IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received March 27, 2020, accepted April 5, 2020, date of publication April 20, 2020, date of current version May 4, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2988652

Flowlet-Based Stateful Multipath Forwarding in Heterogeneous Internet of Things

JINYU SHI, WEI QUAN, (Member, IEEE), DEYUN GAO, (Senior Member, IEEE), MINGYUAN LIU, GANG LIU, (Graduate Student Member), CHENGXIAO YU, (Graduate Student Member), AND WEI SU

School of Electronic and Information Engineering, Beijing Jiaotong University, Beijing 100044, China

Corresponding author: Wei Quan (weiquan@bjtu.edu.cn)

This work was supported by the Fundamental Research Funds for the Central Universities of China under Grant 2019JBM011.

ABSTRACT Concurrent multipath transfer effectively improves network throughput and reliability by utilizing the unoccupied network links. However, the performance of concurrent multipath transfer degrades significantly in the heterogeneous Internet of Things. The reason is that the disorder and retransmission will increase sharply if the quality of links are greatly different. This paper proposes a multipath stateful forwarding mechanism in the granularity of flowlet, which is featured with planning the flowlet forwarding dynamically to improve the transmission success rate. Besides, our mechanism implements stateful forwarding by improving the state information processing capability of the data plane. It can adjust the size of the flowlet based on the network performance of the path, thereby improving the bandwidth resource utilization of the system. In particular, the proposed mechanism can sense network congestion status dynamically, detect network anomalies, and re-plan the forwarding strategy. Experimental results show that in the harsh environment (link quality varies greatly among the three test paths), the proposed mechanism can improve the throughput by 114.7% and reduce the percentage of out of order packets by 29.5% compared with the Round-Robin Scheduling mechanism.

INDEX TERMS Concurrent multipath transfer, stateful forwarding, flowlet-based forwarding, IoT.

I. INTRODUCTION

The Internet of Things (IoT) is an essential part of the new generation of Internet technology, which is considered to be the third wave of the development of the information industry [1]–[5]. The development of IoT technologies is based on the realization of the Internet of Everything, so, the core of the Internet of Things is still the Internet. However, IoT inevitably brings the complexity of access devices, which poses considerable challenges to traditional Internet technologies. At the same time, users are increasingly demanding network bandwidth and mobility. It is difficult for one single access method to meet network demands of users. With the diversified development of multiple access technologies, plenty of terminals begin to support multiple network access collaborations, such as cellular links (5G, 4G, 3G), WiFi, satellite links and Ethernet [6]–[8]. Therefore, how to reasonably utilize the multiple access methods provided by terminal equipment to aggregate network resources, and how to deliver higher network transmission performance to meet users' network needs is receiving widespread attention [9]. Under this background, the concurrent multipath transfer (CMT), as an access technology that can effectively improve data throughput, network resource utilization, and system robustness, has become one of the most popular research hotspots.

CMT is a technique for transmitting packets in more than one path. It can improve the network performance of the system by aggregating network resources. At the same time, it can improve the reliability of the system by transferring data from the most congested and interruptible path to a better path automatically. Thanks to the joint efforts of scientific researchers, CMT has achieved great progress. However, in a heterogeneous wireless network environment, the network performance in the scenario of multipath transmission is inferior to that in the scenario of single-path transmission. A large number of experiments [10], [11] have proved that the throughput of the multipath parallel transmission system will be lower than expected in a heterogeneous network environment because the huge difference between paths

The associate editor coordinating the review of this manuscript and approving it for publication was Nan Cheng.

will cause unnecessary retransmission and lots of disordered packets.

Currently, the forwarding algorithms for CMT can be summarized as follows:

1) Round-Robin Scheduling (RRS) [12]: RRS is a kind of stateless algorithm, in which packets are forwarded from different paths by polling. The advantage is the simple implementation, and the disadvantage is that it cannot obtain optimal network performance and cope with the problem of path interruption.
2) Weighted Round-Robin Scheduling (WRRS) [13]: WRRS algorithm assigns different weights to the path. The higher the weight, the more chances that the path will forward packets. Generally, the algorithm can achieve higher network throughput but still cannot solve the problem of path interruption.
3) Hashing Scheduling (HS) [14]: HS will use the hash function to calculate the five-element tuple information carried in the message to determine the forwarding path. The data of the same stream will be transmitted from the same path, thereby reducing the disorder of the data packets. The path interruption, however still cannot be handled. Besides, the advantages of concurrent multipath transfer cannot be fully utilized, when there is only one flow in the network, or a severe hash conflict occurs.

Based on the above research, and considering the emergence of new network technologies and programmable data plane technology, we propose a flowlet-based stateful multipath forwarding mechanism (FSMF) to improve the network transmission performance. In our mechanism, when the subpath network is heavily congested, the forwarding mechanism can adjust dynamically to reduce the overall impact on the system. Besides, it is compatible with traditional network architecture and can be deployed quickly. Our experimental results verify that the proposed mechanism is efficient, robust, and scalable.

The contributions of this paper are summarized as follows:

- *F*irst, we design a "match + state + action" programmable data plane with the state forwarding processing paradigm, which endows the data plane with the ability to maintain and process state information. As a result, the data plane supports state forwarding and then realizes more complex processing logic.
- *S*econd, we model the process of flow segmentation based on the bandwidth characteristics of the paths and obtain a reasonable approximate relationship between the size of the flowlets and the bandwidth of paths through the model, to ensure that the throughput of the system can reach the ideal state.
- *T*hird, we design a dynamic forwarding mechanism based on congestion state awareness, which makes the system have a better performance when dealing with network link interruption.

II. RELATED WORK

Over the last recent decades, plenty of research efforts have been dedicated to concurrent multipath transfer. The current mainstream concurrent multipath transfer strategies are mainly implemented at the transport layer and network layer of the Internet protocol stack. The most typical CMT mechanism at the network layer is equal-cost multipath routing (ECMP) [15], which hashes the IP address and port information of the packets and implements the flow granularity load balancing mechanism between the paths that have the same cost. However, most paths of the network are not equal-cost in a real network environment. And, the load balancing mechanism that evenly distributes traffic does not take differences in path bandwidth into account. Therefore, ECMP is not suitable for working in heterogeneous network scenarios with significant performance differences.

Zhou *et al.* [16] proposed the weighted cost multipathing (WCMP), which mainly to improve the problems in ECMP. The WCMP performs weighted load balancing based on network characteristics of different paths. Compared with ECMP, WCMP can make fuller use of redundant network resources, but it is still a flow-based coarse-grained load balancing mechanism. When there are fewer flows in the network, network resources will not be fully utilized. Besides, the difference in the duration of different flows will affect the effect of load balancing.

Since there is more network information in the transport layer, such as transmission delay, packet loss, and available bandwidth, it is more advantageous to implement CMT mechanism at the transport layer. In 2000, Internet Engineering Task Force (IETF) proposed a multipath transport protocol–stream control transmission protocol (SCTP) [17]. In a concurrent multipath transfer scenario, SCTP can provide multiple flows between two communicating parties, thereby alleviating the problem of stack blocking in TCP and improving network performance. However, owing to the late emergence of this protocol and its poor compatibility with existing protocols, it cannot be deployed on a large scale. At the same time, SCTP is designed for wired networks and is less efficient in wireless network environments.

