multiple situations in the load gap between the neighboring nodes, the following rules are formulated to redirect the traffic:

Rule1: if \( F(v_i) < R \) then
\[
    f_i(v_i, v_j) = 0
\]
When $F(v_i) > R$, it means the switch load is greater than the redirection threshold, then $v_i$ has a neighbor switch $v_{i1}, v_{i2}, ..., v_{ip} \in V_i$, and $F(v_{i1}) < F(v_{i2}) < ... < F(v_{ip})$, then

Rule 2: if $F(v_{i1}) < R$ and $F(v_i) - T < R - F(v_{i1})$ then

$$f_i(v_i, v_{i1}) = F(v_{i1}) - T$$

Rule 3: if $F(v_{i1}) < R$, $F(v_i) - T > R - F(v_{i1})$ and $\forall v_{ij}, j = 1, 2, ..., m, m < p$, there is $F(v_{ij}) < R$ and $(m-1)R - \sum_{j=1}^{m-1} F(v_{ij}) < F(v_i) - T < mR - \sum_{j=1}^{m} F(v_{ij})$ then

$$f_i(v_i, v_{ij}) = R - F(v_{ij}), j = 1, 2, ..., m-1$$
$$f_i(v_i, v_{im}) = F(v_i) - (m-1)R + \sum_{j=1}^{m-1} F(v_{ij})$$

Rule 4: if $F(v_{i1}) < R$ & $F(v_i) - T > R - F(v_{i1})$ and $\forall v_{ij}, j = 1, 2, ..., m, m < p$ there is $F(v_{ij}) < R$ & $F(v_{i(m+1)}) > R$ & $F(v_i) - T > mR - \sum_{j=1}^{m} F(v_{ij})$, then

$$f_i(v_i, v_{ij}) = R - F(v_{ij}), j = 1, 2, ..., m$$

Rule 5: if $F(v_{i1}) > R$ and $F(v_i) - R > F(v_{i1}) - R$, then

$$f_i(v_i, v_{i1}) = \frac{F(v_i) - F(v_{i1})}{2}$$

Rule 6: if $F(v_{i1}) > R$ and $F(v_i) - R < F(v_{i1}) - R$, then

$$f_i(v_i, v_{i1}) = 0$$

The redirection algorithm is shown in Algorithm 1:

---

**Algorithm 1: source node flow redirection algorithm**

**Input:** redirection threshold $R$, adjacency matrix $V$, node load situation in the system

1. $\forall v_i \in V$ & $F(v_i) > R$
2. for $v_{ij} \in V_j, i = 1, 2, ..., n$
3. if $F(v_{ij}) < R$
4. if $F(v_i) - T < R - F(v_{ij})$
5. $f_i(v_i, v_{ij}) = F(v_i) - T$
6. else
7. $f_i(v_i, v_{ij}) = R - F(v_{ij})$
8. end if
9. else if $F(v_i) - R > F(v_{ij}) - R$
10. $f_i(v_i, v_{ij}) = \frac{F(v_i) - F(v_{ij})}{2}$
11. end if
12. end for

**Output:** Redirect traffic number and direction
3.2. Transmission path MPLS algorithm

For a stream that needs to be transmitted using the MPLS algorithm, the most important thing is to select the switch that sends the flow table in the path. In this paper, the greedy algorithm is adopted, and the switch with the lowest load on one path is selected as the intermediate node each time until the switch with no load lower than the threshold $R$ is optional or the path cost of the MPLS algorithm is lower than a certain threshold $C'$, and then the flow table is performed. Send and calculate the next stream, as shown in Algorithm 2:

### Algorithm 2: MPLS algorithm switch selection

**Input:** flow path vector $l_f$, switch load $F(v_i)$ on the path

**Initialization:** The switch collection sending the flow table $V_d = \{v_i\}$

1. Sort the switches on the path $l_f$ by load size:

   $F(v_1') < F(v_2') < ... < F(v_n')$

2. for $v_i' \in l_f, i = 1,2,...,n$

3. if $F(v_i') > R$ or $Inf(l_f) < C'$

4. break

5. else add $v_i'$ in $V_d$

6. end if

7. end for

**Output:** The switch collection sending the flow table $V_d$

3.3. Algorithm complexity analysis

For each flow table installation request per unit time, the calculation process of the algorithm includes: (1) detecting the overload condition of the switch; (2) selecting the traffic for redirection; and (3) selecting the traffic for the MPLS label installation. When it detects that the switch is overloaded, it changes back to start the subsequent process, traverses all neighbor nodes of the overloaded switch, performs a redirection, and then selects the redirected flow to perform MPLS algorithm switch selection. The worst case is to select all the switches, so the complexity of the algorithm can be obtained.

