A Universal Mechanism to Handle ION Packets in SDN Network

Lixuan Wu[1], Jiang Liu[1,2] , Tao Huang[1,2] , Weihong Wu[1] , Bin Da[3]

[1]State Key Laboratory of Networking and Switching Technology
Beijing University of Posts and Telecommunications
Beijing, China
[2]Beijing Advanced Innovation Center for Future Internet Technology
Beijing, China
[3] Network Technology Laboratory, 2012 Laboratories
Beijing Huawei Digital Technologies Co., Ltd.
Beijing, China

Emails: {shinning@bupt.edu.cn , liujiang@bupt.edu.cn, htao@bupt.edu.cn, 344259446@qq.com, dabin@huawei.com}

***Abstract*—Identity Oriented Networks (ION) provide mechanisms for scalability, mobility and operations across heterogeneous entities by disseminating unique identity of end points from their position in the network. However, current devices cannot parse the newly added ID field in 3.5 layer. This paper puts forward a universal Software Defined Networking (SDN)-based packet processing mechanism to parse information in high layers and exchange redundant information in low layers with key information in high layers at the entrances of network. Thus, the key field in high layer is visible in low layer and can be parsed by current protocol and routing devices. The article takes GPRS Tunneling Protocol (GTP) packets for example to explain the packet processing method. Besides, delay caused by the packet processing module is measured and an experiment is made to verify that parsing high-layer field succeeds and strategies on high-layer field can be made.**

Keywords-ION; SDN; OpenFlow; GTP; TEID.

I. INTRODUCTION

With the development of network technologies and user requirements, mobility has been a significant trend. Firstly, the number of mobile devices including M2M modules keeps increasing and is expected to be 11.6 billion by 2020. Secondly, mobile traffic has grown faster year by year. Mobile traffic content also tends to carry more video traffic including streaming video, which requires high bandwidth transmission capabilities. Thirdly, offload mobile traffic (traffic from dual-mode devices over Wi-Fi or small-cell networks) is taking more and more proportion of the whole traffic. In 2015, mobile offload traffic exceeded cellular traffic for the first time. In general, more devices, more traffic and more offload connectivity pattern should be taken into consideration in next-generation network.

In 5G era, network has five performance requirements: 1) bandwidth and speed throughput: 10Gps; 2) latency: less than 1ms; 3) scale: 10-100 times than Long Term Evolution (LTE) [1]; 4) session continuity: ubiquitous; 5) mobility speed: 500km/h. However, current LTE architecture has many constraints. The handoff delay and latency is noticeable. Due to requirement of global IP addresses, multi-homing features make IP addresses aggregation difficult and lead to large RT on routers.

In the context, Identity Oriented Networks (ION) has been put forward to improve mobility performance. The fundamental premise of ION is the one that provides mechanisms for scalability, mobility and operations across heterogeneous entities by disseminating unique identity of end points from their position in the network. A 3.5 layer is added between the IP layer and the TCP/UDP layer. An identity field is carried in the new layer to identify a node, an app or anything. ID can be bond with an IP address locator to complement forwarding. Thus, network has the following improvements: 1) native mobility; 2) ID-based Apps; 3) multi-homing ID with global scope; 4) context awareness based on ID profile; 5) ID-based security.

Although ION improves network performance in many ways, current switching and routing devices can only parse information no higher than the IP layer. The long-term trend is to develop new devices that are capable to parse the 3.5 ID layer information. However, it will take a long time to update all routing devices and Capital Expenditure (CAPEX) should also be considered. To solve the issue, this article put forward a universal SDN-based packet processing mechanism to parse information in high layers and exchange redundant information in low layers with key information in high layers. Thus, the key field in high layer is visible in low layer and can be parsed by current protocol and routing devices.

The paper is organized as follows. Section II introduces related knowledge of the example case. Section III specifically explains the SDN-based packet processing mechanism to parse the high-layer field incompatible with OpenFlow protocol and utilizes the field to control network at a smaller granularity. Section IV measures the delay caused by the packet processing module and verifies the success of parsing high-layer field and making strategies based on the high-layer field. Finally, conclusions are presented in Section V.