In 2013, Internet Engineering Task Force (IETF) formally proposed the Multipath TCP (MPTCP) [18]. It is developed based on the traditional TCP protocol. By creating a sub-flow, it can carry out joint congestion control and data allocation according to the network characteristics of the sub-flow, allocating resources more equitably and improving the overall network performance significantly. Notably, since MPTCP relies on RTT to adjust the timeout retransmission time, it can discover the link connectivity in time. However, the congestion control algorithm and congestion window adjustment algorithm of MPTCP are not scientific enough, and MPTCP works poorly in a heterogeneous network environment. Besides, although MPTCP protocol is compatible with the current network architecture, it can work generally only when both sides of communication support the protocol at the same time. The emergence of MPTCP fundamentally changed the scheduling and transmission strategies of multipath transmission. After that, the researchers continued to propose a lot of improvements for MPTCP [19]–[22].

In addition to the research of the CMT mechanism itself, the emergence of the stateful forwarding plane also brings new inspiration to the concurrent multipath transfer. Specifically, the seminal paper by Bianchi *et al.* [23] sketched out a stateful forwarding plane abstraction (versus the stateless OpenFlow [24] match/action table). After that, Zhu *et al.* [25] proposed a novel Stateful Data Plane Architecture (SDPA) for the SDN data plane. Now, the stateful forwarding technology is fully considered as a new forwarding paradigm, which can improve processing efficiency in specific scenarios [26]. For example, Liu *et al.* [27] proposed a method for mitigating DDOS attacks based on a stateful forwarding plane. Arashloo *et al.* [28] proposed the SNAP, which offers a simpler centralized stateful programming model instead of a central controller, to improve the efficiency of the data plane significantly. Hu *et al.* [29] introduced the FLOWGUARD, which is considered as a comprehensive framework to build stateful firewalls for OpenFlow-based networks. Besides, the researchers at home and abroad have proposed a lot of stateful forwarding strategies to improve the delivery efficiency of the data plane [30]–[33].

However, processing state information in the data plane has always been a complicated task. Fortunately, Bosshart *et al.* [34] proposed the programming protocol-independent packet processors in 2014. The emergence of P4 enhances the flexibility and programmability of the data plane, making the network and devices truly open to users. Besides, the P4 Language Consortium provides a P4 specific switch simulator—BMV2 (behavior model version 2), which is not a production-level software switch. It is intended to be a tool for developing, testing, and debugging P4 data planes. In terms of throughput and latency, the performance of BMV2 is much lower than production-grade software switches such as Open vSwitch. Since BMV2 depends on the operating platform, its performance is also very inconsistent. From the perspective of development testing and solution verification, however, BMV2 is an excellent switch simulation engine. The emergence of programmable data planes provides powerful tools for stateful forwarding and cross-layer information processing [35].

Thanks to these efforts, it has brought some progress to achieve an efficient multipath forwarding mechanism. However, due to the complexity of heterogeneous IoT, we still face many challenges. This paper proposes a stateful dynamic forwarding mechanism, which can improve network performance and robustness.

III. SYSTEM MODEL AND ALGORITHM

Figure 1 shows the system block diagram of our mechanism. The entire system mainly consists of three parts: hash block, forwarding block, and state processing block. The specific functions of each block are as follows:

1) Hash Block: The hash block is responsible for processing the five-element tuple information of packets, grouping all packets received by BMV2 in the unit of flow, and triggering the status block to update the status table.
2) Status Block: The status block is mainly responsible for maintaining and updating the status tables according to the logic of the hash block and the forwarding block.
3) Forwarding Block: The forwarding block determines the forwarding path according to the forwarding mechanism and status information and divides the flow into flowlet of different sizes according to the weight of the path in the status block.

It should be noted that the flowlet is a sub-flow which consists of several consecutive packets from the same TCP flow. Next, we will detail the implementation of each module.

A. HASH BLOCK DESIGN

According to the granularity of forwarding, we usually divide the CMT mechanism into packet-based forwarding, flow-based forwarding, and flowlet-based forwarding. Specifically:

1) Packet-based Forwarding: It uses packets as the smallest forwarding unit for scheduling. The advantage of this mechanism is that it can simply and efficiently distribute traffic among multiple paths [36]. However, the problem of out of order packets is more prominent, which usually leads to the degradation of system performance.
2) Flow-based Forwarding: flow-based forwarding mechanism will forward packets in the same flow from the same path [37]. It solves the out-of-order problem well, but it is challenging to balance the load of each path. Specifically, when the hash algorithm design is unscientific, it will reduce the forwarding efficiency of the system.
3) Flowlet-based Forwarding: this mechanism artificially divides flow into flowlets of different sizes [38]. On the one hand, compared to packet-based forwarding, the out-of-order problem is alleviated in this mechanism. On the other hand, compared to flow-based forwarding, it has a better load balancing effect on packets. Even if there is only one flow in the system, load balancing can still be performed.

In heterogeneous IoT networks, due to the complexity of access devices and the differences in network performance of paths, it is difficult to obtain excellent network performance based on the packet-based or flow-based forwarding mechanism. So, we design a flowlet granularity multipath forwarding mechanism to achieve the balance between the lower percentage of out of order packets and sufficient network resource utilization.

The main function of the hash block is getting the hash value of the five-tuple in the parsed data packet. According to the hash value, different streams are distinguished, and

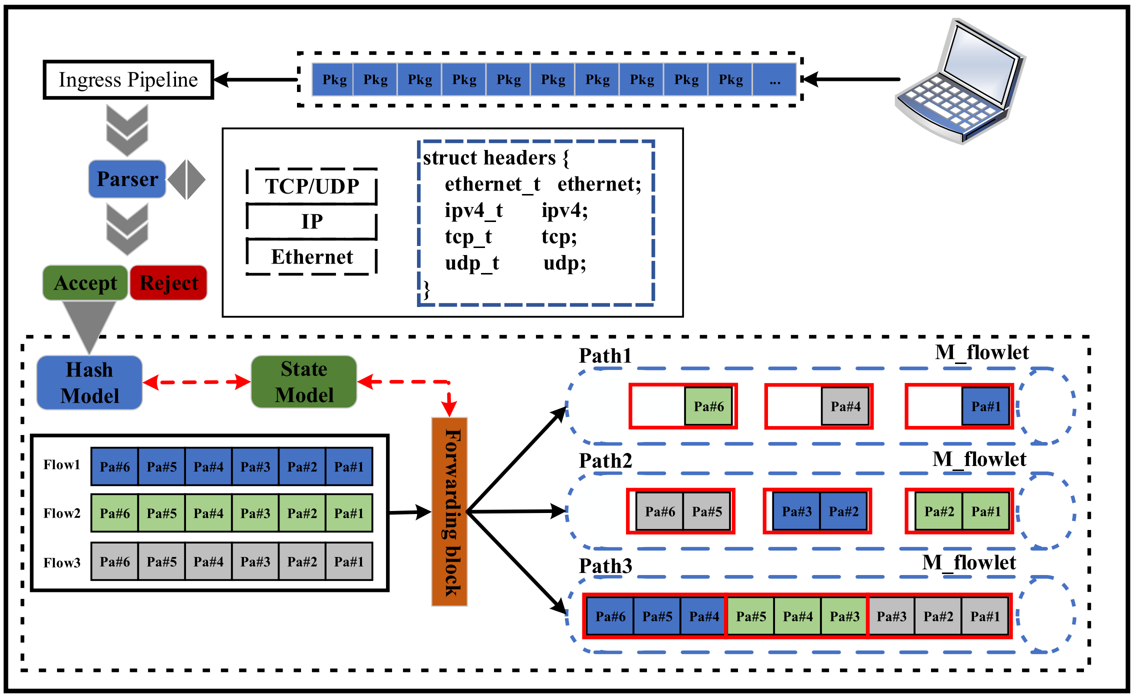


FIGURE 1. The system block diagram.