4. Simulation and analysis

4.1. Performance index and contrast algorithm

Performance indicators: For the downlink control link congestion problem proposed in this paper, it will be analyzed based on the following three performance indicators: (1) The first indicator is the load factor $\omega$ of the system, and the closer the load factor is to 1, the more the total load of the system is proved. A load factor of more than 1 proves that the flow table to be installed in the system exceeds the maximum value that the system can carry, and a system crash may occur. (2) The second indicator is the maximum flow table insertion delay. It is known from Little's law that the system table entry insertion delay is $1/c_d(v) - \delta_i$. The simulation assumes that the flow table entry insertion speed is 200/sec. (3) The third indicator is the network throughput rate, which is the maximum number of traffic that the network is capable of carrying.

Contrast algorithm: The comparison algorithm selects two algorithms. The first one is the classic OSPF routing algorithm [13], which uses the most classic shortest path algorithm. The second is the
OPT-L algorithm in the literature [11], which is the linear optimal solution algorithm for linear relaxation of the LRD problem.

4.2. Simulation experiment

This experiment builds a simulation experiment environment on the mininet [14] platform, and selects a campus network topology with 100 switch nodes and 50 terminal nodes [15]. Set the network data link capacity to 100Mbps. A random algorithm is used to simulate the flow of random arrivals in the system. In order to reduce the impact of the random algorithm on the simulation results, each group of experiments is averaged 50 times to obtain the final simulation results.

After building the simulation platform, the maximum downlink control link load of the three algorithms is compared. The maximum downlink control link load is the load of the switch with the largest load in the system, which can measure the situation of the earliest point in the system. The simulation results are shown in Figure 3. It can be seen that as the number of flows arriving at a switch in the system increases, the maximum downlink control link load of the OSPF algorithm switch increases. The OPT-C algorithm can alleviate the maximum downlink control link load in the system, but can only reduce the load to 50% of the OSPF algorithm, and still cause a large delay in the system load exceeding the installation rate. The FRML algorithm in this paper is consistent with the OSPF algorithm load before the system load reaches the threshold of 200, but does not exceed the load limit per second. As the number of traffic in the system increases, the redirection and MPLS labeling mechanism triggers, which can reduce the load in the system near the threshold of 200 and maintain its slow growth rate. It can be seen that when the load is low, the FRML algorithm can not perform any creation and reduce the system overhead. When the threshold is exceeded, the FRML algorithm can reduce the significant system load, and the effect is better than the OPT-C algorithm.

![Figure 3. Maximum downlink load](image)

Next, the data link load factor and the system flow table insertion delay are simulated. The simulation results are shown in Figure 4. 4.(a) shows the relationship between the load factor and the total number of traffic in the system. It can be seen that as the number of traffic in the system increases, the load in the OSPF system increases significantly, and finally the system is fully loaded, which seriously affects system performance. The OPT-C system can improve the load on the OSPF system, but as the number of traffic increases, the load in the system will eventually reach full load. The FRML algorithm can reduce the system capacity in disguise due to the fundamental reduction of the number of flow tables sent to the system, greatly reducing the rate of full load. Figure 4.(b) shows the relationship between the maximum flow table insertion delay in the system and the number of flows in the system. It can be seen that as the number of traffic in the system increases, the OSPF system gradually reaches full load, so a serious delay increase occurs. Congestion causes the flow table entry insertion delay to increase exponentially, causing serious delay. In contrast, both OPT-C and FRML algorithms use a redirection algorithm, so there is no significant increase in the insertion delay of a switch flow table, but the number of flow tables that FRML needs to insert is relatively small.
Therefore, the FRML algorithm flow table entry insertion delay has certain advantages over the OPT-C algorithm.

