Data-driven Routing Optimization based on Programmable Data Plane

Qian Li, Jiao Zhang, Tao Huang, Yunjie Liu
State Key Laboratory of Networking and Switching Technology, BUPT, China
Email: {liqian06, jiaozhang, htao, liuyj}@bupt.edu.cn

***Abstract*—To meet the growing demand for high bandwidth of Multimedia network, IP Network Providers spend millions of dollars overprovisioning bandwidth of their network. However, due to the lack of reasonable traffic scheduling, the over-provisioning network still has a severe issue of utilization imbalance. Traffic Engineering (TE) is proposed to solve this problem. Network measurement and routing optimization strategies are two key components of TE. Existing out-band network telemetry that transmits extra probes to measure network status has the problem of inaccurate measurement information in the highly time-varying network. And the relationship between complex network status and routing optimization strategy is difficult to describe with an exact mathematical model. Therefore, we propose a novel TE approach, which is called DPRO. It combines In-band Network Telemetry based on programmable language P4 with Reinforcement Learning to minimize network congestion. Extensive experiments show that our approach significantly outperforms several widely-used baseline methods in terms of max-link-utilization.**

***Index Terms*—Traffic Engineering, P4, Reinforcement Learning, In-band Network Telemetry**

I. Introduction

The Internet has currently changed into a Multimedia network, which supports many types of multimedia applications with high bandwidth demand. Thanks to the rapid development of communication network hardware, IP network providers can adopt a strategy of bandwidth overprovisioning in their networks to support the sharply growing traffic of customers [1]. However, this approach is no longer applicable due to the increasing traffic demand, the high update cost, and the low link utilization. Traffic Engineering (TE) is proposed to solve this problem.

Network measurement and routing optimization strategies are two key components of traffic engineering (TE). Correspondingly, there are two major challenges to designing a good TE solution. One is that the current network measurement can not report network status accurately and timely. Some related works [2] [3] [4] aim to manage flow traffic by timely detection of significant flows or links that carry a large amount of data, with existing out-band network telemetry (ONT) methods. For example, NetFlow [5] tries to detect network status by periodic polling of traffic statistics from switches, and [6] achieves network measurement by sampling packets from switches. However, these ONT methods have high monitoring overheads, which easily incur significant switch resource consumption and long detection times. Another key challenge of TE is that the relationship between network state and routing optimization strategy is quite difficult to describe with an exact mathematical model [7]. Besides, overly complex mathematical models often place high demands on the storage and computing resources of the network nodes that run the model.

The development of Software Defined Networking (SDN) [8], has presented us with new chances for TE. Flexible and programmable data plane makes the network more operational and transparent. The novel network measurement, INT (In-band Network Telemetry) [9], allows the cost-effective network monitoring by encapsulating device-internal states into probe packets. With INT, we can get the real-time network status information with the finest granularity we desire, and that is the basis of good routing optimization strategies. Also, the recent breakthrough of reinforcement learning (RL) [10] provides a promising technique to enable effective experience-driven model-free control. It hits us that we could combine RL with a centralized control plane to make decisions.

Therefore, we propose an innovative and highly effective data-driven, model-free TE solution called DPRO in this paper. In DPRO, the data plane is responsible for timely monitoring network states with INT. Based on the network status uploaded by the data plane, the control plane performs route optimization under the guidance of a reinforcement learning algorithm.

On the flip side, leveraging INT information and reinforcement learning in TE is not straightforward. There are two main challenges to design DPRO. First, the P4-based INT specification [9] defines a reference format for packet encapsulation, including all metadata that INT can measure within the device. However, this encapsulation format in [9] takes up a lot of processing latency for the switch, and not all information is what DPRO needs. Therefore, we redesign the P4-based packet processing logic to combine INT with the processing of regular packets in DPRO. Second, the design of the model has a lot to do with performance in reinforcement learning. If the correlation between input and output in the reinforcement learning model is small, the RL model may be difficult to converge to satisfactory performance. Our approach simplifies the complexity of the reinforcement learning model by learning the mapping from link loads to link weights, speeding up the model's convergence. We summarize our contributions in the following:

- We are the first to propose a complete practical INT-based data-driven control framework, DPRO, for TE.

- We design a clever switch packet processing logic based on programmable language P4 to enable fast forwarding and ultra-low overhead measurement.
- We show via building an experimental platform that DPRO significantly outperforms several widely-used baseline methods concerning three metrics, including max-link-utilization, flow completion time and packet delay.

