International Journal of Education (IJE) Vol.02, No.2, February 2020

The Research of OpenFlow Flow Table Idle Timeout Optimization

1th Zhaohui Ma, 2th Jianlong Fan() ,3th Suifeng Li ()

School of Information Science and Technology, Guangdong University of Foreign Studies
Guangzhou, China.

Abstract

SDN (Software defined networking), by separating the data plane from the control plane and providing a programmable interface, solves some problems in the traditional network, such as complex management, and is considered as the next generation network architecture. However, in the process of practical application and popularization of SDN, there are still a lot of technical problems to be solved, among which the performance of SDN is the most important. Aiming at the problem of limited flow table space of OpenFlow switch in SDN network architecture, this paper proposes a dynamic adjustment scheme for OpenFlow switch. Firstly, the SDN architecture, OpenFlow architecture, flow table structure and flow table matching process were described, and the existing flow table updating methods were analyzed. Then, considering the load of the controller and the remaining space of the flow table, the idle timeout mechanism of dynamic adjustment of flow table items was proposed, and the system was designed. Simulation experiments show that the new scheme can improve the performance of OpenFlow switch flow table more effectively than the original flow table update scheme.

Keywords

SDN, OpenFlow Switch, Flow Table Capacity, Flow Table, Idle Timeout

- Introduction
- SDN Network Architecture

With the development of computer network, the current network system and data center have become more and more bloated, more and more complex, and the amount of data is also larger and larger, so the network system designer needs to often change the network software, computer architecture and network resources according to the needs. However the traditional network architecture is more and more difficult to meet the needs of operators, enterprises and end users, because the decision-making ability of traditional network is distributed on each network component, which makes it very complex to add new network equipment. With larger network size, network management and configuration will become increasingly complex and error prone.
To overcome these shortcomings of traditional networks, professor Nick McKeown and his team first proposed the concept of SDN in 2009 [1]. SDN is an architectural approach that optimizes and simplifies network operations, integrating applications more closely with network services and interactions between devices, whether they are physical or virtual. The basic structure of SDN is to

physically separate the control plane and the data plane, and directly program the application layer. SDN consists of three parts: 1) The infrastructure layer(also called forwarding layer)is composed of network equipments, which combines the forwarding functions of the second and third layers of traditional network and is only responsible for data forwarding. 2)The control layer is responsible for formulating, calculating and controlling the forwarding strategy. 3) Application layer, namely the application program running on the controller, is used to realize some extended functions, such as quality of service, network monitoring, etc. Users can write different applications according to their own needs to meet different needs. The communication between the application layer and the control layer is called the north interface. Currently, there is no unified standard.
The communication between the control layer and the infrastructure layer is called the south interface.

- **OpenFlow Protocol**

Openflow is a standard of SDN south interface proposed by the ONF(Open Network Foundation)[2]. It is the communication protocol between the controller and the bottom switching equipment. The controller sends flow tables or any other messages to an Openflow switch through a transmission channel. Openflow switches match and forward each received packet according to the flow table items sent by the controller[3]. Once the match fails, the Openflow switch will send the Packet_In message to the controller requesting the controller to issue the flow table rules. When the controller receives the Packet_In message, it will calculate the rules for the switch to forward the unknown flow by the established rules and will send the Packet_Out message to the switch. The basic architecture of Openflow is shown in figure 1.

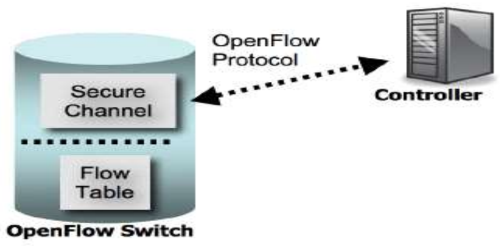
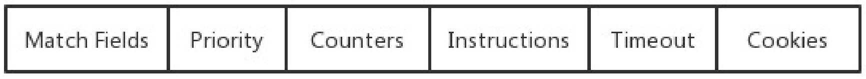


Fig. 1. Openflow architecture

- **Flow Table**

As shown in figure 2, the items in the flow table are composed of three parts :Header Fields, also known as matching Fields, which are composed of the matched Fields of the 12-tuple group; Counters, which are used to maintain a set of Counters, such as flow table related Counters, port related Counters, etc; Actions, which included one or more actions and is used to instruct the switch how to handle matched packets, such as flooding matched packets or forwarding them to the controller[4].



Match Fields	Priority	Counters	Instructions	Timeout	Cookies

Fig. 2. Flow table properties

- **Flow Table Matching Process**

When the data packet enters the OpenFlow switch, the data packet parsing module of the switch first parses the data packet and obtains the header information of the data packet. The header information of the packet is then matched with the matching field of the active flow table item. If the match is successful, the packet will conduct some operations such as forwarding according to the successful match action and refresh the soft timeout. If the match fails, table-miss will be triggered, and the switch will transmit the packet in the means of packet-in to the controller. After the controller calculates the path, a flow-mod message is sent to all switches on the packet forwarding path to install the new flow table item.

