P4-based implementation of traffic engineering with service priority

Kouji Hirata*, Takumi Tabuchi*, Hideyoshi Miura*, and Shohei Kamamura†
*Faculty of Engineering Science, Kansai University, Osaka 564-8680, Japan
†Faculty of Science and Technology, Seikei University, Tokyo 180-8633, Japan
Email: {hirata, k720081, k846996}@kansai-u.ac.jp, shohei-kamamura@st.seikei.ac.jp

***Abstract*—With the growth of network services in recent years, the integrated design of service-guaranteed virtual private networks and traditional best-effort networks has been required. For such integrated design, traffic engineering is an important technology in order to accommodate the rapidly increasing traffic. In the past, a traffic engineering method considering service priority has been proposed, which determines routing paths of traffic for each network service, assuming that various types of services are deployed on an integrated network. In this paper, we examine the implementation of the traffic engineering method, using P4 language which can describe data plane operations in software-defined networking. We confirm the operation through demonstration experiments.**

I. Introduction

With the growth of network services in recent years, the network traffic has been rapidly increasing. In order to efficiently accommodate the network traffic, the integrated design of service-guaranteed virtual private networks in addition to traditional best-effort networks has been required. Traffic engineering [7], [8] is an important factor in order to determine the routing paths of each network traffic according to the network state for the integrated design. The recent development of Software-Defined Networking (SDN) [6] enhances the feasibility of dynamic traffic engineering that can dynamically change the routing paths. Because it is difficult for us to predict the variation of traffic demands, the utilization of network resources generally becomes unbalanced with the time elapsed. Applying the dynamic traffic engineering could resolve such unbalanced situations.

We can efficiently utilize the network resources with the traffic engineering. On the other hand, the traffic engineering could cause the temporary quality degradation of network services due to the dynamic change of routing paths. Therefore, the traffic engineering may not be necessary for clients requiring mission-critical communications such as crucial meeting and remote surgery. In order to resolve this problem, the author in [5] has proposed a traffic engineering method that aims to dynamically determine a routing path for each service according to the service priority, assuming environments where multiple services are deployed. This traffic engineering method changes only the routing paths for general services. Meanwhile, it uses the fixed routing paths for important services such as remote surgery, treating them as Very Important Packet (VIP) traffic.

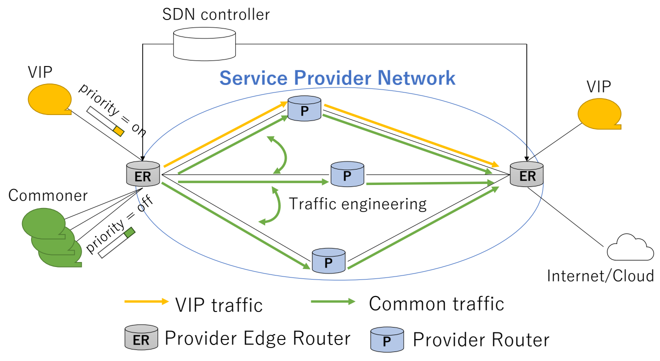

Fig. 1. System model.

This paper examines the implementation of the traffic engineering method including a service priority control mechanism based on [5]. In [5], the concept and implementability of the traffic engineering method using Programming Protocol-independent Packet Processors (P4) [4], which enables us to describe the behavior of data planes in SDN environments, has been just discussed. However, it has not provided ways of the practical implementation. Therefore, in this paper, we discuss the practical implementation of the traffic engineering method according to service priority using P4. Through demonstration experiments, we verify the behavior of the implementation.

II. Traffic engineering considering service priority [5]

Before explaining the practical implementation, we here briefly discuss the traffic engineering method considering service priority, which has been proposed in [5]. Fig. 1 shows the system model assumed in the traffic engineering method. There exists a service provider network which consists of provider edge routers and provider routers. The provider edge routers are controlled by an SDN controller. VIP and commoner users connect to the service provider network and inject VIP and common traffic, respectively, into the network. We assume that the VIP traffic has the priority over the common traffic. The header in each packet of VIP and common traffic has the information on the priority identification. The provider edge router performs traffic engineering to forward the packet to an appropriate output port based on the information. Specifically, the VIP traffic uses the fixed

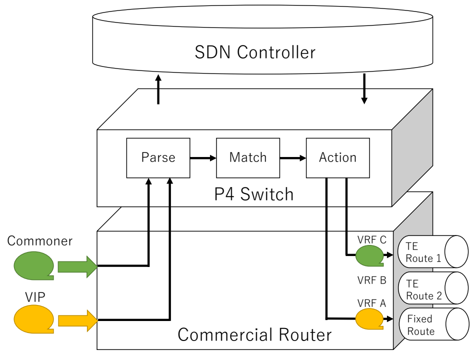

Fig. 2. Edge router construction.

```
header ipv4_t {
    bit<4>    version;
    bit<4>    ihl;
    bit<6>    diffserv;
    bit<2>    priority;
    bit<16>   totalLen;
    bit<16>   identification;
    bit<3>    flags;
    bit<13>   fragOffset;
    bit<8>    ttl;
    bit<8>    protocol;
    bit<16>   hdrChecksum;
    ip4Addr_t srcAddr;
    ip4Addr_t dstAddr;
}
```

Fig. 3. Definition of the IP header.