II. Background and Motivation

In this section, we present the background that is necessary to understand the main aspects of our approach, with a focus on the programmable data plane and reinforcement learning. Then, we discuss some prior works and our motivation.

A. Background

1) Programmable data plane: The emergency of data plane programmable language P4 [11] greatly solves the limitations imposed by the OpenFlow protocol. Unlike the OpenFlow protocol, P4 is a high-level language for programming protocol-independent packet processors, and it intends to be used for the description of package processing capabilities of programmable switches. Compared with OpenFlow which has fixed header fields, P4 can custom the header fields of the packet and the processing workflow. This new type of interaction with the forwarding devices can effectively bring programmability.

The P4 code is compiled specifically for a platform and requires the compiler to be compatible with the hardware architecture. An API is auto-generated and allows control plane to operate table rules or other objects.

2) Reinforcement learning: In the reinforcement learning framework, an agent repeatedly interacts with an environment in discrete decision epochs. At the beginning of each time epoch t, the agent observes the current state s_t, of the environment and selects an action a_t from a fixed set of actions. Once the agent makes a decision and chooses an action a_t, the state of environment changes into s_{t+1}, and the agent receives a reward r_t that signifying how good or bad the action that the agent took was. Therefore, the agent needs to try and try gradually, learn the environment and adjust its own decisions to achieve a greater reward. Finally, the agent learns a mapping π from the set of possible states $\widehat{S}$ to the space of actions $\widehat{A}$. A more detailed exposition of reinforcement learning can be found in [10].

B. Prior Work

1) Traffic management: TE optimization is a well-researched challenge. Recent interest in centralized TE is driven by SDN to running and optimizing such networks at scale (such as B4 [12], FFC [13], and Teavar [14]). These schemes exploit a global view of the network and perform global updates to configure flow allocations.

2) Network measurement: In-band network telemetry is a hot topic in recent years. [15] proposes INT-path, a network-wide telemetry framework, by decoupling the system into a routing mechanism and a routing path generation policy, transforming network troubleshooting into pattern recognition problems. HPCC [16] is a new high-speed congestion control mechanism, which leverages in-band network telemetry to obtain precise link load information and controls traffic precisely.

C. Motivation

In recent years, the outstanding performance of machine learning especially reinforcement learning in other fields (such as AlphaGo [17]), attracts the attention of network researchers. More and more attempts have been made to exploit machine learning techniques to solve network problems, such as TE. [18] is the first attempt to introduce reinforcement learning to the TE problem. It takes the traffic matrix of K-long historical time slices as input and the weight of all links as the output to minimize network congestion. DRL-TE [19] presents an experience-based DRL-based control framework for TE and aims to learn the proportion of traffic distribution of K TCP streams on multiple sub-paths.

These above studies all focus on the design of machine learning models, but the overall scalability and operability of these schemes are neglected. For example, [18] assumes that the flow demand matrix in each time slice is fixed and can be accurately calculated or well predicted by network measurement. But this assumption has always been a lot of controversy [19]. Otherwise, [19] has no consideration of scalability problems in the RL model. When the number of TCP connections in the network increases, the state space and action space of the reinforcement learning model will increase sharply. Until then, the learning rate and efficiency of the machine learning algorithm will decrease and the performance of DRL-TE will be difficult to guarantee.

Motivated by these approaches, we try to propose DPRO, a complete data-driven control framework for TE, based on fine-grained and real-time network measurement with INT. We describe the details of our approach in the following sections.

III. Design

DPRO addresses TE problems with the perspective of integration between the data plane and the control plane. Fine-grain network measurement in real-time is supported by the data plane, while route updates that require a global view of the network are performed by the control plane. As such, DPRO relies on an adaptive feedback control-loop: an offline control-loop that is periodically executed by the controller to analyze routes and make global changes, based on the network status uploaded by the data plane. An overview of DPRO is shown in Fig.1.

A. Design of Data Plane using P4

In our proposal, we program the processing logic of switches with programmable language P4 [11] to implement both forwarding and monitoring tasks.

1) Data plane routing mechanism: The flow concept is defined by the tuple: source IP, destination IP, protocol version (4 or 6), the source port, and the destination TCP port.