 ISBN: 978-1-61208-550-0

II. RELATED KNOWLEDGE

Since ION is in conceptual phase, this paper takes GTP packets for example to explain the packet processing of parsing high-layer information and utilizing high-layer information to handle packets. The augment supporting the analogy is that GTP packets face the similar problem in incapability of recognizing high layer information when being processed by OpenFlow [2] devices.

GTP is an IP-based communication protocol in Evolved Packet Core (EPC) [3]. The protocol consists of GTP-C, GTP-U and GTP', which are respectively used in the GTP control plane, GTP user plane and charging data transmission. Since this paper studies packet processing in data plane, GTP-U is the core concern. GTP-U protocol stack is shown in Figure 1. For uplink packets, the radio layer ends in eNodeB. ENodeB encapsulates GTP packets and establishes the GTP tunnel to S-GW. S-GW establishes the GTP tunnel to P-GW and P-GW decapsulates GTP packets and forwards it to Internet. For downlink packets, the processing procedures are reversed. There is a key field named Tunnel endpoint identifier (TEID) in GTP-U header. TEID is used to multiplex different connections in the same GTP tunnel. A GTP connection is uniquely confirmed by a source tunnel IP, a destination tunnel IP and a TEID. In this article, GTP packets refer to GTP-U packets without special statement.

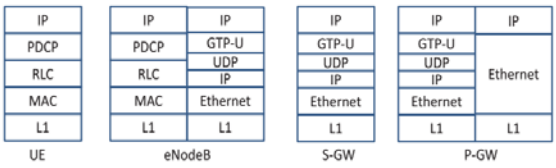

Figure 1. GTP-U protocol stack

Software-defined networking (SDN) [4] is an emerging solution for fine-grained control and management of networks. It separates the control plane (SDN controller) and data plane (switching and routing devices) of network. OpenFlow is a generally accepted southbound protocol between the controller and switching devices. There is a widely agreed trend that SDN should be integrated into 5G core network. However, GTP-U header locates over UDP layer. In current OpenFlow protocol, only information below layer 4 (including layer 4) can be parsed. Therefore, TEID is invisible in OpenFlow and GTP connection could not be recognized by OpenFlow switches. To address the problem, a packet processing method is put forward to parse the high layer field and exchange it with redundant fields in low layer.

III. PACKET PROCESSING IMPLEMENTATION

By analyzing the packet transmission within GTP tunnels, it could be found that forwarding decisions are made depending on the destination tunnel endpoint IP address. Therefore, during transmission in GTP tunnels, the source IP address is redundant information. Besides, both source IP address field and TEID field have a length of 32 bit in common. As a result, exchanging TEID field with source IP address field at the tunnel entrance would expose useful TEID field when forwarding the GTP packets in GTP tunnel without influencing normal processing mechanisms of GTP packets. Moreover, OpenFlow devices work normally complying with OpenFlow protocol since TEID field is already in layer 3 and could be parsed by OpenFlow protocol.

A. Architecture

In Evolved Packet Core (EPC) architecture, control functions and forwarding functions of S/P-GW are coupled, which restricts core network flexibility. Thus, separating control functions and forwarding functions of S/P-GW is a main trend of 5G mobile core network developments. Base on that, this article designs the core network architecture in Figure 2. This architecture retains most structure of the current EPC architecture. The major differences are: 1) separating the control functions and forwarding functions of S/P-GW; 2) introducing SDN controller to cooperate with S/P-GWc to manage the network between S-GWs and P-GWs; 3) data plane are OpenFlow-enabled and comply with the control of SDN controller.