then the flows are scheduled independently. Cao *et al.* [14] proposed that hashing using a 16-bit Cyclic Redundant Checksum (CRC) over the five-tuple gives excellent load distribution performance. Specifically, the ISO standard proposed a 16-bit CRC algorithm. Although the algorithm is more complicated, it has a better effect on the hash of the five-tuple. The hash function can be expressed as:

$$H(\cdot) = CRC16(\text{five-tuple}) \bmod N \qquad (1)$$

In Formula 1, the parameter N determines the size of the FlowStateTable. Through the result of the hash function, we get the initial forwarding path corresponding to the flow where the packet is located. Then, based on the network performance of the paths, we use the state mechanism to divide the flow into flowlets with different sizes.

B. STATEFUL FORWARDING MECHANISM DESIGN

The stateful forwarding plane has been proved to be beneficial to delivery efficiency [27]. As mentioned above, the status block is mainly responsible for the maintenance of data plane status information. In the traditional Software Defined Network (SDN) data plane, the processing of data packets can generally be abstracted as a "match + action" forwarding processing paradigm. As shown in Figure 2(a), after the packets enter the data plane, the parsed header information is matched with the local flow table entries, and then forwarded, discarded, encapsulated, or encrypted according to the action table corresponding to the flow table. Based on the above forwarding abstract model, the traditional SDN data plane "match + action" processing paradigm does not

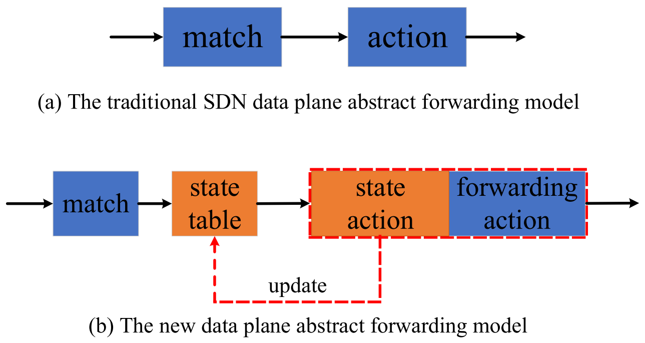

match
action
(a) The traditional SDN data plane abstract forwarding model
match
state table
state action
forwarding action
update
(b) The new data plane abstract forwarding model

FIGURE 2. The forwarding model.

support stateful forwarding, because it cannot process state information. However, we design a state processing mechanism with the programmable switch BMV2 and the protocol-independent programming language, which can be abstracted as "match + state + action." As shown in Figure 2(b), we add a local state table to the data plane, extended the action operation, and divided it into status operations and forwarding operations. We use the status operation to update the status table and implement state transition, and use the forwarding operation to forward data packets to the designated port.

Specifically, there are two status tables in the status block. As illustrated in Table 1, FlowStateTable is used to record the state of flows and packets. We use the "hash" field in the table to match the hash value of the five-tuple of the packet. The purpose is to divide the packet into different flows for separate processing. The "count" field in the table is used to

TABLE 1. FlowStateTable.

hash	count	initial_port	current_port
0x00	0	random(1,2,3)	NULL
0x01	0	random(1,2,3)	NULL
...	...	...	...
0xff	0	random(1,2,3)	NULL

record the current state of the stream. The state information is the key to split the flow into flowlets. The "initial_port" field is used to record the initial forwarding path of the flow. This can prevent multiple flows from being scheduled to the same path when they are transmitted in parallel, thereby achieving a better load balancing effect. The "current_port" field is used to record the current forwarding path of the flow. Each flow has an independent state, which makes the scheduling mechanism more flexible.

TABLE 2. PathStateTable.

port_num	state	invalid_count	weight
1	Valid	0	w_1
2	Valid	0	w_2
...	...	...	...
n	Invalid	0	w_n

PathStateTable records the current status of each path. As shown in Table 2, the "port_num" field records the port numbers of all paths to the device. The "state" field records the state of the path corresponding to the port, this field has two optional states, "Valid" indicates that the path is available, and "Invalid" indicates that the path is unavailable. The "invalid_count" field records the number of consecutive times that the path was determined to be blocked. When this field exceeds the preset "invalid_threshold", the state of the path will be updated to "Invalid", and packets will not be forwarded from the path until the path is reactivated. The "weight" field records the forwarding weight of the path, and the value is calculated by Formula 2. In the formula, bw_i represents the bandwidth of the path corresponding to the port whose number is i, w_i represents the weight of the path corresponding to the port whose number is i.

$$w_i = \frac{bw_i}{\sum_{j=1}^{n} bw_j} \tag{2}$$

Although programmable switch BMV2 itself does not provide any status mechanism, with the aid of powerful expression ability of P4 language [39] and the programmable switch, we implement the stateful forwarding mechanism mentioned above. In this mechanism, we use registers that in the programmable switch to store state tables and use P4's Match-Action unit to design state operation instructions, to maintain and process network, flow, and message state information in the data plane. The existence of state module improves the "intelligence" of the data plane and endows data plane with the ability to process state information, and improves the efficiency of data plane processing. The dynamic forwarding algorithm and the flowlet-based forwarding algorithm of the mechanism proposed in this paper all benefit from the state processing mechanism of the data plane.

C. THE IMPLEMENTATION OF DYNAMIC FORWARDING MECHANISM

The dynamic mechanism can dynamically adjust the forwarding strategy according to the current network state in the unreliable link-state, thereby improving the stability of the system [40]. Specifically, the implementation of our dynamic forwarding mechanism depends on the perception of network congestion. The traffic management engine of the BMV2 switch supports strict priority scheduling with rate limitation, whose buffer model is illustrated in Figure 3. In BMV2 switch, the first-in-first-out (FIFO) buffers of input and output are shared by all ports, but the priority queue for traffic management is divided based on ports. Specifically, assuming that there are no packets in the BMV2 switch, it will be enqueued into the input buffer when BMV2 receives the first packet. Then, the ingress thread is notified and transfer the packet from the input buffer to the row of priority queue corresponding to the egress port, which is determined by the flow table.