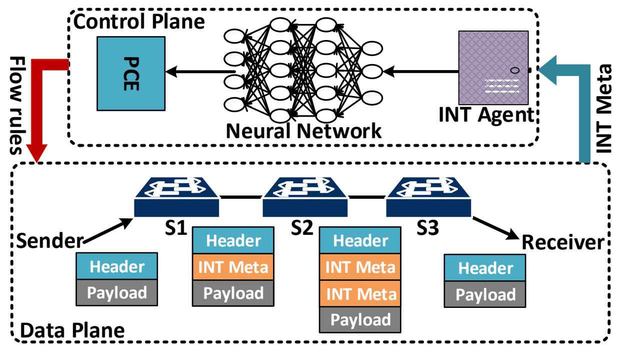


Fig. 1. The overview of DPRO

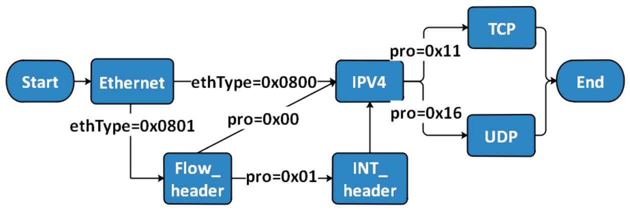


Fig. 2. Parser order

Our proposal is routed at the granularity of flow. Routing optimization with flow granularity has many advantages over granularity with packets or Flowlet [20]. To minimize the number of flow rules in the P4 switches and therefore, reduce the network reconfiguration time, we use a special flow header with the following format to guide forwarding:

```
header flow_header_t {
    count :8;      // number of hops
    routeid :16;   // identifier of the path
    protocol :8;   // 1:sample flag, 0:no sample
}
```

On the data plane, each switch can logically plays an intermediate or endpoint role. As for route tasks, all the decisions over the flows happen at the endpoints. The endpoint switch is responsible for switching flows between the routes installed by the controller as well as inserting the flow header (flow_header). For the intermediate nodes, they only need to perform the forwarding operation between the input port and the output port, thereby reducing processing and memory usage. Thus, intermediate switches do not need to perform any flow header modification.

2) Data plane measurement mechanism: Network measurement is a vital part of traffic engineering. Because the measurement results not only can provide feedback on the policy adjustments of the TE control subsystem but also can be an important tool for judging how good or bad the TE solution is. In this work, we try to use in-band network telemetry (INT) [9] technology to measure network states through inflight packets, which is more accurate and reliable as well as wasting little bandwidth overhead. We discuss the overhead of network measurement with INT in Section IV-B.

In-band network telemetry relies on the processing pipeline in the P4 switch. During the processing of a packet, P4 switch records all metadata information of the packet inside the switch, which contains queueing delay, input port, queue depth when leaving the queue and so on. Besides, a counter entity is used to record the forwarding bytes or packets of each port in the P4 switch. The statistics of the counter entity can directly be read or reset by the control plane through the counter interface at any time.

Based on the above features of P4 switches, we try to use INT based on P4 language to measure network status through inflight packets in DPRO, especially link load and switch node load. The *link load* is defined as the ratio between the amount of data being transmitted on the link in a sampling period and link capability. In DPRO, the centralized controller can read the counter statistics of ports at both ends of a link through the counter interface, thus, the load of the link in a period can be obtained after simple calculating. *Switch node load* has never been considered as metrics of network states in prior TE solutions, but it is also a very important parameter to indicate the network status. In the theoretical model of the communication network, the node cost is often averaged to all edges separated from the node, which can illustrate that switch node cost should also be considered when selecting the best path. Due to the lack of a unified definition of node load, we define the ratio between the queue depth and the limited length of the queue as the switch node load in DPRO.

As for measuring tasks, we can program the processing logic of switches to achieve periodic automatic sampling of packets. And the INT information is piggybacked on inflight packets. Specifically, only the endpoints are responsible for setting sampling flag and all switches perform inserting INT information with a format as follows when meeting sampling flag:

```
header int_t {
    swid:8;      // identifier of switch
    qdepth:16;   // the depth of queue when the packet was dequeued
}
```

3) Packet processing logic: When a new packet arrives at the edge switch, the packet header field will be parsed firstly in the defined order as depicted in Fig.2. P4 describes state machine that traverses packet headers from start to finish, directly as the set of transitions from one header to the next. Each transition may be triggered by values in the current. Here, we define that the *etherType* field in the Ethernet header is 0x801, indicating that the next header type is the route header (flow_header). The *protocol* field of the routing header is 0x01, indicating that the next header type is the measurement header (int_header). After the parser block, the extracted headers are forwarded to the control block for processing.