The data plane consists of S/P-GW's user plane devices and OpenFlow devices, while the control plane consists of SDN controller, S/P-GW's control plane, Mobility Management Entity (MME), Home Subscriber Server (HSS) and Policy Control and Charging Rules Function (PCRF). In this case, MME works the same in EPC. It manages mobility, chooses S-GW for user equipment (UE) and establishes the GTP tunnel between eNodeB and S-GW. The control functions of S/P-GWs operate over SDN controller and communicate with it by JSONRPC messages to swap UE information and S/P-GW information. The S/P-GW control plane strategies are implemented in coordination with MME, HSS and PCRF, including user IP allocation and traffic flow template (TFT) assignment. SDN controller controls data plane devices with OpenFlow protocol and manage TFT with S/P-GW. S/P-GWs provide terminal of GTP tunnels and anchor GTP tunnels during handoff. Applying SDN to manage the transmission network makes it convenient to realize overhead control and routing optimization. Besides, this architecture is compatible with current 3GPP [5] standards, which is the smooth evolution for mobile core network to integrate SDN.

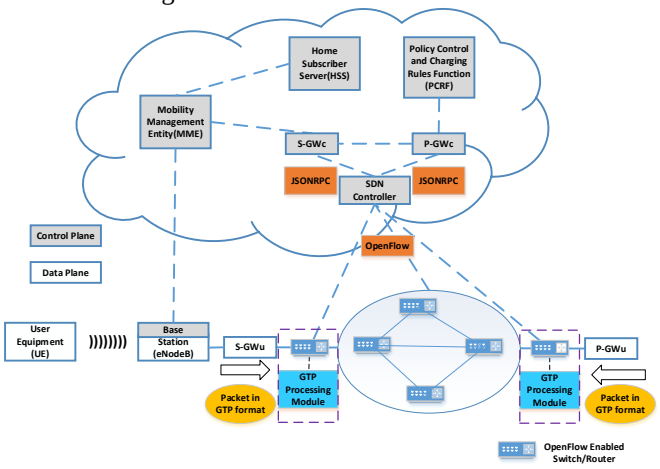

Figure 2. SDN-based mobile core network architecture

B. *Data Plane Implementation*

To parse the high layer information of TEID, a GTP processing module is attached to a logical port of an OpenFlow device at the entrances of S/P-GWs, cooperating with OpenFlow pipeline processing to handle GTP packets. The OpenFlow device with GTP processing module can be regarded as an extension of gateway functions. For uplink GTP packets at S-GWs and downlink GTP packets at P-GWs, TEID field and source IP address field should be exchanged to make the TEID field visible to SDN controller and OpenFlow devices between S-GWs and P-GWs. Thus, for downlink GTP packets at S-GWs and uplink GTP packets at P-GWs, TEID field and source IP address field should be exchanged to restore the previous packets.

The OpenFlow devices utilize OpenFlow pipeline processing to handle packets. There are at least five flow tables in the OpenFlow device. Table 0 has the highest priority. It matches GTP packets (udp_port is 2152) whose in_port is not connected to the GTP processing module and then sends them to the GTP processing module. Table 1 has the second highest priority. It matches GTP packets whose in_port is connected to GTP processing module and sends them to Table 2, otherwise it sends them to Table 3. Table 2 has the third highest priority. It matches GTP packets whose source MAC address is the S/P-GW to which it connects and sends them to Table 4, otherwise it outputs them to the connected S/P-GW. Table 3 forwards regular packets except GTP packets and Table 4 forwards GTP packets to be sent to the core network. The GTP processing module implements the exchange of high-layer TEID field and low-layer source IP field. Firstly, it parses the high-layer TEID field. Then, it exchanges TEID field with the low-layer source IP field. Lastly, it sends the processed packet back to the logical port where the packet comes. The OpenFlow multi-stage flow tables at entrances of S/P-GWs are shown in Figure 3.