- **The Challenges of SDN**

As a new network architecture, SDN still faces many challenges to popularize and apply, for example, the short of unified standard for the northbound interface, the lack of universality of the network equipment, the security problems of each layer of SDN, and the excessive connection problems with the traditional network. However, the most critical issue is the performance of SDN. Currently, the forwarding device generally has two or more communication protocols at the same time, which poses a challenge to the controller to maintain flow tables of different protocols. With the increase of network traffic, it is more likely to come across the computing performance bottleneck of the controller. In addition, the network forwarding device runs OpenFlow protocol and maintains one or more flow tables. As the protocol version is updated, the space taken up by single flow table item is larger and larger, which requires more storage space. In practical applications, in order to meet the demand of high-speed data forwarding, switches tend to cache the flow table with TCAM (three-state content addressing register). However, due to the defects of TCAM, such as power consumption and cost, its flow table capacity space is often[5] small. For example, the commercially available TCAM switch based on HP5406zl can store up to 1500 flow table items[6], which can only meet the needs of small and medium-sized network applications. With the continuous expansion of the network scale, limited flow Table space can easily lead to overflow of flow Table, packet loss and other problems. At the same time, excessive triggering of table-miss will lead to the submission of a large number of packet-in messages to the controller, resulting in greater network delay, which is very unfriendly to the network quality. Therefore, this paper proposes an idle_timeout flow table optimization scheme which considers the residual flow table space and the load of the controller. Simulation results show that this mechanism can greatly optimize the flow table space.

- **The Structure of the Paper**

This paper is organized as follows:
Section 1: This section mainly expounds the background, significance and present situation of SDN research.
Section 2: Overview of realted work.
Section 3: An optimization strategy to adjust the timeout and other relevant research directions and present status.

Section 4: System design.
Section 5: Performance evaluation and analysis.
Section 6: Summary of this paper.

• The Related work

• ***The Introduction of Research Status***

At present, relevant researches on flow table optimization can be roughly divided into three directions:

- *The research based on flow table resources reuse.* The basic idea of the research is to combine and compress some associated flow table items through the design of specific algorithms, making the combined flow table items be able to match more data flow[7], saving the flow table space at the same time, in order to store more flow table items in a disguised form under the condition of the same hardware.
- *The optimization research based on flow table items idle timeout.* Its purpose is to realize the dynamic control of the number of active flow table items by optimizing the idle timeout of flow table items, so as to avoid the problem of unrestricted growth of flow t[8]able items and overflow of flow table.
- *The optimization research based on multiple flow table.* This method divides the matching rules into multiple sub-tables[9], establishes a decision tree of limited depth in each sub-table, reduces the space wasted by storing worthless wildcards in the flow table items, and then matches the data flow by cascaded multiple flow tables.

• *The Optimization Research Based on Flow Table Items Idle Timeout*

OpenFlow protocol defines the mechanism of flow table item timeout, including soft timeout(idle_timeout) and hard timeout. The soft timeout means that if the flow table item is not matched within the set time, it will be deleted in the flow table after the time expires. The hard timeout is that the flow table item is deleted from the flow table immediately after the set time, whether it was successfully matched or not during that period. Flow table items are deleted when either of the two timeout limits is reached. Therefore, the flow table space can be used more reasonably by setting a reasonable timeout. Since the hard timeout is usually several times more than the idle timeout, a reasonable idle timeout can better adapt to the high-rate network environment. The research on the optimization of the convection table timeout basically focuses on the optimization of the idle timeout. In the process of data forwarding, the processing of convection table items is the key. Insufficient active flow table items will reduce the matching rate, resulting in more table-misses, increasing the load of the controller, increasing the network delay, and reducing the network quality. On the other hand, if there are too many active flow table items and the remaining flow table space is insufficient, the network will be unable to cope with a large amount of new burst traffic, resulting in overflow of flow table and loss of data packets, etc. Therefore, it is necessary to balance the active flow table items and the remaining flow table space.However, the existing fixed time-out scheme is not satisfactory and cannot adapt to the changeable network environment. For example, literature [10] proposed a dynamic adjustment timeout mechanism based on exponential growth. This mechanism first records the most recent network data flow and gives the service flow an initial time to adapt to the shorter time limit. After the initial time expires, a new time determined by exponential growth is issued to adapt to the actual network data flow, and the timeout regression and routing calculation are added to improve the efficiency of the mechanism at the same time. From the perspective of data flow, literature [11] proposed a common network flow time limit strategy, which can be used for both SDN network and conventional network data flow. Although the above two schemes can improve the network performance to a certain extent, they also have shortcomings. The algorithm of the former to calculate the timeout time simply approximates the network data flow, and does not treat the data flow into categories. Moreover, exponential time does not necessarily adapt to the incoming data flow well. Although the latter divides the data flow into single-packet data flows and other data flows, it requires more space to store new data streams. In view of this, this paper makes optimization based on a dynamic adjustment scheme for OpenFlow switch which considering the load of the controller and the remaining space of the flow table. Simulation experiments show that the new scheme can improve the performance of OpenFlow switch flow table more effectively than the original flow table update scheme.