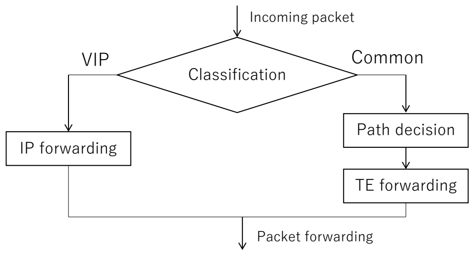

Fig. 4. Procedure of the implementation.

routing path while the common traffic is distributed to routing paths that are optimized by the traffic engineering method. The intermediate provide routers perform the general IP routing process based on the header information of each packet, and as a result, the packet arrives at the egress edge router.

Fig. 2 illustrates the configuration of an edge router assumed in this paper. The edge router consists of a P4 switch and a commercial router. The commercial router forwards each arrival packet from input ports to the P4 switch. The header information of the arrival packet is extracted to determine the routing path based on the destination address and the priority of the packet. Then it transfers the packet to the commercial router. The commercial router forwards the packet to the output port for the routing path decided by the P4 switch. We assume that Virtual Routing and Forwarding (VRF) [1] is used for forwarding the packet to the corresponding routing path. As shown in Fig 2, in the router, we construct the same number of VRFs as routing paths needed for the traffic engineering. By allocating the VLAN ID corresponding to each VRF to the P4 switch, the commercial router can forward incoming packets to appropriate output ports. The match and action fields in the P4 switch are controlled by the SDN controller, which collects the traffic information, determines the optimal routing path, and performs the periodic path construction. The SDN controller determines the optimal routing path, i.e., the distribution ratio of common traffic, by solving the optimization problem based on Linear Programming (LP) discussed in [5]. The optimization problem minimizes the maximum link utilization, to distribute load on each path. In order to save the space on the paper, we omit the detailed explanation on the optimization problem.

III. P4-BASED IMPLEMENTATION

In this paper, we examine the practical implementation of the traffic engineering method. In this implementation, we focus only on the operation of the traffic engineering method in the P4 switch. We do not consider the operation of other technologies in the commercial router such as VRF. Furthermore, we do not consider the functions of an SDN controller such as the optimal routing path calculation. Therefore, in what follows, we explain how to realize the traffic engineering method using only the function of the P4 switch, assuming that a routing table is manually prepared in advance. Specifically, we first explain the header configuration for identifying the service priority and the operation of the traffic engineering based on the service priority. We then explain how to create the match-action table for performing the traffic engineering.

A. Header information

As shown in Fig. 2, in the P4 switch, the header of an incoming packet is first extracted to be analyzed in the parser. The input match-action table receives the extracted header field. It consists of parameters, reference keys (such as IP address and MAC address), actions (such as drop and forward). The match-action table processes the packet header based on the actions and reference keys, which can be changed by programming, and determines the output port. Before being forwarded to the output port, the packet header is further forwarded to the output match-action table, which can also be programmed to define actions. After the output match-action processing, the packet is finally forwarded to the output port.

P4 can define the header for each layer such as Ethernet, IP, and TCP. In this paper, we use the Type of Service (TOS) field in the IP header to distinguish VIP traffic and common traffic. Fig. 3 shows the definition of the IP header in the P4 program. The lowest two bits of the TOS field (i.e., bit$\langle 2 \rangle$ priority) are defined as the priority field, which indicates the priority identifier. Note that other six bits of the TOS field is defined as the diffserv field. The value of the priority field is set to 1 for VIP traffic and 0 for common traffic, which are determined by the application of sender hosts. The P4 switch refers to the priority field of the incoming packet and selects the routing

```
apply{
 if(hdr.ipv4.isValid()){
  if(hdr.ipv4.priority==1){
   ipv4_lpm.apply();
  }
  else if(hdr.ipv4.priority==0){
   compute_hashes(hdr.ipv4.srcAddr,
    hdr.ipv4.dstAddr,
     (bit<16>)hdr.ipv4.srcAddr,
     (bit<16>)hdr.ipv4.dstAddr);
   det_vid.apply();
   vid.apply();
  }
 }
}
```

Fig. 5. Traffic classification.

path according to the appropriate routing table. Specifically, VIP traffic uses a pre-configured fixed routing path while common traffic is distributed to routing paths according to the distribution ratio calculated by the optimization method.