The complete logic of a packet being processed in the control block is shown in Alg.1. If the flow is new, the flow_header should be added to all packets of the flow by applying `create_flow_header` table. The `create_flow_header` table is indexed by IP/mask and is used to store the active route identifiers grouped by source-destination node pairs. To avoid the “herd” mentality, K equivalent paths are configured by the controllers between each pair of nodes. According to the target IP, the flow matches one route group, then chooses an appropriate path number according to the result of the five-tuple hash. Besides, `routeid_fwd` table is indexed by the route identifier and returns which port to be forwarded. As for sampling tasks, `check_register` table and `sample_flag` table are designed to set sample flag. Specifically, *protocol* field in flow_header is bound to be set to 1 when the time interval between the arrival of the current packet and the last sampling exceeds the sampling period T_s. `add_load_info` table is a no index table and used to add INT information as listed in int_t header to sampling packets.

Algorithm 1 Data Plane Packet Processing

```
1:  procedure CONTROL INGRESS
2:      if meta.routeid = 0 then
3:          apply create_flow_header
4:          apply check_register
5:          if (current_tstamp − last_sample_tstamp) > $T_s$ then
6:              apply set_sample
7:          end if
8:      end if
9:      apply routeid_fwd
10:     if eth.ethType = 0x0800 & flow_hdr.pro = 0x01 then
11:         apply clone_to_controller
12:     end if
13: end procedure
14:
15: procedure CONTROL EGRESS
16:     if flow_hdr.pro = 0x01 then
17:         apply add_load_info
18:     end if
19:     if std_meta.istance_type = 0 & eth.ethType = 0x0800 then
20:         apply remove_additional_header
21:     end if
22:     if std_meta.istance_type = 2 then
23:         apply update_clone_flag
24:     end if
25: end procedure
```

When the packet arrives at the destination switch and is bound to be delivered to the target host next, the packet should be reverted to the normal format for reception by the destination host, that is, the flow_header (that used to guide routing)and int_header (that used to carry INT information) should be removed (`remove_additional_header` is applied). Before that, the packet is bound to be replicated (`clone_to_controller` is applied) if there is some load information carried on the packet header. Note that the processing of normal packets and replicated packets are independent of each other. The normal packet will be sent to target hosts while the replicated packet will be delivered to a special port that connects to the controller.

B. Control Plane Operation

In DPRO, the centralized controller is responsible for delivering route optimization strategies with a global view of network status. However, because the regularity of network state changes is difficult to figure out, we try to use reinforcement learning techniques with self-exploration characteristics to perceive network changes and thus, assist in the generation of route optimization strategies.

1) Design of reinforcement learning model: To utilize the RL techniques (no matter which method/model to use), we first need to design the state space, action space, and reward function. Note that the design of state space, action space and reward is critical to the success of an RL method.

- *State Space*: The state space consists of two components: link load and switch node load. Formally, the state vector is defined as S, where l_i is the link load of link i and s_j is the node load of switch j.

$$S = (l_1, l_2, ..., l_i, ..., l_n, s_1, s_2, ..., s_j, ..., s_m)$$

- *Action Space*: Considering that the direct output routing strategy or traffic distribution ratio will make the action space too large, affecting the performance of the RL model, thus we decide to regard the weight values of all edges in the network as the action space. Formally, the action vector is defined as A, where w_i is the weight value of link i and $|w_i| <= 2$.

$$A = (w_1, w_2, ..., w_i, ..., w_n)$$

- *Reward*: Minimizing congestion is the most basic optimization goal of TE. Because in an IP network, congestion can affect latency, jitter and even packet loss. Minimizing congestion can also indirectly reduce flow completion time and increase overall network throughput. Therefore, our optimization goal is to minimize congestion (a.k.a. minimize max-link-utilization). Formally, the reward is defined as R, where u_i is the link utilization of link i.

$$R = -max(u_1, u_2, ..., u_i, ...u_n)$$

In DPRO, time is divided into consecutive intervals, called “epochs”. At the beginning of each epoch t, the controller (agent) decides on a routing strategy $R^{(t)}$ for that epoch based on the routing strategies and the network load status, which constitutes the observed state $S^{(t)}$ of the environment at that point. Then, the state changes as the network load status for epoch t, $S^{(t+1)}$, is revealed and the reward $R^{(t)} = -u(t)$ is received, where $u(t)$ is the max-link-utilization under $R^{(t)}$ for $S^{(t+1)}$. Our goal is to learn a mapping from network load status to the weights of links that maximize the expected discounted reward, as formulated above.