- THE Optimization of flow table item idle timeout based on controller load and remaining flow table space

- **Conceptual Background**

The flow table space of switch is limited. In order to cope with peak or network emergency traffic, most of optimization of the table items of stagnation timeout focuses on how to use the flow table space more reasonably, such as the method trying to realize the dynamic balance between the writing speed of the new flow table items and the failure speed of existing flow table items, or the method trying to predict the number of new flow, etc.. However these methods simply considers the only factor, i.e, the flow table space of switch. In fact the performance of the controller is a very important factor in SDN network. In addition, the existing method directly adjusts the soft timeout time of all active flow table items, which has higher requirements on the switch performance. At the same time, in the process of adjustment, the processing of the newly arrived flow will lag behind and increase the network delay. Therefore, this paper proposes a stagnation timeout optimization scheme that takes into account both the load of the controller and the existing flow table space, so as to make full use of the flow table space resources, and at the same time, try to avoid excessive load of the controller due to the optimization timeout, which will make the controller unable to carry out other operations. When there is a traffic request, the stagnation timeout is dynamically adjusted according to the existing flow table space and the current controller load, and the flow table entry is issued.

- **Dynamic Adjustment Mechanism**

As shown in figure 3, this mechanism regularly collects the current remaining flow table space of each switch, while monitoring the load of the controller.The paper takes the maximum value of the packet-in message which can be handled by the controller at the same time as the maximum load measure of the controller. When the load of controller is more than 80%, it means the current load of controller is big. In order to avoid that more packet-in messages submitted by switches need to be dealt with by the controller, the controller will directly issue the flow table items which contains the default soft stagnation time. If the load of the controller is less than 80%, it indicates that the current controller has sufficient computing power. The next step is to judge according to the active flow entries of the collected switches. When the remaining flow table space of the switch is sufficient, that is, the active flow entries is less than 30% of the maximum flow table space, the controller issues flow table items with a long idle timeout(priority 1). When the active flow entries of switch is great than 80% of the maximum flow table space, the controller issues flow table items with a short idle timeout(priority 2). When the active flow entries of switch is more than 30% of the maximum flow table space and less than 80% of the maximum flow table space, the flow table space of switch is considered to be in a state of being relatively healthy, and soft timeout setting is reasonable. It is no nesessary to adjust the timeout and the controller issued the default timeout.

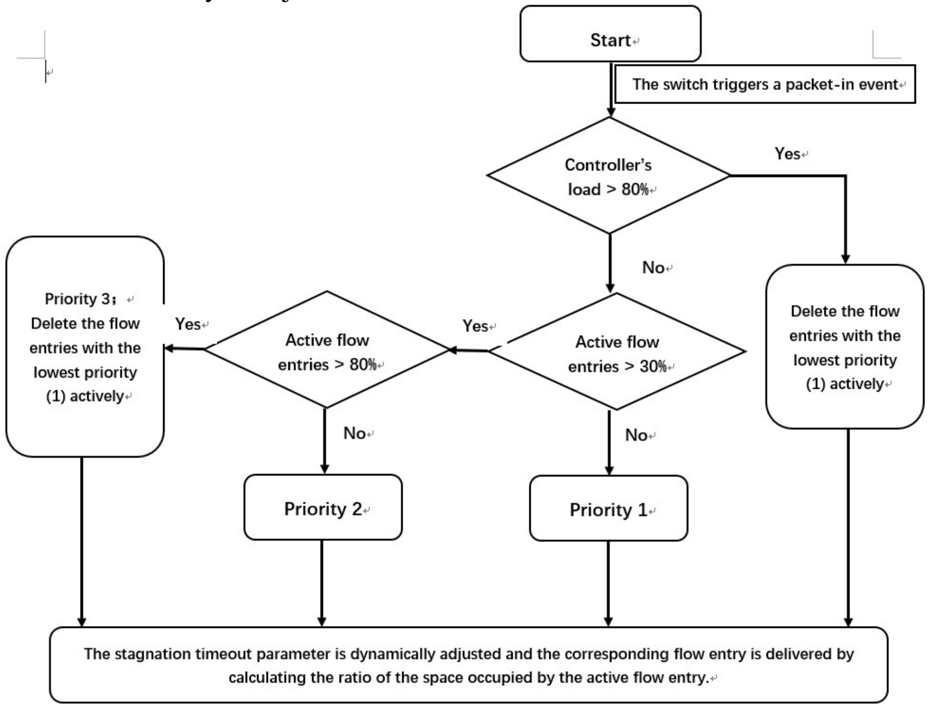

Start
The switch triggers a packet-in event
Controller's load > 80%
Yes
No
Priority 3; Delete the flow entries with the lowest priority (1) actively
Yes
Active flow entries > 80%
Yes
Active flow entries > 30%
Delete the flow entries with the lowest priority (1) actively
No
No
Priority 2
Priority 1
The stagnation timeout parameter is dynamically adjusted and the corresponding flow entry is delivered by calculating the ratio of the space occupied by the active flow entry.

Fig. 3. Dynamic Adjustment Mechanism

- **The Program Pseudocode**

The program pseudocode of dynamic Adjustment mechanism is shown below.