B. Implementation of traffic engineering

Fig. 4 shows the procedure of the implementation at the P4 switch. A packet arriving at the P4 switch at the ingress edge router is first identified whether it is VIP traffic or common traffic depending on the value of the priority field to execute corresponding processes. Fig. 5 shows an example of the implementation of this function using P4. First, the value of the priority field in the IP header is checked. If the value is 1, the packet is judged as VIP traffic and normal IP routing using the IP forwarding table (ipv4_lpm.appply()) is applied to the packet. If the value is 0, the packet is judged as common traffic. Then, a hash value is calculated based on the combination (source IP address, destination IP address, source port number, destination port number) in the packet header. Depending on the hash value, the routing path number corresponding to the VLAN ID is determined using the path decision table (det_vid.apply()), and the actual forwarding process is executed using the TE forwarding table (vid.apply()).

The IP forwarding table (ipv4_lpm.apply() in Fig. 5) performs a normal process that performs routing based on the destination IP address, which is defined as shown in Fig. 6. On the other hand, the path decision table (det_vid.apply() in Fig. 5) is implemented as shown in Fig. 7. Here, common traffic flows are distributed according to the traffic distribution ratio given by solving an optimization problem based on LP discussed in [5]. In this optimization problem assuming that there exist M routing paths, the traffic distribution ratio R_m ($\sum_{m=1}^{M} R_m = 1$, $0 \leq R_m \leq 1$) is calculated for each routing path m ($m = 1, 2, \ldots, M$).

In our implementation, a hash table of size 100 is prepared for mapping the traffic distribution ratios to the hash table. Note that traffic distribution ratios are normalized to the integer values such that the sum of them is 100. For example, we assume the situation where $R_1 = 0.272$, $R_2 = 0.364$, and $R_3 = 0.364$ are calculated for $M = 3$. In this case, the normalized integer values are 27, 36, and 37 for R_1, R_2, and R_3, respectively. In this case, as shown in Fig. 7, if the hash value is between 0 and 26, the packet is forwarded to the first route using VLAN ID 1. If it is between 27 and 62, the packet is forwarded to the second route using VLAN ID 2. Also, the packet is forwarded to the third route using VLAN ID 3 if it is between 63 and 99. The selected route is recorded in the upper 6 bits of the TOS field (i.e., the diffserv field in Fig. 3).

```
action drop(){
 mark_to_drop(standard_metadata);
}

action ipv4_forward(macAddr_t dstAddr,
 egressSpec_t port){
  standard_metadata.egress_spec = port;
  hdr.ethernet.srcAddr
    = hdr.ethernet.dstAddr;
  hdr.ethernet.dstAddr = dstAddr;
  hdr.ipv4.ttl = hdr.ipv4.ttl - 1;
}

table ipv4_lpm{
 key = {
  hdr.ipv4.dstAddr: lpm;
 }
 actions = {
  ipv4_forward;
  drop;
  NoAction;
 }
 size = 1024;
 default_action = drop();
}
```

Fig. 6. IP forwarding table.

```
action set_vid(bit<6> val) {
 hdr.ipv4.diffserv = val;
}

table det_vid {
 key = { get_position : range; }
 actions = { set_vid; }
 const entries = {
  0  .. 26 : set_vid(1);
  27 .. 62 : set_vid(2);
  63 .. 99 : set_vid(3);
 }
}
```

Fig. 7. Path decision table.

As shown in Fig. 8, the TE forwarding table (vid.apply() in Fig. 5) determines the output port based on the route recorded in the diffserv field (hdr.ipv4.diffserv) to forward common traffic to the corresponding routing path. In this way, different routing tables are prepared for VIP traffic and common traffic, and routing for common traffic is performed based on the value of the diffserv field, instead of the IP address. Note that the routing tables are expected to be prepared by the SDN controller, but the routing table is created manually in this paper because this implementation focuses on checking the data plane operation, i.e., P4 switch. We leave the control plane implementation as future work.

IV. Demonstration Experiments

In this paper, we verify the operation of the traffic engineering method considering service priority implemented

```
action vid2port(macAddr_t dstAddr,
 egressSpec_t port) {
  standard_metadata.egress_spec = port;
  hdr.ethernet.srcAddr
    = hdr.ethernet.dstAddr;
  hdr.ethernet.dstAddr = dstAddr;
  hdr.ipv4.ttl = hdr.ipv4.ttl - 1;
}

table vid {
 key = {hdr.ipv4.diffserv : exact; }
 actions = {
  vid2port;
  drop;
  NoAction;
 }
}
```

Fig. 8. TE forwarding table.