The TE problem is a continuous control problem. PPO algorithm is a state-of-the-art deep reinforcement learning algorithm PPO [21] for continuous control and its basic idea has been introduced in Section II-A. Thus, we choose PPO [21] algorithm for our approach.

2) Routing update mechanism: Based on the weights of network links which are the output of PPO algorithm, the Dijkstra algorithm is used to calculate the shortest path between all pairs of source-destination nodes. Then, the optimal path is bound to replace the current worse path between a pair of nodes. Traffic engineering is generally an online strategy, and the traffic status changes quite violently. However, the excessively large-scale adjustment of the network routing policy is likely to cause the instability of the route, inducing route oscillating and packet loss. To solve this problem, we use the polling method to update the routing entries of all pairs of nodes. That is, the routing policy of only one pair of source-destination nodes will be updated within an update period T_u.

IV. Evaluation

In this section, we evaluate the performance of the proposed DPRO. Section IV-A describes the topology and workloads used as well as the metrics we considered in our evaluation, and Section IV-B presents the evaluation results.

A. Experimental Settings

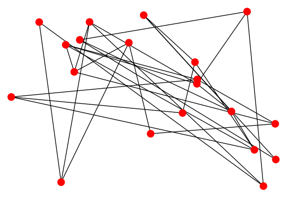
We implemented our approach with the bmv2[1] software switch simulator and a customized controller using Python language. The veth-pair was used to simulate links. All evaluations were performed in the same setup with identical configuration, operation system Ubuntu Linux virtualized with QEMU/KVM, with 128GBytes of RAM and access to the host processor, Intel(R) Xeon(R) CPU E5-2609. The Maximum Transmission Unit (MTU) was set to 1530 bytes for all switch NICs. The bmv2 simulator was used to interpret the P4 code. The capacity of each link was set to 100Mbps.

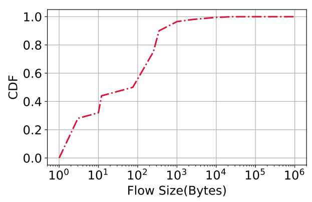
1) Network topology and workload: We model the network as a capacitated directed graph $G = (V, E, c)$, where V and E are the vertex and edge sets, respectively, and c assigns capacity to each edge. Let m denote the number of vertices in V and n donate the number of edges in E. To prove the generality of DPRO, we randomly generated a network topology with 20 nodes and 31 links as our experimental topology (as shown in Fig.3(a)). That is, $m = 20$ and $n = 31$. Besides, the flow size distribution that we use to evaluate is shown in Fig.3(b), which is a realistic workload released by Facebook. And the traffic generator that we use is an open-source traffic generator released by [22], which is commonly used for evaluations.

2) Metrics and compared methods: Although the focus of DPRO is on the conventional optimization objective of minimizing link over-utilization (a.k.a. minimizing congestion) from traffic engineering literature, it can also influence the changes of other metrics such as flow completion time and packet delay to some extent. Therefore, we choose link max-link-utilization and flow completion time as well as packet delay as performance metrics to evaluate DPRO. Also, we compared our approach with three widely used baseline solutions:

- Shortest Path (SP): each source-destination pair uses the shortest path to deliver all its packets.
- Equal Cost Multi-Path (ECMP): each source-destination pair evenly distributes its traffic load to K candidate paths.
- Dynamic Routing (LINEAR) [23]: it obtains TE solutions by modeling link loads and the cost (which is the same as link weights in DPRO) to a piecewise linear function. The cost of the link l has to do with the relation between $l(a)$ and $c(a)$, as shown in function Φ_a.

$$\Phi_a(x) = \begin{cases} 1 & \text{if } x \in [0, \frac{1}{3}c(a)), \\ 3 & \text{if } x \in [\frac{1}{3}c(a), \frac{2}{3}c(a)), \\ 10 & \text{if } x \in [\frac{2}{3}c(a), \frac{5}{6}c(a)), \\ 100 & \text{if } x \in [\frac{5}{6}c(a), \infty). \end{cases}$$


(a) Network topology



(b) Flow size distributions

Fig. 3. Network topology and Workload

B. Performance Results

To better understand experiments, only 2 edge nodes are selected to act as clients and 6 servers are chosen to reply to clients' requests. Each source node creates 6 TCP connections and sends the flow requests with flow sizes in polling. And each server replies with required flows to clients. Otherwise, we set the update period as 5 seconds and the sample period as 1 second, that is, $T_u = 5s$ and $T_s = 1s$. Each source-destination pair has two optimal paths, that is, $K = 2$.