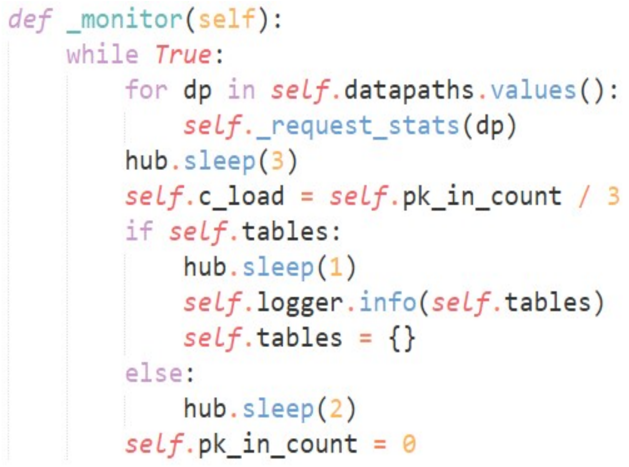
```
1：If self.tables
2：    If controller load<=80% of max_load then
3：        If flow entries>=80% of max flow table space
4：                idle_time--
5：        Elseif flow entries<=30% of max flow table space
6：                idle_time++
7：        Else  default idle_time
8：    Else
9：        default idle_time
10：END
```

- **The system design**

The system design is based on RYU controller for secondary development. RYU controller is an open source controller based on Python language, which supports OpenFlow v1.0 to OpenFlow v1.5. Since RYU controller needs to run in Linux environment, the design and development platform of this system is Linux and the development language is Python. The system is divided into three functional modules: 1) Monitoring module; 2) Idle timeout adjustment module; 3) Flow table items issuing module.

- **Monitoring Module**

Counters in the OpenFlow v1.0 protocol maintain the flow table information of the switch, including max_entries of the flow table space and active_entries of the flow table. As shown in figure 4, the monitoring module of the controller can send flow table information requests to different switches through the API of OFPTablesStatus provided by OpenFlow v1.0 protocol by using datapath parameter. It can get the return information of the switch by listening for EventOFPTablesStatus event, and then read the max_entries of the table space and active_entries of the current active flow table in the returned information，and then collect and store the information in Python's named tables dictionary. Since collecting the flow table information too frequently will occupy the effective network bandwidth, the monitoring module collects the flow table information of all switches every 5s for the judgment basis of the module to adjust the stagnation timeout event. At the same time, the module will monitor the load of the controller. When there is packet-in message, the packet-in counter will accumulate continuously, and c_load is the average load of the controller in this period of time.

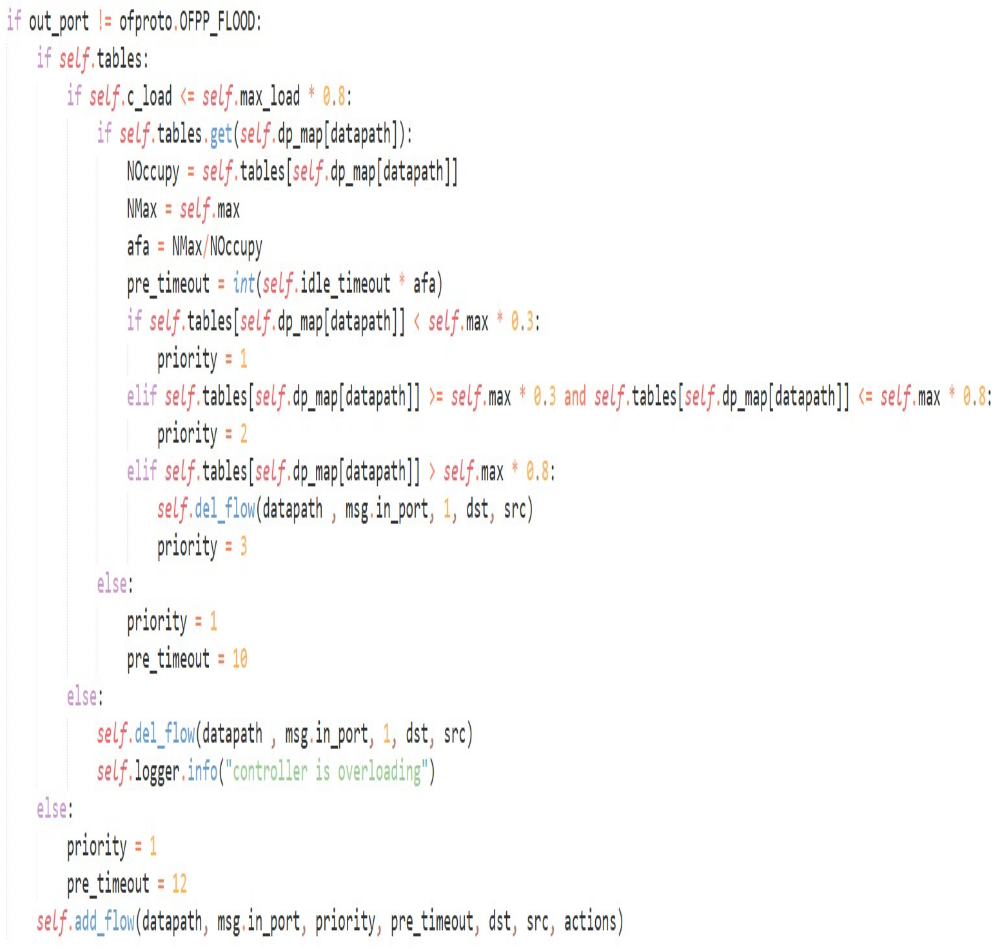


```
def _monitor(self):
    while True:
        for dp in self.datapaths.values():
            self._request_stats(dp)
        hub.sleep(3)
        self.c_load = self.pk_in_count / 3
        if self.tables:
            hub.sleep(1)
            self.logger.info(self.tables)
            self.tables = {}
        else:
            hub.sleep(2)
        self.pk_in_count = 0
```

Fig. 4. Monitoring module

- **Idle timeout Adjustment Module**

As shown in figure 5, the module reads the tables dictionary maintained by monitoring module, to identify different switch by datapath. Through comparing the number of the current active flow table entry with the prescribed upper limit and lower limit value, by judging the current controller load transferred by monitoring module, according to the adjustment mechanism mentioned in the third part of the article, invoke the function added with flow chart, add flow table items of different timeout of different switches respectively.