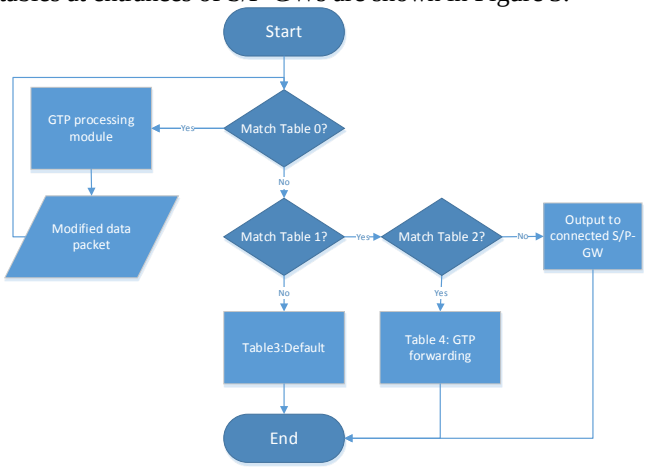

Figure 3. OpenFlow multi-stage flow tables at entrances of S/P-GW

After the packet processing at the entrance of core network, OpenFlow devices and SDN controller could have access to TEID field of GTP packets in the network between S-GWs and P-GWs. Thus, strategies based on TEID field are easy to implement.

C. *Control Plane Implementation*

As a result of exchange processing at GTP tunnel entrances, traffic control (routing optimization, Qos, etc.) at the granularity of GTP connections could be realized without changing working principles of OpenFlow devices and SDN controller.

Varieties of applications could be developed and run over SDN controller. Those applications leverage TEID field in layer 3 and make specific decisions on different TEIDs. An example of routing optimization based on TEID is given in Figure 4. Leveraging the advantage of the global view of SDN controller, different routing planning can be easily implemented.

Considering that different GTP connections have different routing demands, three routing planning modules are added to SDN controller. One implements shortest path planning, one implements maximum bandwidth path planning and another implements minimum delay path planning. When the controller receives packet-in messages, it matches TEID field and invokes corresponding path planning module for different TEIDs. The designed route is distributed to the network by flow-mod messages.

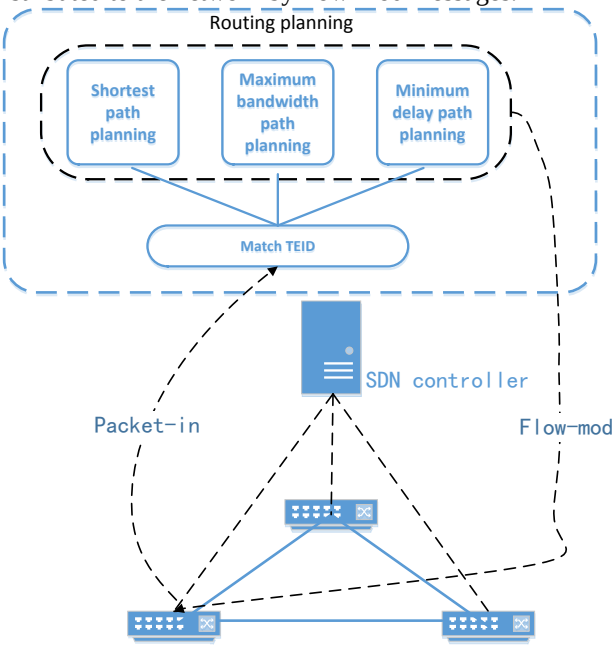

Figure 4. Routing planning example based on TEID

Routing modules based on different demands can be developed by similar methods.

IV. EVALUATION

Introducing GTP processing module into the network would cause extra delay during packet transmission. Thus, evaluation on delay should be taken into consideration. Besides, whether traffic control at the granularity of TEID is realized or not is also to be proved.

A. *Experimental Setup*

The experimental system is set on Ubuntu 14.04 LTS. Mininet [6] 2.3.0d1 was utilized to simulate the network and

Ryu [7] was utilized as an SDN controller. The topology is shown in Figure 5. SGW1 and PGW1 are two virtual machines which generate GTP packets. MS1 and MS2 are set to send GTP packets to a logical port attached by the GTP processing module. MH1 and MH2 are two network namespaces and work as GTP processing modules, respectively attached to the logical port of MS1 and MS2. S1, S2, S3, S4 and S5 work as switches in the core network.