1) Online training results: We show the performance of DPRO during the online learning procedure in terms of the reward. For illustration and comparison purposes, we normalize and smooth the reward values using a commonly-used method $(r - r_{min})/(r_{max} - r_{min})$ (where r is the actual reward, r_{min} and r_{max} are the minimum and maximum rewards during online learning respectively) in Fig.4. Compared to ECMP, which we regard as a baseline, DPRO quickly (within just a couple of hundreds of decision epochs) reaches a good solution (that gives a high reward), while ECMP seems to have no awareness of traffic load and perform stable and low reward values in all the training epochs. As expected, we can see from Fig.4 that DPRO significantly outperforms ECMP in terms of the max-link-utilization because its reward function is set to minimizing it.

2) Performance of max-link-utilization: To test the performance of all four methods subject to the max-link-utilization

[1]https://github.com/p4lang/behavioral-model

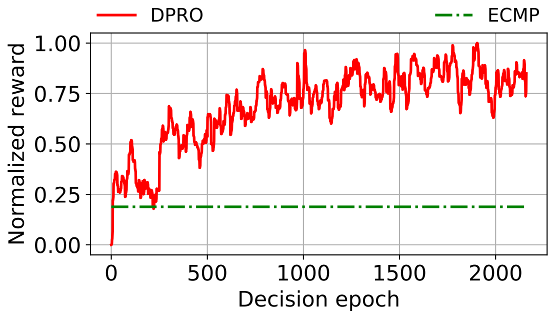
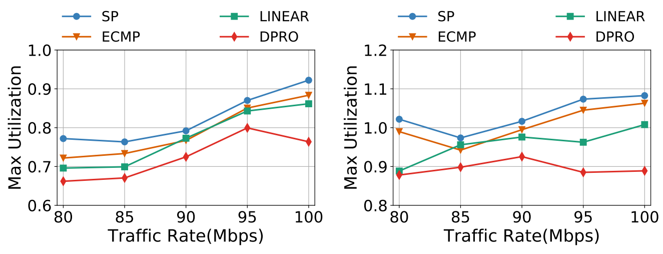


Fig. 4. Reward during online learning



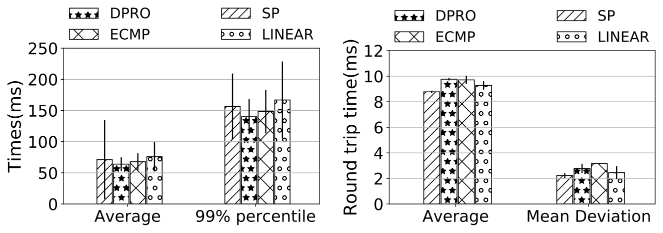
(a) Without background traffic (b) With background traffic

Fig. 5. Max utilization of all the methods over different traffic loads

under different traffic loads, we set the traffic rate to 80Mbps initially and increase the traffic rate with a step size of 5 Mbps for each run. The traffic pattern is the same as the training period. Besides, we also add an evaluation under training traffic pattern with background traffic to illustrate the generality of DPRO. The background traffic is injected into our network at a rate of 200 packets per second through three edge nodes. We present the corresponding experiment results in Fig.5. Note that the numbers on the x-axis are the values of the corresponding traffic rate and the numbers on the y-axis are the max ratios between the link load and the link capacity of four TE methods.