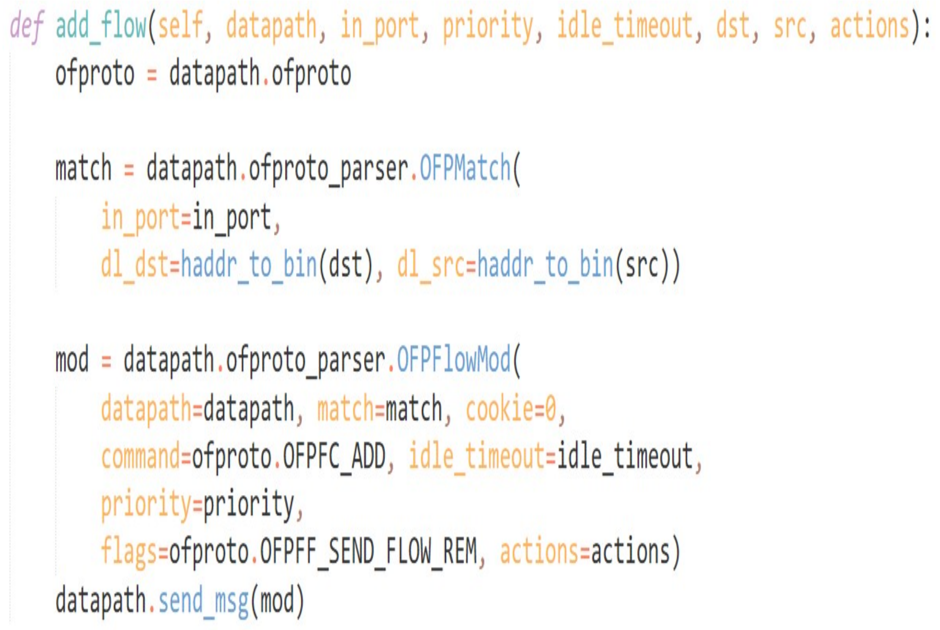


```
if out_port != ofproto.OFPP_FLOOD:
    if self.tables:
        if self.c_load <= self.max_load * 0.8:
            if self.tables.get(self.dp_map[datapath]):
                NOccupy = self.tables[self.dp_map[datapath]]
                NMax = self.max
                afa = NMax/NOccupy
                pre_timeout = int(self.idle_timeout * afa)
                if self.tables[self.dp_map[datapath]] < self.max * 0.3:
                    priority = 1
                elif self.tables[self.dp_map[datapath]] >= self.max * 0.3 and self.tables[self.dp_map[datapath]] <= self.max * 0.8:
                    priority = 2
                elif self.tables[self.dp_map[datapath]] > self.max * 0.8:
                    self.del_flow(datapath , msg.in_port, 1, dst, src)
                    priority = 3
            else:
                priority = 1
                pre_timeout = 10
        else:
            self.del_flow(datapath , msg.in_port, 1, dst, src)
            self.logger.info("controller is overloading")
    else:
        priority = 1
        pre_timeout = 12
    self.add_flow(datapath, msg.in_port, priority, pre_timeout, dst, src, actions)
```

Fig. 5. Idle timeout adjustment module

- **Flow Table Items Issuing Module.**

When the switch fails to match the packet, the packet will be sent to the controller in the form of packet-in. When the controller receives the packet-in message from the switch, the packet will be parsed by the processing module. If it is layer 2 forwarding, extract the source MAC address and destination MAC address of the packet, and the packet enters in_port, and then query MAC address maintained by the module and the mapping table of port , mac_to_port. If it is found successfully, the port corresponding to the destination MAC address is taken as the outbound port of the packet, then the soft timeout adjustment module is called to decide whether to modify the idle timeout, and finally the flow table distribution function in figure 6 is called. At the same time, the module checks whether the input port and source MAC address are in the mapping table and updates the mapping table if not. If the port corresponding to the destination MAC address is not found in the mapping table, the module will change the forwarding mode of the packet to flood mode. The flow table issue or flow table installation function encapsulates information about the flow table entry and sends it to the appropriate switch according to the datapath parameter. The switch installs and updates the flow table entry.



```
def add_flow(self, datapath, in_port, priority, idle_timeout, dst, src, actions):
    ofproto = datapath.ofproto

    match = datapath.ofproto_parser.OFPMatch(
        in_port=in_port,
        dl_dst=haddr_to_bin(dst), dl_src=haddr_to_bin(src))

    mod = datapath.ofproto_parser.OFPFlowMod(
        datapath=datapath, match=match, cookie=0,
        command=ofproto.OFPFC_ADD, idle_timeout=idle_timeout,
        priority=priority,
        flags=ofproto.OFPFF_SEND_FLOW_REM, actions=actions)
    datapath.send_msg(mod)
```