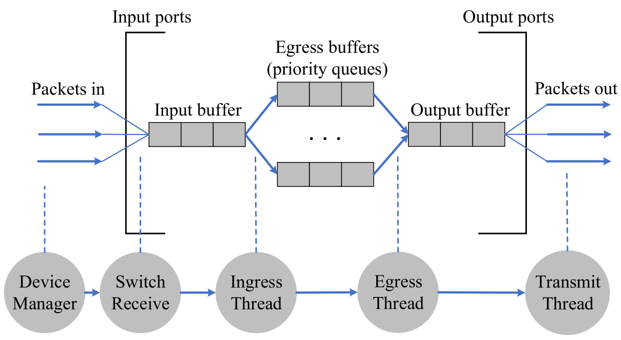


FIGURE 3. Ingress and egress ports and their threads pipeline models.

The egress buffer supports rate-limited strict priority scheduling, and the rate has packets per second (PPS) units. When the egress thread is activated, the packets will be dequeued from the egress buffer and enqueued to the output buffer. We can obtain the bandwidth and connectivity of each link periodically, then we can determine the packet forwarding rate corresponding to different ports according to the bandwidths and update the "state" field of the path state table. In BMV2, we can set the packets sending rate of port i to K PPS by using the command "set_queue_rate k i". Subsequently, the cache queue depth change can reflect the congestion status of the path in real-time. However, in the real scenario, the state of the network changes dynamically. If the observation period is too long, we can not get the state of the network in real-time, but frequent network status observations will consume too much network resources, so it is difficult to obtain the instantaneous state of the network without increasing the network load. In this paper, we collect

Algorithm 1 The Dynamic Forwarding Mechanism

```
Require:
    current_port: The number of current forwarding port.
    que_threshold: The threshold of network congestion.
    invalid_threshold: The threshold of path interruption.
Ensure:
    temp_port: The next forwarding port.

1:  procedure SELECTPATH start:
2:  temp_port ⇐ getport(current_port) depend on the round-robin scheduling;
3:  state ⇐ get the state from PathStateTable where port_num = temp_port;
4:  if state == valid then
5:      que_depth ⇐ getque(temp_port);
6:      if que_depth < que_threshold then
7:          update the invalid_count to 0 in PathStateTable where the port_num = temp_port;
8:          return temp_port;
9:      else
10:         invalid_count ⇐ get from the PathStateTable where port_num = temp_port;
11:         if invalid_count < invalid_threshold then
12:             update the invalid_count to invalid_count + 1 in PathStateTable where the port_num = temp_port;
13:             temp_port ⇐ get the port with the minimum que_depth;
14:             return temp_port;
15:         else
16:             update the state to Invalid in PathStateTable where the port_num = temp_port;
17:             current_port = temp_port;
18:             go to start;
19:         end if
20:     end if
21: else
22:     current_port = temp_port;
23:     go to start;
24: end if
25: end procedure
```

data for 10 seconds every 5 minutes and use the average value as the current state of the network.

As shown in Algorithm 1, the processing flow of the dynamic mechanism can be described as follows:

- *F*irst, after determining the temp forwarding port of the packet depends on the round-robin scheduling, the forwarding block gets the state of the *temp_port* from the *PathStateTable*. If the state is valid, go to the next step; otherwise, repeat the first step.
- *T*hen, get the queue depth of the *temp_port* from the BMV2 and compare the value with the congestion threshold—*que_threshold* (the *que_threshold* is 5 in this paper). If the obtained real queue depth does not exceed the threshold value, forward it directly, update the *invalid_count* to *PathStateTable* and return *temp_port*, otherwise perform the next step.
- *T*hird, read the value of *invalid_count* from the table *PathStateTable* and compare the value with the interruption threshold—*invalid_threshold* (the *invalid_threshold* is 4 in this paper). If the value of *invalid_count* is not less than *invalid_threshold*, change the *state* of the *temp_port* from valid to invalid and set the weight of the *temp_port* to 0 in the *PathStateTable* then go back to the first step, otherwise go to the next step.
- *F*inally, update the *invalid_count* of the *temp_port* in table *PathStateTable* to *invalid_count* + 1, then obtain the *que_depth* values of all ports and find the port number corresponding to the queue with the lowest value as the new *temp_port*. After that, return *temp_port*.

D. THE IMPLEMENTATION OF FLOWLET-BASED FORWARDING MECHANISM

As described in Section A, the flowlet-based forwarding mechanism has more advantages comparing with the packet-based forwarding mechanism and the flow-based forwarding mechanism. On the one hand, it has a lower percentage of out of order packets; on the other hand, it has more flexible scheduling capabilities. The implementation of this mechanism benefits from the implementation of the stateful forwarding mechanism and the programmable switch. Specifically, The pseudo-code of the flowlet-based forwarding algorithm proposed in this paper is shown in the algorithm 2.

Firstly, we need to initialize the state tables and parameters, and to set parameters based on the network performance of the paths. There are two important parameters in the algorithm: *que_threshold* and *invalid_threshold*, which will directly affect the sensitivity of the dynamic algorithm. This parameter should be set according to the actual network requirements. The higher the value of *que_threshold*, the slower the algorithm responds to path congestion, the higher the value of *invalid_threshold*, the slower the algorithm responds to path interruption.In this paper, the parameters *que_threshold* and *invalid_threshold* are set to 5 and 3, respectively. The parameter *flowsize* calculated by Formula 3, where w_i represents the weight of the port whose number is i, m represents the minimum weights.

$$\text{flowsize} = \left\lfloor [w_1, w_2, \cdots, w_n] \times \begin{bmatrix} \frac{1}{m} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \frac{1}{m} \end{bmatrix} \right\rfloor \quad (3)$$

Then, the hash block calculates the five-tuple hash of the packet, while dividing the packets into flows based on the hash value. After that, split the flow into flowlets based on the count field in *FlowStateTable* and the parameter *flowsize*. After all the packets of the current flowlets are forwarded, call

Algorithm 2 Flowlet-Based Forwarding Mechanism

```
1: Initialize
2: $FlowStateTable.count \Leftarrow default\ 0$;
3: $FlowStateTable.initial\_port \Leftarrow random(1, 2, 3)$;
4: $FlowStateTable.current\_port \Leftarrow initial\_port$;
5: $PathStateTable.state \Leftarrow defaultValid$;
6: $PathStateTable.que\_depth \Leftarrow default\ 0$;
7: $PathStateTable.block\_count \Leftarrow default\ 0$;
8: $PathStateTable.weight \Leftarrow$ calculated by formula 2.
9: procedure start:p
10: $que\_threshold = 5$;
11: $invalid\_threshold = 3$;
12: $flowsize \Leftarrow$ get the size of flows by formula 3;
13: while receive one packet do
14:     $key \Leftarrow hash$(the five-tuple of packet);
15:     $current\_count \Leftarrow$ get from $FlowStateTable$ where $hash = key$;
16:     $current\_port \Leftarrow$ get from $FlowStateTable$ where $hash = key$;
17:     if $current\_count < (\text{flowszie}_{\text{current\_port}} - 1)$ then
18:         update the $current\_count$ to $current\_count + 1$ in $FlowStateTable$ where $hash = key$;
19:     else
20:         update $current\_count$ to 0 in $FlowStateTable$ where $hash = key$;
21:         $next\_port$ = SELECTPATH($current\_port, que\_t-hreshold, invalid\_threshold$);
22:         update the $current\_port$ to $next\_port$ in the table $FlowStateTable$ where $hash = key$;
23:     end if
24:     send packet to $current\_port$;
25: end while
26: end procedure
```

the function SELECTPATH to dynamically select the next forwarding path.