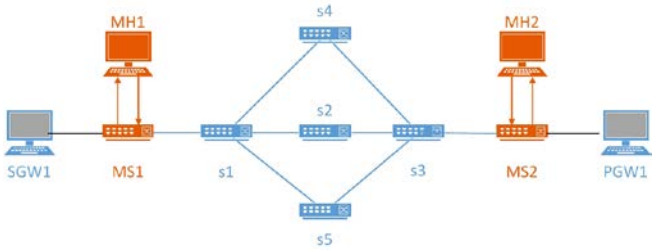

Figure 5. Experiment topology

- SGW1 and PGW1

OpenGGSN-0.91 [8] is used to build GTP tunnels. The PGW1 serves as a GGSN node that has one tunnel IP, and the SGW1 serves as an SGSN node that has several tunnel IPs. The TEID of each tunnel is allocated by orders. For the purpose of a simple design, the IP address of SGW1 and PGW1 are 10.0.0.1 and 10.0.0.2, and the MAC addresses are 00:00:00:00:00:01 and 00:00:00:00:00:02.

- MH1 and MH2

MH1 and MH2 work as the GTP processing module, aiming at exchanging source IP field with TEID field. There are two network devices in MH1, including eth0 and etSGW1. EtSGW1 works in a promisc mode to get each packet it receives. And eth0 works in multicast mode and has an IP address of 10.0.0.3 and a MAC address of 00:00:00:00:00:03. A python application named gtp_modify.py is running on MS1 to realize the GTP process module's functions. The application firstly captures a GTP packet from etSGW1, and then it cuts out the data segment and parses TEID field by counting the bit position. After getting the TEID, it exchanges the TEID field with source IP address field, and finally sends the packet back to the network through eth0.

Except that MH2's eth0 has an IP address of 10.0.0.4 and a MAC address of 00:00:00:00:00:03, the other settings and working pattern of MH2 are the same as MH1's.

- MS1 and MS2

MS1 is located between SGW1 and core networks, connected to MH1. And MS2 is located between PGW1 and core network, connected to MH2. The flow tables in MS1 and MS2 are described in Section III. GTP-U packets that should be sent into or out of core network will be delivered to GTP processing module.

- Ryu controller

Ryu serves as a SDN controller and is mainly responsible to set flow tables to OpenFlow switches. It will create a default route at exactly the time when the network is built, so that some control message like GTP-C can be forwarded successfully.

B. *Result Analysis*

In order to verify traffic control abilities at the granularity of GTP connections, there are three GTP connections in our scenario. The PGW1 has one tunnel IP of 192.168.0.1, while SGW1 has three tunnel IPs: 192.168.0.2 for tunnel1, 192.168.0.3 for tunnel2 and 192.168.0.4 for tunnel3. Besides, tunnel1's TEID is 0x00000001, and tunnel2's is 0x00000002, and tunnel3's is 0x00000003. Iperf [9] is utilized to generate traffic flows. Flow1 is sent at a speed of 100kbps in tunnel1, while flow2 is at a speed of 80kbps in tunnel2 and flow3 is at a speed of 60kbps in tunnel3. Based on TEID and destination IP address, different routes are designed. Flow1 is designed to go through s1->s4->s3, while flow2 is designed to go through s1->s2->s3 and flow3 is designed to go through s1->s5->s3. Traffic bandwidth is measured in each route, and the delay caused by MH1's GTP processing module is also detected.

Figure 6 shows the delay caused by GTP processing module in MH1. It can be seen that the delay range from 46ms to 98ms. The average delay is approximately 75ms. The delay value is a bit large. However, the delay will only be generated at the entrance of the whole network and the value is relatively stable regardless of the network scale. When network scale expands and the whole delay increases, the delay caused by GTP processing module tends to have a smaller influence to the whole network. Besides, the delay can be diminished by promoting the hardware performance.