Fig. 6. The process of issuing the flow table

SIMULATION RESULTS AND ANALYSIS

- **Test Environment**

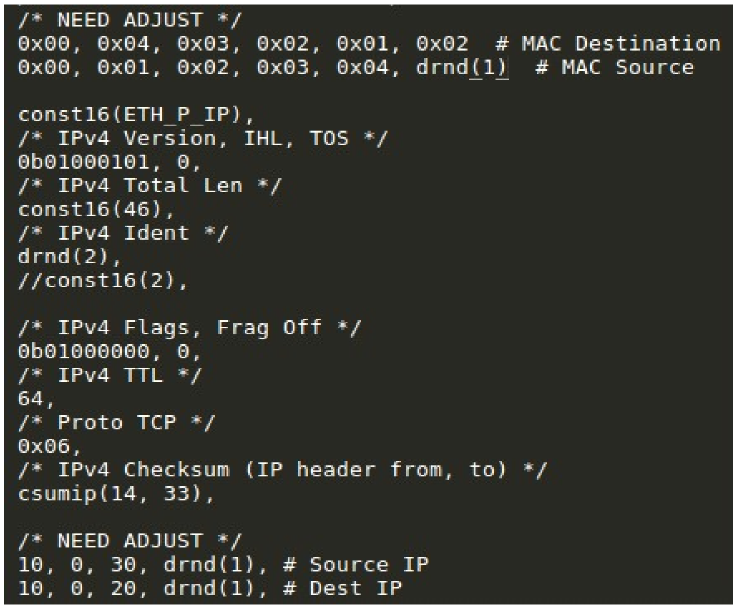
The test of this paper is based on Ubuntu 16.04 system, using OpenVSwitch switch and a single RYU controller, which is implemented in mininet2.2.2 experimental platform. Since OpenVSwitch supports 255 flow tables by default and each flow table supports 1 million maximum flow table entries, the maximum flow table space needs to be controlled through the management interface provided by OpenVSwitch.

- **Test Tools and Test Method**

Test tools. In this paper, two kinds of network performance testing tools are used for performance testing. The first is netsniff-ng, which is a system-level high-performance network sniffer based on Linux system platform. In this paper, the trafgen tool that comes with this program is used for traffic packet generation. The second is the iperf network performance testing tool.

*Test method.*Using trafgen tool, different scripts were run on switch s3, s4, s5 and s6 to generate and inject traffic, and part of the traffic generated the code of the script, as shown in figure7.

Traffic is generated by self-constructing the SYN packet of TCP's three-way handshake, where the key is to randomly generate MAC addresses and IP addresses to generate different traffic, and the rest is other components of the SYN packet, such as headers and content. The remaining flow table space for all switches is then calculated and averaged as a performance indicator. Iperf tool is used to test the quality of network communication. In addition, customers can use different parameters to control the sending of UDP packet, and test repeatedly to find the best parameters to test the pacet loss rate.



```
/* NEED ADJUST */
0x00, 0x04, 0x03, 0x02, 0x01, 0x02  # MAC Destination
0x00, 0x01, 0x02, 0x03, 0x04, drnd(1)  # MAC Source

const16(ETH_P_IP),
/* IPv4 Version, IHL, TOS */
0b01000101, 0,
/* IPv4 Total Len */
const16(46),
/* IPv4 Ident */
drnd(2),
//const16(2),

/* IPv4 Flags, Frag Off */
0b01000000, 0,
/* IPv4 TTL */
64,
/* Proto TCP */
0x06,
/* IPv4 Checksum (IP header from, to) */
csumip(14, 33),

/* NEED ADJUST */
10, 0, 30, drnd(1), # Source IP
10, 0, 20, drnd(1), # Dest IP
```

Fig. 7. The process of issuing the flow table

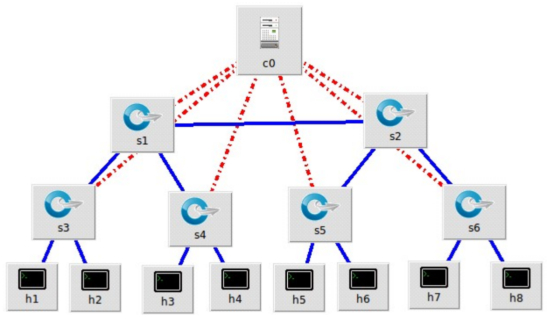


Fig. 8. The simulation topology

The Simulation Topology

As shown in figure 8, the test adopts the classical tree topology, which is closer to the application scenario in a small network than other topology types.

- *Performance Indicators*
- *Packet Loss Rate*

(1)

PLR refers to Packet Loss Rate, P_l is the total number of packets lost in a cycle time, and P_r is the total number of Packet requests in a cycle time.