IV. PERFORMANCE EVALUATION

In this section, we evaluate the performance of the FSMF mechanism in the asymmetric network by comparing it with the traditional RRS mechanism through experimental simulations. In the simulation, two scenarios are included: the normal condition and path interruption condition. For the first scenario, we compare the system performance of the two strategies when all paths are reachable. In the second scenario, we analyze the system performance of the two strategies when path P_1 is interrupted. The experimental simulation tools used in this paper are the network simulator minnet (2.3.0d4) [41], [42] and the programmable switch BMV2 (1.11.0). The evaluation of the two mechanisms' network performance is based on the network performance test tool, iperf3, and the network packet analysis software, Wireshark. During the experiment, we minimize the experimental errors by repeating experiments.

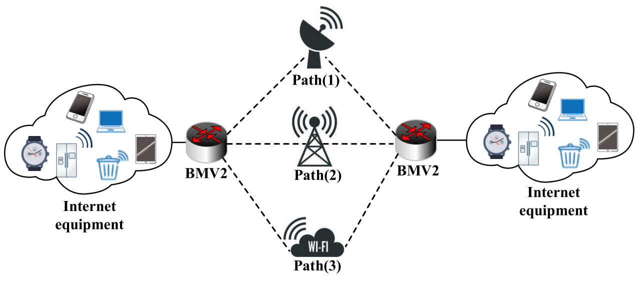


FIGURE 4. The testbed topology.

TABLE 3. The network parameters of paths.

Path Index	Network Parameters	
	Bandwidth	*Delay*
P_1	1 Mbps	100 ms
P_2	2 Mbps	40 ms
P_3	3 Mbps	60 ms

TABLE 4. Hardware parameters.

CPU	Inter Core i7-6700 CPU @ 3.4GHz 3.41GHz
RAM	8.0G
Storage	100G
Operating System	16.04.1-Ubuntu x86_64 GNU/Linux
BMV2 forwarding rate	1350 Mbps

A. EXPERIMENTAL ENVIRONMENT

The topology of wireless network testbed is shown in Figure 4. There are three optional communication paths between the data sender and the data receiver. To simulate complex network environments, especially wireless networks, the network parameters of the three paths in the topology are significantly different, and the parameters of the experimental system are shown in Table 3. Expressly, the bandwidth of three paths P_1, P_2 and P_3 are set to 1, 2 and 3 Mbps, respectively. The delay of three paths P_1, P_2 and P_3 are set to 100, 40 and 60 ms, respectively. The hardware configuration parameters for system simulation are shown in Table 4. Under the current hardware conditions, we benchmarked the forwarding performance of BMV2. The maximum throughput of the switch is 1350 Mbps.

B. PERFORMANCE EVALUATION IN TYPICAL SCENARIOS

In this section, we will analyze the network performance of the system in typical scenarios, where all paths are reachable. As mentioned above, the FSMF mechanism proposed in this paper is a flowlet-based forwarding mechanism. It ensures that the packets in the same flowlet will be forwarded from the same path, so it has a higher probability to arrive in order. Compared with the traditional packet-based forwarding mechanism, our mechanism has a lower frequency of path switching during the data forwarding process. Therefore, at the system level, it will have a smaller delay jitter. It also means that the percentage of out of order packets can be reduced, thereby benefiting the system throughput.

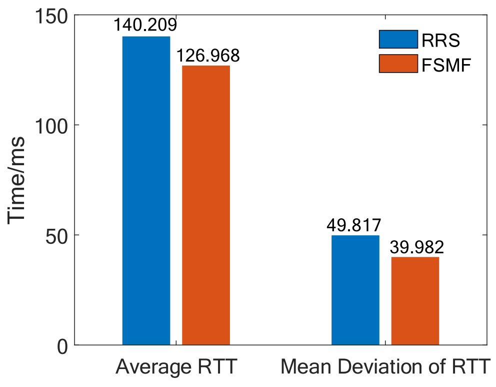
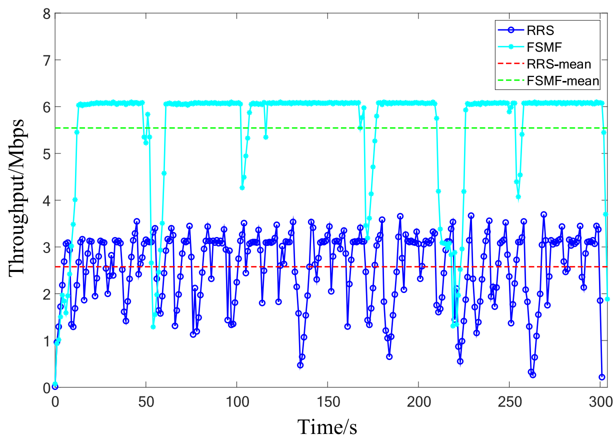


FIGURE 5. The system RTT analysis.

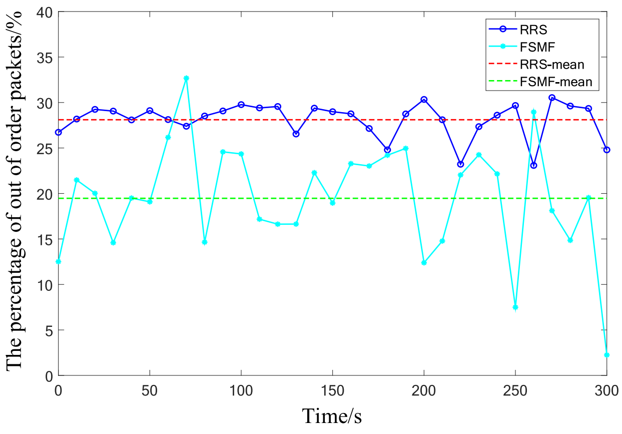
Figure 5 presents the average RRT and the mean deviation of RRT in the FSMF mechanism are better than those in the SSR mechanism. Explicitly, the average RTT of the mechanism proposed in this paper and the contrast mechanism are 126.968 ms and 140.209 ms, respectively; the mean deviation of RRT in the mechanism proposed in this paper and the contrast mechanism are 39.982 ms and 49.817 ms respectively. Experimental results show that our mechanism achieves the goal of reducing system delay jitter by trying to avoid frequent switching between paths with significant delay differences. Besides, the mechanism proposed in this paper also improves network throughput and the percentage of out of order packets.