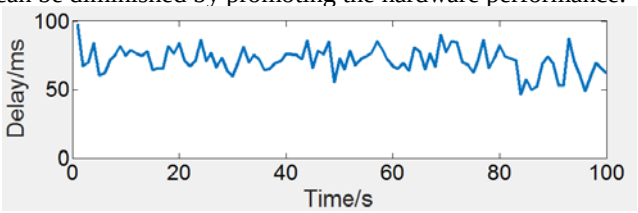

Figure 6. Delay caused by GTP processing module in MH1

Figure 7 shows traffic bandwidth in route s1->s4->s3, s1->s2->s3 and s1->s5->s3. It can be seen that the average bandwidth on the three routes are approximately 100kbps, 80kbps and 60kbps. It matches the bandwidth of flow1, flow2 and flow3, which illustrates the route design based on TEID is realized. Besides, Wireshark [10] is also utilized to capture packets at switch s2, s4 and s5. The results show that only packets with TEID 1 appear at switch s4, while only packets with TEID 2 appear at switch s2 and only packets with TEID 3 appear at switch s5. It further verifies that traffic control at the granularity of GTP connections is realized.

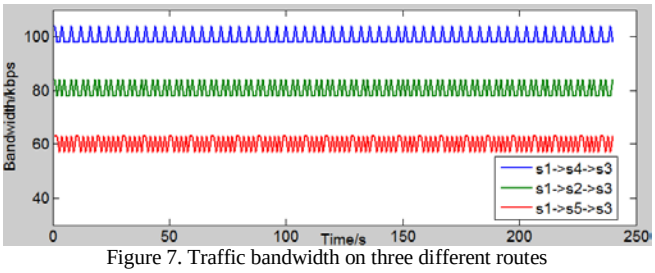

Figure 7. Traffic bandwidth on three different routes

Results show that traffic control abilities at the granularity of GTP connections are achieved.

V. CONCLUSIONS

This paper designs a universal SDN-based mechanism to handle packets containing high-layer field incompatible with OpenFlow protocol, without extending protocols or upgrading devices. An example of handling GTP packets is shown to explain the processing mechanism. Besides, the processing delay is measured and an experiment verifies that TEID field is successfully parsed and strategies at the granularity of TEID can be made. By utilizing the packet processing mechanism, ION packets can be transferred in SDN network and strategies at the granularity of ID field can be made. The mechanism could be universally utilized to tackle the problem of parsing a relatively small field incompatible with OpenFlow protocol as a temporary solution.

ACKNOWLEDGMENT

The authors thank Fei Yang and several engineers from Huawei for their valuable knowledge support.

REFERENCES

[1] "LTE", [Online]. Available from: http://www.3gpp.org/technologies/keywords-acronyms/98-lte 2017.04.01

[2] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, et al, "OpenFlow: enabling innovation in campus networks," ACM SIGCOMM Computer Communication Review, vol. 38, no. 2, 2008, pp. 69-74

[3] "EPC", [Online]. Available from: http://www.3gpp.org/technologies/keywords-acronyms/100-the-evolved-packet-core 2017.04.01

[4] N. McKeown, "Software-defined networking," INFOCOM keynote talk, vol. 17, no. 2, 2009, pp. 30-32.

[5] "3GPP", [Online]. Available from: http://www.3gpp.org/ 2017.04.01

[6] "Mininet", [Online]. Available from: http://mininet.org/ 2017.04.01

[7] "Ryu", [Online]. Available from: https://osrg.github.io/ryu/ 2017.04.01

[8] "OpenGGSN", [Online]. Available from: https://sourceforge.net/projects/ggsn/ 2017.04.01

[9] "iPerf", [Online]. Available from: https://iperf.fr/ 2017.04.01

[10] "Wireshark", [Online]. Available from: http://wireshark.com/ 2017.04.01

 ISBN: 978-1-61208-550-0