- *Remaining flow table space.* Remaining flow table space is the maximum number of flow table entries that the switch can store in a period of time without overflow.
- **Test Results and Analysis**

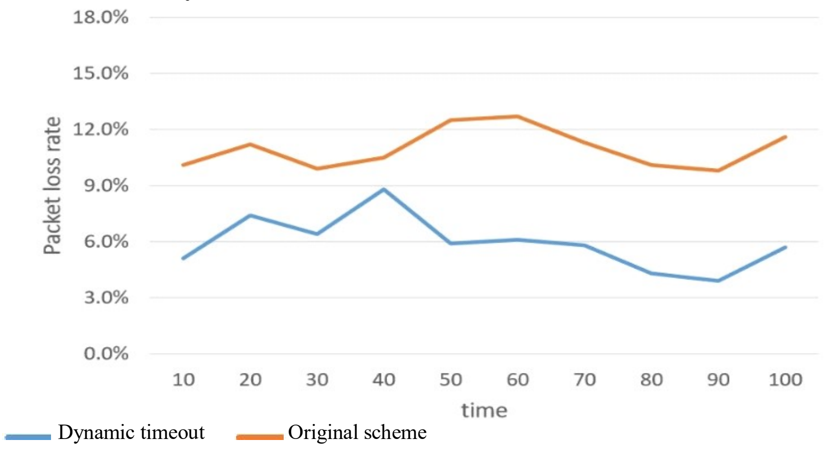


Fig. 9. Packet Loss Rate

The test data were obtained at a rate of 230 packets per second. Figure 9 shows the original flow table updating scheme and the dynamic scheme proposed in this paper. With the passage of time, the packet loss rate changes. The blue line represents the dynamic adjustment flow table updating scheme, while the red represents the original flow table updating scheme. According to the curve in the figure, the dynamic adjustment scheme of soft timeout proposed in this paper keeps the loss rate around 6%, with the lowest to 3.9% and the highest up to 8.8%, and the average loss rate is 5.94%. In the original flow table updating scheme, the loss rate fluctuated from about 10% to 13%, with the highest reaching 12.7% and the lowest reaching 9.8%, and the average loss rate was 10.97%. Compared with the original scheme, the loss rate of the dynamic scheme proposed in this paper is 5.03% lower on average. After analysis,during the peak time of the network traffic since there is no sufficient remaining flow table space in the original plan, it will lead to flow table overflow, thus discarding packets, which causes the packet loss rate to increase. However when remaining flow table space is not enough, the dynamic adjustment scheme proposed in this paper will reduce the stagnation of the flow table items timeout, speed up failure rate of the flow chart, spare more flow table space to respond to network traffic and reduce the packet loss, so that the packet loss rate has remained at a relatively low level, improving the quality of network service.

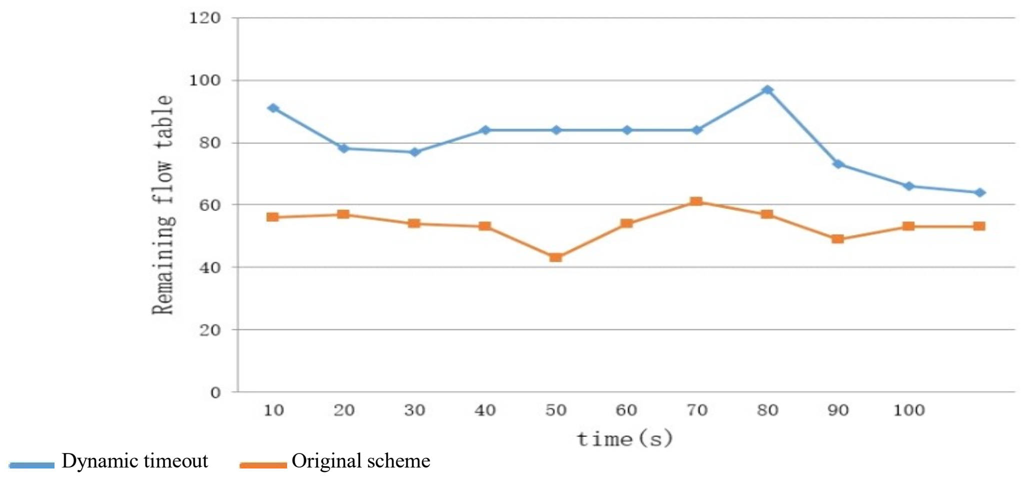


Fig.10. remaining flow table

At low load, the dynamic adjustment scheme selects longer soft timeout,so the number of active flow table items gets more.The more number of active flow table items will increase data flow matching success rate, which is no doubt more effective compared with the original scheme. During the middle section,in the original plan timeout is changeless, and the timeout which the dynamic adjustment scheme uses to issue the flow table items will be consistent with the original plan whether controller overloads or not. At the same time in order to avoid the overflow of flow table due to large amount of data table, the test will control the contract rate at 200 packets per second or so. The fluctuations are due to the nature of the test tool itself. Figure 10 shows the change of the space of the remaining flow table as time goes by, among which the blue line represents the dynamic adjustment flow table updating scheme proposed in this paper, and the red line represents the original flow table updating method. It can be seen from the figure 10 that because the dynamic scheme needs to consider the controller load and remaining flow table space at the same time and update the stagnation timeout of the flow table items dynamically,the fluctuations of the remaining flow table space is relatively large. In contrast, the original scheme directly issues flow table only according to the traffic requests without considering too many factors, so the fluctuation of the remaining flow table space is relatively small. At the same time, it is not difficult to see that in the 90s and 100s, the remaining flow table space of the two schemes is not much different. This is because the dynamic scheme considers the load of the controller and adjust the timeout time to 5s, which is consistent with the original scheme. The remaining space is not much different from that of the original scheme. In addition, figure10 shows clearly that when dealing with the large data flow, the remaining flow table space in the dynamic adjustment scheme is bigger and has greater ability to deal with more data flow. In the test the original flow table updating scheme has smaller remaining flow table space. The flow table space use is close to full load. If more data flow needs to be processed, it is likely to come to overflow, thus discarding arriving packets, reducing the quality of network service.