Figure 6 shows the data collected at one time, which lasted 300 seconds. In Figure 6(a), we present simulation results of throughput for the bandwidth and delay asymmetric network scenarios. As mentioned in the previous section, the bandwidths of the three paths are set to 1,2 and 3 Mbps. Therefore, if the bandwidth resources of each path are fully utilized, the capacity of the system should be 6 Mbps. But in fact, the simulation results show that the throughput of the system based on the RRS mechanism is only 2.58 Mbps, which is much lower than what we expected. In Figure 6(b), we present simulation results of the percentage of out of order packets for the asymmetric network scenario. It is generally considered that the percentage of out of order packets in the single-path scenario is about 3%. But the simulation results present that the the percentage of out of order packets in the multipath scenario basing on the RRS mechanism is 28.5%, which is much higher than 3%. Generally, compared with the RRS mechanism, our mechanism has improved the throughput by 114.7%, from 2.58 Mbps to 5.54 Mbps; the disorder rate decreased by 29.5%, from 28.5% to 20.1%.

Additionally, just as shown in Figure 6(a), our mechanism has higher network throughput. The reason is that our mechanism can make full use of bandwidth resources of each path. In Figure 6(b), we find that our mechanism has a lower percentage of out of order packets in most cases. However, in the 70th second and 260th second, the percentage of out of order packets in our mechanism becomes more serious, even exceeding the comparison mechanism. This is because, during the experimental simulation, we must separately verify



(a) Throughput



(b) The percentage of out of order packets

FIGURE 6. The network performance analysis in typical scenarios.

the RRS mechanism and the FSMF mechanism. Although the experimental parameters we specified are entirely consistent during the verification process, due to the volatility of the network, we cannot guarantee that the network performance is entirely consistent. Therefore, we collected real experimental data for 5 minutes in a row. By focusing on the mean, we prove that our mechanism has a universal optimization effect.

Furthermore, we compare the TCP windows of RRS mechanism and FSMF mechanism in Figure 7. The picture mainly contains information such as the window size, the bytes in flight, and the free receiver window size. Specifically, compared with the RRS mechanism, our mechanism is significantly higher in both the window size and the bytes in flight, which corresponds to higher network throughput. And, the mechanism proposed in this paper can more smoothly and efficiently adjust the receiving window size; at the same time, it can utilize network resources more effectively.

C. PERFORMANCE EVALUATION IN ABNORMAL SCENARIOS

In this section, we will analyze the network performance of the system in abnormal scenarios. The abnormal situation

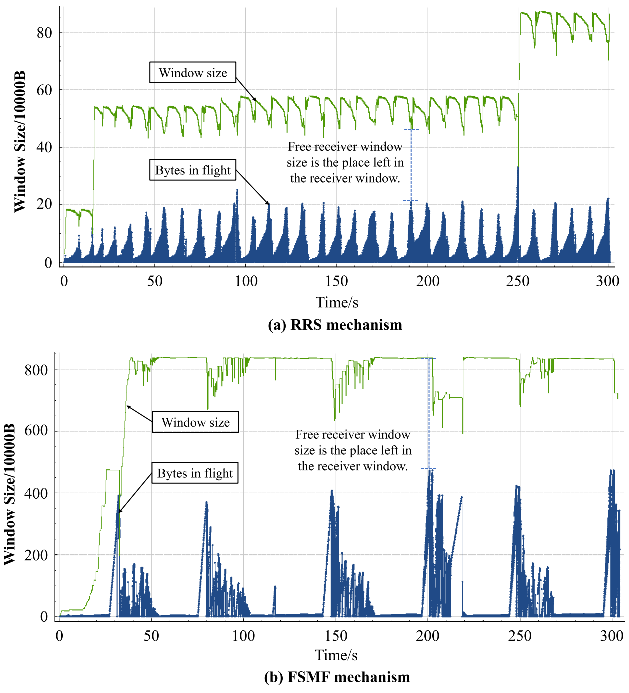


FIGURE 7. **The TCP window analysis in typical scenarios.**

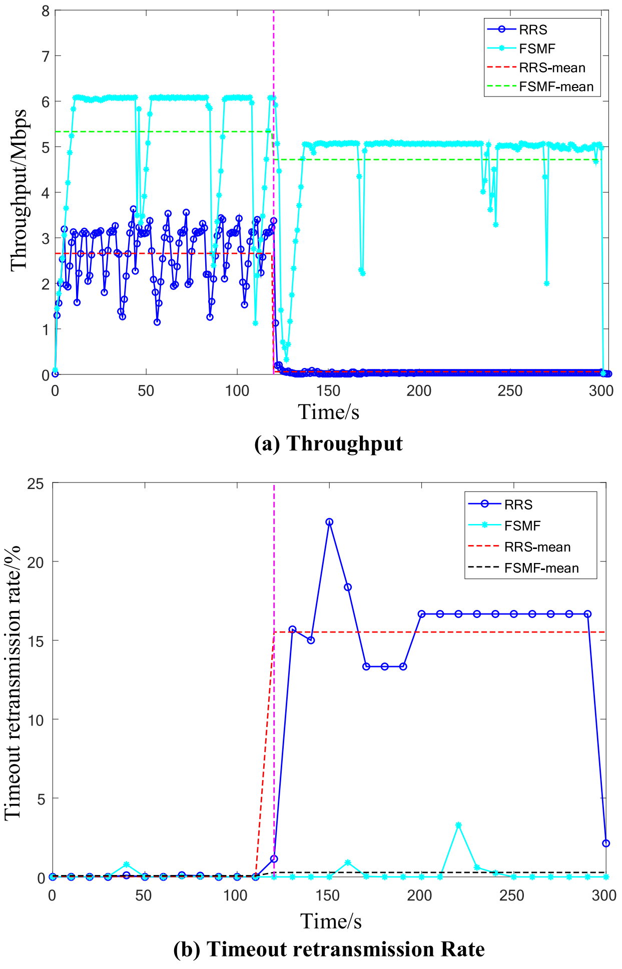


FIGURE 8. **The network performance analysis when path P_1 interrupted.**

means that we will artificially interrupt one of the three paths in the course of the experiment to analyze the difference between our mechanism and the RRS mechanism in the actual application scenario when facing extreme network conditions. Specifically, when the simulation reaches 120 s, we will manually cut off the path P_1, whose bandwidth and delay are 1 Mbps and 100 ms, respectively. Same as the previous simulation experiment, to avoid some random abnormal data affecting the final experimental conclusions, we collect experimental data for 5 minutes. When analyzing experimental results, we focus on overall results.

In Figure 8, we present the simulation results of the throughput and the timeout retransmission rate when path P_1 interruption occurs, and the vertical pink dotted line in the figure indicates the moment when the path interruption occurs. As depicted in Figure 8(a), in the multipath forwarding scenario based on the RRS mechanism, the throughput of the system dropped sharply from 2.66 Mbps at the beginning to 0.04 Mbps after the path interruption occurred. The decrease in network throughput is happening because the RRS mechanism cannot sense the congestion status of the network in real-time, so it will continue sending packets to the path which has been interrupted. Consequently, a large number of packets are retransmitted due to timeout. Afterward, the TCP congestion control mechanism was triggered to reduce the size of the congestion window. Finally, the throughput of the network decreased dramatically.