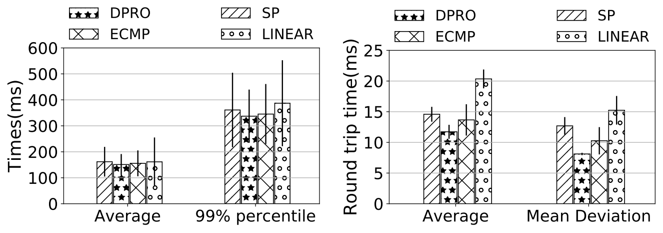
When the traffic load is without background traffic, as shown in Fig.5(a), DPRO always keeps the max-link-utilization below 0.8, while all the other three methods even up to 0.85 with the traffic load increases. Overall, DPRO achieves an average reduction of 13.9%, 9.3% and 6.9% respectively, compared to SP, ECMP, and LINEAR. When the traffic load is with background traffic, as shown in Fig.5(b), DPRO significantly maintains the max-link-utilization at about 0.9, with a floating range at 0.02. All the other three methods lead to poor performance with max-link-utilization up to 1.0, which can easily cause network congestion. On average, DPRO outperforms SP, ECMP, and LINEAR by 15.5%, 12.6% and 7.1% respectively.

3) Effective of FCT and packet delay: Network congestion affects flow completion time and packet delay to some extent. Fig.6 shows the FCT and packet delay under such a traffic pattern as the online learning procedure. By using DPRO, the



(a) FCT (b) Packet delay

Fig. 6. FCT and packet delay without background traffic



(a) FCT (b) Packet delay

Fig. 7. FCT and packet delay with background traffic

average FCT (as shown in Fig.6(a)) is reduced by 11.4%, 6.1% and 18.8% compared with SP, ECMP, and LINEAR. And the 99th percentile FCT is cut down from $156.7ms$ to $140ms$, more than 19.2% improvement that LINEAR. Note that as for packet delay (as shown in Fig.6(b)), these four TE approaches have similar delay performance. Especially, SP has the smallest average packet delay than the other three approaches. This is because that the training traffic load is quite light and the overall network link utilization is low, and only one single path is chosen to forward packets between a pair of nodes in SP while the other three methods have two, thus the advantages of multipath have no chances to show.

Fig.7 shows the FCT and packet delay respectively under training traffic patterns with background traffic. DPRO has much better performance than the other three methods because it can adjust routing strategy at the basis of network load status monitoring in real-time. Compared with SP, as shown in Fig.7(a), DPRO reduces the average FCT by 7.2% and the 99th percentile FCT by 7.1%. Especially, in terms of packet delay, as shown in Fig.7(b), DPRO outperforms SP, ECMP, and LINEAR by 56.8%, 27.2%, and 75.3% respectively in mean deviation, and 24.8%, 17.1%, and 53.8% in average packet delay.

4) Measurement overhead: The measurement overhead via INT in our approach is very low, especially on network bandwidth. In our proposal, we set the size of one int_header is 24 bit and the maximum hops of one route is 8. Thus, the maximum size of measurement information carried by a sampling packet is 24 bytes.

From the view of the only packet, the Ethernet header is 18 bytes, and the IP header, as well as TCP header, is generally 20 bytes. Thus, the largest proportion of int_headers in the headers of one packet is $24/(24+18+20*2) = 29.3\%$.

As we know, the maximum length of Ethernet packet is 1518 bytes and the minimum length is 64 bytes, therefore, we can calculate that the largest proportion of measurement information is $24/(24+64) = 27.3\%$ in minimum Ethernet packet and $24/(24+1518) = 1.56\%$ in maximum Ethernet packet.

From the perspective of bandwidth resources, DPRO has an extra-low requirement to bandwidth resources. As illustrated before, the sampling period is set to 1 second and only 14 edge switches are responsible for taking sampling tasks (based on the simulation topology). Therefore, the bandwidth occupied by network measurement is $3Bps * 14 = 42Bps$, which is only 0.000336% of the link capacity (the link capacity is set to 100MBps).

V. Conclusion

In this paper, we investigated the traffic engineering problem in SDN, for which we aimed to find the policy that minimizes network congestion with network load status changes. We explored in-band network telemetry technology based on programmable language P4 to measure network load status with ultra-low overhead. We introduced a reinforcement learning PPO algorithm to the controller to solve the problem that the relationship between network load status and routing strategies is hard to model and calculate. We named our approach DPRO. Evaluation results showed that DPRO offers significant performance improvement compared to three baseline TE solutions concerning three metrics including max-link-utilization, flow completion time, and packet delay.

References

[1] N. Wang, K. H. Ho, G. Pavlou, and M. Howarth, "An overview of routing optimization for internet traffic engineering," *IEEE Communications Surveys & Tutorials*, vol. 10, no. 1, pp. 36–56, 2008.