- Conclusions and expectations
- *Conclusions*

Traditional network is complex and difficult to manage. One of the reasons is that the control plane and data plane is tightly integrated, and network equipment use and maintenance are related to the different network equipment suppliers, for example, cisco has private EIGRP protocol, that is, routing protocol to increases the internal network management, which is not supported by other company's device; Another reason is that traditional network devices are different series of products, which means that different series of network devices may have different configuration methods and different management interfaces. These are not friendly to the owner or user of the network device.
However, with the emergence of SDN, it is possible to solve these problems left over by history.SDN supports dynamic programming to network forwarding devices through southbound interface. SDN decouples and separates the control plane and data plane, which makes the control plane become the center of the whole network and makes the control plane have global "vision"over the entire network, so the application can run better on the top floor. Based on the above reasons, compared with the traditional network, the application development and deployment of SDN is easier and simpler.

Although SDN has various advantages compared with the traditional network, SDN is after all a totally new network architecture. All aspects of the development is relatively immature and there are many problems to be solved to popularize SDN, such as SDN safety, SDN performance problems. For SDN safety, SDN has three layer architecture, northbound and southbound interface, which will bring five aspects of security issues and every aspect has multiple security challenges; SDN performance problem is an inevitable question if SDN wants to replace traditional network architecture. Along with the social development, the network data grows rapidly the user can't accept the low efficiency of network service for the network provider's convenience to manage the network. In order to SDN popularization, the research related to enhancing the performance of SDN is very hot, which can generally be divided into the research to enhance controller performance and the research to enhance the data plane performance. The dynamic timeout adjustment mechanism based on controller load and remaining flow table space of switch proposed in this paper is a scheme to optimize the forwarding performance of forwarding layer network equipment. This article first expounds the three layer architecture of SDN, OpenFlow architecture, flow table structure and matching process flow table, and then expounds the challenges of the SDN. This paper briefly introduces the current research on flow table optimization, analyzes the disadvantages of the optimization method, finally puts forward the method that comprehensively considers the controller load for flow table optimization, which realizes the flow chart optimization to the controller's largest capacity range and effectively improves the success rate of data forwarding.

- **Expectations**

For better optimization of flow table and improvement of switch performance, there are still some deficiencies in this paper. For example, limited by the experimental environment, simulation test cannot be truly achieved, and a good simulation test is the premise of deployment in the real environment. In addition this paper has not distinguished data flow. Data flow type and its duration is more closely related. If the data flow were not distinguished, the unified adjustment of its stagnation timeout is obviously unreasonable. Although the method without distinguishing data flow can bring performance improvement, the classifying of flow is one direction for further research. Therefore one of research directions in the future is to classify the data flow using machine learning method. Distribute different stagnation timeout according to the characteristics of the different data flow. A relatively simple solution is to divide data flow into elephant flow and mouse flow, for the elephant flow longer stagnation timeout time and mice flow shorter idle timeout.In the condition of the data flow classification, consider controller load and remaining flow table space to optimize flow table. It is hoped that multiple controllers can be assembled into and even switch migration scheme can be added, so as to improve the performance of the controller and even balance the performance of the controller load, and extend the optimization of the whole system to the control plane instead of just monitoring the load of the controller.

References

1) A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim,P. Lahiri, D. A. Maltz, P.Patel,and S. Sengupta. VL2: a scalable and flexible data center network. In SIGCOMM,2009.

2) Manyika J, Chui M, Brown B, et al. Big Data: The Next Frontier For Innovation,Competition, And Productivity[J]. Analytics, 2011.

3) Pallis G. Cloud computing: the new frontier of internet computing[J]. IEEE Internet Computing, 2010 (5):70-73.

4) Cisco . Cisco Visual Networking Index: Forecast and Trends,2017–2022.2018 https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-741490.html

5) Benson T , Akella A , Maltz D . Unraveling the Complexity of Network Management[A]//Proceedings of the 6th USENIX symposium on Networked systems design andimplementation. NSDI09[C], 2009;335-348.

6) Mc Keown N, Anderson T, Balakrishnan H, Parulkar G, Peterson L, Rexford J, Shenker S,Turner J. Open Flow: Enabling innovation in campus networks. ACM SIGCOMM CCR,2008,38(2):69-74.

7) Open Networking Foundation. Software-Defined Networking: The New Norm for Networks. ONF White Paper, 2012.

8) Zuo qingyun, Chen Ming, zhao guangsong, et al. Research on SDN technology based on Open Flow. Journal of software,2013,24(5):1078-1097.

9) Wang weizhen. Study on flow meter timeout optimization of SDN switch based on OpenFlow [D]. Zhengzhou university of light industry, 2019.6.

10) Shi Shaoping. Research on OpenFlow table optimization technology [D]. Zhengzhou university,2016.

11) Katta N,Alipourfard 0,Rexford J,et al.CacheFlow:Dependency-AwareRule-Cachingfor Software-Defined Networks[C]. Proc.ACM Sympos ium on SDN Research(SOSR),2016:1-12