TABLE I
RATIO OF TRAFFIC VOLUME.

Path	Hash value	Distribution ratio	Ave. measured traffic ratio
1	0-26	0.27	0.211
2	27-62	0.36	0.360
3	63-99	0.37	0.429

by P4 through demonstration experiments using Mininet [3]. Mininet is a network emulator that can virtually build an SDN environment by combining hosts, switches, and routers. We use iPerf [2] to generate packets. Fig. 9 shows the network implemented in the experiments. In the network, 10 hosts (h1 to h10) acting as senders are connected to switch s1, and 10 hosts (h11 to h20) acting as receivers are connected to switch s5. We assume that a common traffic flow is generated between each sender and receiver pair, i.e., 100 distinct traffic flows are totally generated. There are three routing paths. Common traffic flows are distributed to each routing path according to the distribution ratios. In the experiments, the distribution ratios of the routing paths are set to $R_1 = 0.272$, $R_2 = 0.364$, and $R_3 = 0.364$, assuming that there exists VIP traffic using the routing path 1.

Fig. 10 shows the total volume of common traffic flows on each routing path in the case where the common traffic flows of 0.1 [Mbps] are generated between 10 pairs of sender and receiver hosts every 11 seconds for 10 seconds (for a total of 110 seconds). Also, Table I shows the average ratio of the traffic volume measured over all the time intervals. As we can see from Fig. 10, common traffic flows are distributed to each routing path at each time interval. Note that the distribution ratio differs depending on the time intervals because there is a bias in the hash values. On the other hand, as shown in Table I, the ratio of the average traffic volume is roughly close to the distribution ratio. Because there are only 100 sender and receiver pairs (i.e., 100 traffic flows), it does not strictly correspond to the distribution ratio. It is expected that the ratio of average traffic volume approaches the distribution ratio as the number of flows increases.

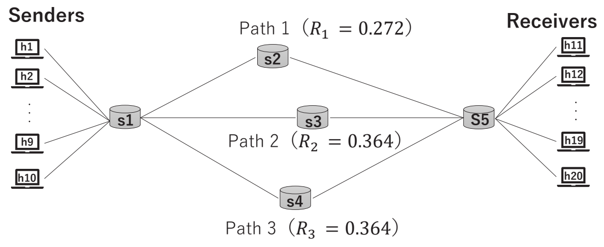

Fig. 9. Network environment built on Mininet.

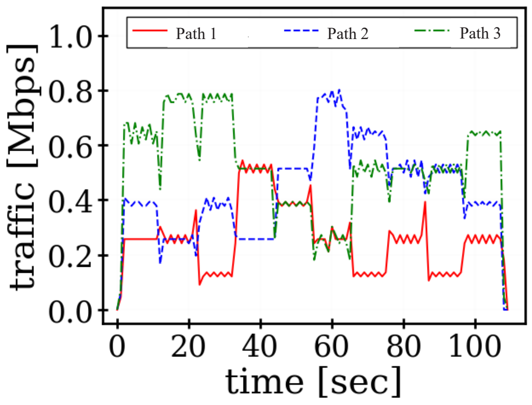

Fig. 10. Traffic volume on each path.

V. CONCLUSION

In this paper, we implemented the traffic engineering method considering the service priority, using P4. Through demonstration experiments, we confirmed the operation of the traffic engineering implementation. In this paper, we focus on the behavior of the data plane. As future work, we will examine the operation of the control plane such as the construction of routing tables.

Acknowledgement This research was partially supported by JSPS KAKENHI Grant No. 21K11857 and 24K14921.

REFERENCES

[1] RFC 7246, "Multipoint label distribution protocol in-band signaling in a virtual routing and forwarding (VRF) table context," 2014.
[2] iPerf, https://iperf.fr/
[3] Mininet, http://mininet.org/
[4] P. Bosshart et al., "P4: Programming Protocol-independent Packet Processors," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 3, pp. 88–95, 2014.
[5] S. Kamamura, "Dynamic traffic engineering considering service grade in integrated service network," *IEEE Access*, vol. 10, pp. 79021–79028, 2022.
[6] B. Nunes, M. Mendonca, X.-N. Nguyen, K. Obraczka, and T. Turletti, "A survey of software-defined networking: Past, present, future of programmable networks," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 3, pp. 1617–1634, 2014.
[7] N. Varyani, Z.-L. Zhang, and D. Dai, "QROUTE: an efficient Quality of Service (QoS) routing scheme for software-defined overlay networks," *IEEE Access*, vol. 8, pp. 104109–104126, 2020.
[8] Z. Xu et al., "Teal: learning-accelerated optimization of WAN traffic engineering," in *Proc. ACM SIGCOMM 2023 Conference*, Sep. 2023, pp. 378–393.