[2] M. Alizadeh, T. Edsall, S. Dharmapurikar, R. Vaidyanathan, K. Chu, A. Fingerhut, F. Matus, R. Pan, N. Yadav, G. Varghese *et al.*, "Conga: Distributed congestion-aware load balancing for datacenters," in *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 4. ACM, 2014, pp. 503–514.

[3] I. F. Akyildiz, A. Lee, P. Wang, M. Luo, and W. Chou, "A roadmap for traffic engineering in sdn-openflow networks," *Computer Networks*, vol. 71, pp. 1–30, 2014.

[4] N. Katta, M. Hira, C. Kim, A. Sivaraman, and J. Rexford, "Hula: Scalable load balancing using programmable data planes," in *Proceedings of the Symposium on SDN Research*. ACM, 2016, p. 10.

[5] C. Estan, K. Keys, D. Moore, and G. Varghese, "Building a better netflow," *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 4, pp. 245–256, 2004.

[6] M. Wang, B. Li, and Z. Li, "sflow: Towards resource-efficient and agile service federation in service overlay networks," in *24th International Conference on Distributed Computing Systems, 2004. Proceedings.* IEEE, 2004, pp. 628–635.

[7] Y. Azar, E. Cohen, A. Fiat, H. Kaplan, and H. Racke, "Optimal oblivious routing in polynomial time," in *Acm Symposium on Theory of Computing*, 2003.

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

[9] C. Kim, P. Bhide, E. Doe, H. Holbrook, A. Ghanwani, D. Daly, M. Hira, and B. Davie, "In-band network telemetry (int)," *P4 consortium*, 2015.

[10] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *Journal of artificial intelligence research*, vol. 4, pp. 237–285, 1996.

[11] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese *et al.*, "P4: Programming protocol-independent packet processors," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 3, pp. 87–95, 2014.

[12] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu *et al.*, "B4: Experience with a globally-deployed software defined wan," in *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 4. ACM, 2013, pp. 3–14.

[13] H. H. Liu, S. Kandula, R. Mahajan, M. Zhang, and D. Gelernter, "Traffic engineering with forward fault correction," in *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 4. ACM, 2014, pp. 527–538.

[14] J. Bogle, N. Bhatia, M. Ghobadi, I. Menache, N. Bjørner, A. Valadarsky, and M. Schapira, "Teavar: striking the right utilization-availability balance in wan traffic engineering," in *Proceedings of the ACM Special Interest Group on Data Communication*. ACM, 2019, pp. 29–43.

[15] T. Pan, E. Song, Z. Bian, X. Lin, X. Peng, J. Zhang, T. Huang, B. Liu, and Y. Liu, "Int-path: Towards optimal path planning for in-band network-wide telemetry," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE, 2019, pp. 487–495.

[16] Y. Li, R. Miao, H. H. Liu, Y. Zhuang, F. Feng, L. Tang, Z. Cao, M. Zhang, F. Kelly, M. Alizadeh *et al.*, "Hpcc: high precision congestion control," in *Proceedings of the ACM Special Interest Group on Data Communication*. ACM, 2019, pp. 44–58.

[17] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, "Mastering the game of go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, p. 484, 2016.

[18] A. Valadarsky, M. Schapira, D. Shahaf, and A. Tamar, "Learning to route," in *Proceedings of the 16th ACM Workshop on Hot Topics in Networks*. ACM, 2017, pp. 185–191.

[19] Z. Xu, J. Tang, J. Meng, W. Zhang, Y. Wang, C. H. Liu, and D. Yang, "Experience-driven networking: A deep reinforcement learning based approach," in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*. IEEE, 2018, pp. 1871–1879.

[20] S. Kandula, D. Katabi, S. Shan, and A. Berger, "Flare: Responsive load balancing without packet reordering," 2007.

[21] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[22] B. Li, K. Tan, L. Luo, Y. Peng, R. Luo, N. Xu, Y. Xiong, and C. Peng, "Clicknp:highly flexible and high-performance network processing with reconfigurable hardware," in *Conference on Acm Sigcomm Conference*, 2016.

[23] B. Fortz and M. Thorup, "Optimizing ospf/is-is weights in a changing world," *IEEE journal on selected areas in communications*, vol. 20, no. 4, pp. 756–767, 2002.