It is worth noting that different from the concurrent multipath transfer basing on the RRS mechanism. The throughput of the mechanism proposed in this paper does not decrease dramatically due to the path interruption. Because it can find the path interruption in time, and transfer the packets that should be forwarded from the interrupted path to other available paths. As shown in Figure 8(a), the throughput of the system reduced from 5.34 Mbps at the beginning to 4.70 Mbps after the path interruption occurred. The path interruption reduced the system's network throughput by approximately 12.0%. However, In the contrast mechanism, path interruption reduced the system's network throughput by about 98.5%. This set of data proves that the mechanism proposed in this paper has better performance in the face of extreme abnormal conditions such as path interruption.

Figure 8(b) shows the timeout retransmission rate of RRS and FSMF mechanism in abnormal scenarios. Specifically, the retransmission timeout rate of the RRS mechanism increased sharply from 0.02% to 15.52%, at the same time, the retransmission timeout rate of our mechanism increased Slightly from 0.07% to 0.28%. Compared with the contrast mechanism, our mechanism has a significantly lower timeout retransmission rate when the path interruption occurs, and our mechanism can make the timeout retransmission rate within a reasonable range. Therefore, the mechanism proposed in this paper has higher reliability.

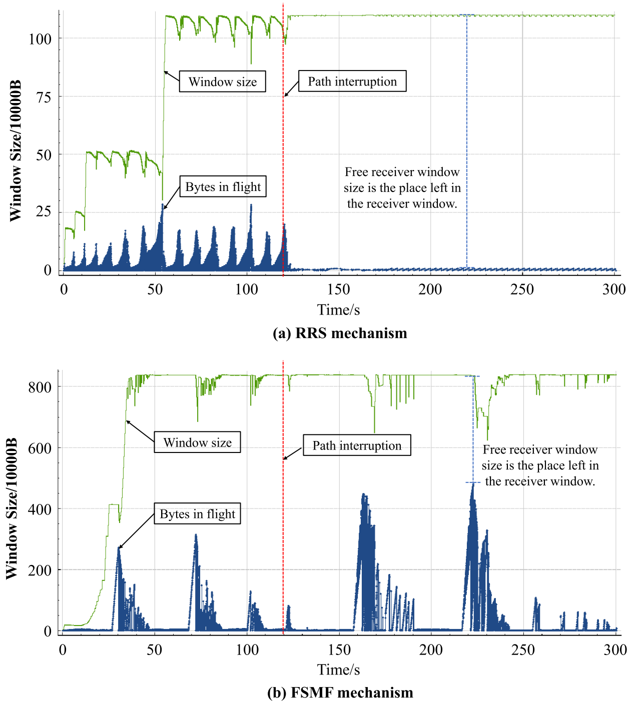


FIGURE 9. The TCP window analysis when path P_1 interrupted.

Additionally, we find that when a path interruption occurs, the congestion control mechanism of TCP will also have a significant influence, which will lead to the deterioration of network performance indicators such as system throughput. Figure 9 describes the comparison of the TCP window for the RRS mechanism and FSMF mechanism in abnormal scenarios. In Figure 9(a), we find that the number of bytes in flight decreases dramatically after the path is interrupted. At the same time, the receiving end no longer needs to adjust the window size frequently due to the reduction of the received data packets. In general, when the path is interrupted, the network resource utilization of the RRS mechanism is significantly reduced.

On the other hand, Figure 9(b) shows that when the path is interrupted, the TCP congestion control mechanism is significantly less affected in our mechanism. Furthermore, the number of bytes in flight actually increases after the path is interrupted. Before the path was interrupted, the total bandwidth resource of the system was 6 Mbps. The actual throughput we observed was 5.34 Mbps, and the utilization rate of the bandwidth resource was 89%. After the path was interrupted, the total bandwidth resource of the system was 5 Mbps. The actual throughput we observed was 4.70 Mbps; the utilization of bandwidth resources is 94%. It means that, after the path is interrupted, the system's utilization of bandwidth resources becomes even higher. This is because when we interrupt path P_1, the difference of the network performance parameters between the remaining paths in the system becomes smaller. Although the total bandwidth of the system is reduced, the delay jitter of the system is also lower. The reliability of the system in abnormal scenarios benefits from the dynamic routing mechanism in the FSMF mechanism.

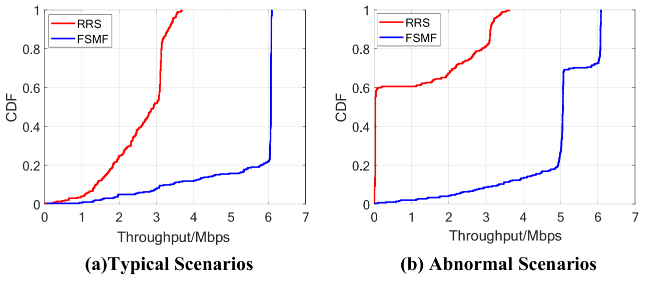


FIGURE 10. CDF of throughput values.

Figure 10 presents the cumulative distribution function (CDF) of throughput values of RRS mechanism and FSMF mechanism in different scenarios. As summarized in the previous chapters, no matter in the traditional scenario or the abnormal scenario, compared with the RRS mechanism, the network throughput of the FSMF mechanism is always better. Because the mechanism proposed in this paper fully considers the bandwidth differences of different paths in the multipath scheduling process and minimizes the frequency of path switching as much as possible. At the same time, our mechanism predicts the congestion status of different pathways in real-time through the state information of the data plane. On the one hand, it can achieve more flexible scheduling of packets; on the other hand, it can sense whether the path is reachable in real-time and then decide whether re-plan a more optimal path scheduling mechanism. Besides, When the path interruption is recovered, the network performance will be restored to the condition before the interruption in 5 minutes.

V. CONCLUSION

This paper proposed a flowlet granularity multipath dynamic forwarding mechanism for the heterogeneous internet of things, which mainly includes three contributions. First, in the programmable data plane, we designed a stateful forwarding mechanism to improve the ability to process cross-layer information to achieve complex processing logic and higher forwarding efficiency. Second, we implemented a flowlet-based multipath forwarding method, which reduces the transmission delay jitter by avoiding frequent path switching. Third, we proposed a dynamic scheduling mechanism based on congestion awareness, which senses the path congestion and reachability by analyzing the status in the programmable data plane in real-time and dynamically adjust the scheduling mechanism to improve the reliability. In the future work, we will optimize the calculation method for path weight, w, by considering network parameters comprehensively, which will make the path weight more reasonable and further improve the network performance of the system. The network parameters that can be considered may include delay [43], throughput, and packet loss rate.

REFERENCES

[1] M. Presser, P. Barnaghi, M. Eurich, and C. Villalonga, "The SENSEI project: Integrating the physical world with the digital world of the network of the future," *IEEE Commun. Mag.*, vol. 47, no. 4, pp. 1–4, Apr. 2009.

[2] X. Cheng, F. Lyu, W. Quan, C. Zhou, H. He, W. Shi, and X. Shen, "Space/aerial-assisted computing offloading for IoT applications: A learning-based approach," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 5, pp. 1117–1129, May 2019.