![Load Factor and Flow Table Insertion Delay](image)

(a) Load Factor  
(b) Flow Table Insertion Delay  

Figure 4. Comparison of load factor and maximum flow entry insertion delay

Finally, the traffic throughput in the system is simulated. The results are shown in Figure 5. It can be seen that the OSPF algorithm is prone to excessive load on a single switch, which seriously affects the load capacity of the entire system and limits network throughput. The OPT-C algorithm can alleviate the load capacity of a single switch, but the total load capacity of the system remains unchanged. The throughput bottleneck will still be reached when the traffic increases to a certain limit. The FRML algorithm proposed in this paper reduces the number of flow entries in the system. Therefore, network throughput can be further increased and the capacity of the system can be increased.

![Network Throughput](image)

Figure 5. Network throughput

5. Summary

In this paper, for the downlink control link load problem in the SDN system, the method of redirecting and decentralizing the MPLS label is used to alleviate the system performance degradation caused by the low rate of the flow entry of the source node switch and the path switch. It has been verified on the SDN experimental platform, which reduces the link load by nearly 60% compared with the traditional OSPF algorithm, and reduces the link load by nearly 20% compared with the OPT-C algorithm. The maximum entry insertion delay is reduced by nearly 90% compared to the OSPF algorithm, which is slightly better than the OPT-C algorithm. At the same time, the network throughput is increased by nearly 200% and 30% compared to the OSPF algorithm and the OPT-C algorithm.
References

[1] McKeown N, Anderson T, Balakrishnan H, et al. OpenFlow: enabling innovation in campus networks[J]. ACM SIGCOMM Computer Communication Review, 2008, 38(2): 69-74. DOI: 10.1145/1355734.1355746

[2] Kannan K, Banerjee S. Compact TCAM: Flow entry compaction in TCAM for power aware SDN[C]//International conference on distributed computing and networking. Springer, Berlin, Heidelberg, 2013: 439-444. DOI: 10.1007/978-3-642-35668-1_32

[3] Wang A, Guo Y, Hao F, et al. Scotch: Elastically scaling up sdn control-plane using vswitch based overlay[C]//Proceedings of the 10th ACM International on Conference on emerging Networking Experiments and Technologies. ACM, Sydney 2014: 403-414. DOI: 10.1145/2674005.2675002

[4] Gude N, Koponen T, Pettit J, et al. NOX: towards an operating system for networks[J]. ACM SIGCOMM Computer Communication Review, 2008, 38(3): 105-110. DOI: 10.1145/1384609.1384625

[5] Kim T, Lee K, Lee J, et al. A dynamic timeout control algorithm in software defined networks[J]. International Journal of Future Computer and Communication, 2014, 3(5): 331.

[6] Wang J, Luo M. Packet prioritization in a software-defined network implementing OpenFlow: U.S. Patent 9,923,831[P]. 2018-3-20.

[7] Hohlfeld O, Kempf J, Reisslein M, et al. Scalability Issues and Solutions for Software Defined Networks[J]. 2019.

[8] Civanlar S, Tatlicioglu S, Balci A M. System and method for routing in software defined networks using a flow header: U.S. Patent Application 15/348,778[P]. 2018-5-10.

[9] Curtis A R, Mogul J C, Tournilhes J, et al. DevoFlow: Scaling flow management for high-performance networks[C]//ACM SIGCOMM Computer Communication Review. ACM, Toronto 2011, 41(4): 254-265. DOI: 10.1145/2018436.2018466

[10] Huang D Y, Yocum K, Snoeren A C. High-fidelity switch models for software-defined network emulation[C]//Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking. ACM, Hong Kong 2013: 43-48. DOI: 10.1145/2491185.2491188

[11] Wang P, Xu H, Huang L, et al. Control link load balancing and low delay route deployment for software defined networks[J]. IEEE Journal on Selected Areas in Communications, 2017, 35(11): 2446-2456. DOI: 10.1109/JSAC.2017.2760187

[12] Guo Z, Xu Y, Cello M, et al. JumpFlow: Reducing Flow Table Usage in Software-Defined Networks[J]. Computer Networks, 2015, 92:300-315.

[13] Spragins J. OSPF: Anatomy Of An Internet Routing Protocol [Book Reviews] [J]. IEEE Network, 1998, 12(6):4-4.

[14] Mininet: An instant virtual network on your laptop (or other pc) [EB/OL]. http://mininet.org/. 2014

[15] The network topology from the monash university [EB/OL]. http://www.eese.monash.edu.au/twiki/bin/view/InFocus/LargePacket-switchingNetworkTopologies. 2018