[3] H. Zhou, W. Xu, J. Chen, and W. Wang, "Evolutionary V2X technologies toward the Internet of vehicles: Challenges and opportunities," *Proc. IEEE*, vol. 108, no. 2, pp. 308–323, Feb. 2020.

[4] M. Zhang, Y. Zhou, W. Quan, J. Zhu, R. Zheng, and Q. Wu, "Online learning for IoT optimization: A Frank-Wolfe Adam based algorithm," *IEEE Internet Things J.*, early access, Mar. 30, 2020, doi: 10.1109/JIOT.2020.2984011.

[5] G. Liu, W. Quan, N. Cheng, H. Zhang, and X. Shen, "VLI: Variable-length identifier for interconnecting heterogeneous IoT networks," *IEEE Wireless Commun. Lett.*, early access, Mar. 26, 2020, doi: 10.1109/LWC.2020.2982641.

[6] W. Quan, Y. Liu, H. Zhang, and S. Yu, "Enhancing crowd collaborations for software defined vehicular networks," *IEEE Commun. Mag.*, vol. 55, no. 8, pp. 80–86, Aug. 2017.

[7] H. Zhang, P. Dong, W. Quan, and B. Hu, "Promoting efficient communications for high-speed railway using smart collaborative networking," *IEEE Wireless Commun.*, vol. 22, no. 6, pp. 92–97, Dec. 2015.

[8] N. Cheng, F. Lyu, J. Chen, W. Xu, H. Zhou, S. Zhang, and X. Shen, "Big data driven vehicular networks," *IEEE Netw.*, vol. 32, no. 6, pp. 160–167, Nov. 2018.

[9] A. Ford, C. Raiciu, M. Handley, S. Barre, and J. Iyengar, *Architectural Guidelines for Multipath TCP Development*, document RFC 6182, Mar. 2011. [Online]. Available: https://tools.ietf.org/html/rfc6182

[10] G. Sarwar, R. Boreli, E. Lochin, and A. Mifdaoui, "Performance evaluation of multipath transport protocol in heterogeneous network environments," in *Proc. Int. Symp. Commun. Inf. Technol. (ISCIT)*, Gold Coast, QLD, Australia, Oct. 2012, pp. 985–990.

[11] K.-C. Leung, V. O. K. Li, and D. Yang, "An overview of packet reordering in transmission control protocol (TCP): Problems, solutions, and challenges," *IEEE Trans. Parallel Distrib. Syst.*, vol. 18, no. 4, pp. 522–535, Apr. 2007.

[12] E. L. Hahne, "Round-robin scheduling for max-min fairness in data networks," *IEEE J. Sel. Areas Commun.*, vol. 9, no. 7, pp. 1024–1039, Sep. 1991.

[13] C. Paasch, S. Ferlin, O. Alay, and O. Bonaventure, "Experimental evaluation of multipath TCP schedulers," in *Proc. ACM SIGCOMM Workshop Capacity Sharing Workshop (CSWS)*, 2014, pp. 27–32.

[14] Z. Cao, Z. Wang, and E. Zegura, "Performance of hashing-based schemes for Internet load balancing," in *Proc. IEEE Conf. Comput. Commun., 19th Annu. Joint Conf. IEEE Comput. Commun. Soc.*, vol. 1, Mar. 2000, pp. 241–332.

[15] C. Hopps, *Analysis of an Equal-Cost Multi-Path Algorithm*, document RFC 2992, Nov. 2000. [Online]. Available: https://tools.ietf.org/html/rfc2992

[16] J. Zhou, M. Tewari, M. Zhu, A. Kabbani, L. Poutievski, A. Singh, and A. Vahdat, "WCMP: Weighted cost multipathing for improved fairness in data centers," in *Proc. ACM EuroSys*, no. 5, 2014, pp. 1–14.

[17] R. Stewart, *Stream Control Transmission Protocol*, document RFC 4960, Sep. 2007. [Online]. Available: https://tools.ietf.org/html/rfc4960

[18] A. Ford, C. Raiciu, M. Handley, and O. Bonaventure, *TCP Extensions for Multipath Operation With Multiple Addresses*, document RFC 6824, Jan. 2013. [Online]. Available: https://tools.ietf.org/html/rfc6824

[19] W. Tang, Y. Fu, P. Dong, W. Yang, B. Yang, and N. Xiong, "A MPTCP scheduler combined with congestion control for short flow delivery in signal transmission," *IEEE Access*, vol. 7, pp. 116195–116206, 2019.

[20] Y.-S. Lim, E. M. Nahum, D. Towsley, and R. J. Gibbens, "ECF: An MPTCP path scheduler to manage heterogeneous paths," in *Proc. 13th Int. Conf. Emerg. Netw. Exp. Technol.*, Nov. 2017, pp. 147–159.

[21] K. Gao, C. Xu, J. Qin, L. Zhong, and G.-M. Muntean, "A stochastic optimal scheduler for multipath TCP in software defined wireless network," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Shanghai, China, May 2019, pp. 1–6.

[22] H. Nam, D. Calin, and H. Schulzrinne, "Towards dynamic MPTCP path control using SDN," in *Proc. IEEE NetSoft Conf. Workshops (NetSoft)*, Jun. 2016, pp. 286–294.

[23] G. Bianchi, M. Bonola, A. Capone, and C. Cascone, "OpenState: Programming platform-independent stateful openflow applications inside the switch," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 2, pp. 44–51, Apr. 2014.

[24] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling innovation in campus networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, Apr. 2008.

[25] S. Zhu, J. Bi, C. Sun, C. Wu, and H. Hu, "SDPA: Enhancing stateful forwarding for software-defined networking," in *Proc. IEEE 23rd Int. Conf. Netw. Protocols (ICNP)*, San Francisco, CA, USA, Nov. 2015, pp. 323–333.

[26] C. Yi, A. Afanasyev, I. Moiseenko, L. Wang, B. Zhang, and L. Zhang, "A case for stateful forwarding plane," *Comput. Commun.*, vol. 36, no. 7, pp. 779–791, Apr. 2013.

[27] G. Liu, W. Quan, N. Cheng, H. Zhang, and S. Yu, "Efficient DDoS attacks mitigation for stateful forwarding in Internet of Things," *J. Netw. Comput. Appl.*, vol. 130, pp. 1–13, Mar. 2019.

[28] M. T. Arashloo, Y. Koral, M. Greenberg, J. Rexford, and D. Walker, "SNAP: Stateful network-wide abstractions for packet processing," in *Proc. ACM SIGCOMM Conf.*, 2016, pp. 29–43.

[29] H. Hu, W. Han, G.-J. Ahn, and Z. Zhao, "FLOWGUARD: Building robust firewalls for software-defined networks," in *Proc. 3rd Workshop Hot Topics Softw. Defined Netw.*, 2014, pp. 97–102.

[30] L. Galluccio, S. Milardo, G. Morabito, and S. Palazzo, "SDN-WISE: Design, prototyping and experimentation of a stateful SDN solution for wireless sensor networks," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Hong Kong, Apr. 2015, pp. 513–521.

[31] M. Moshref, A. Bhargava, A. Gupta, M. Yu, and R. Govindan, "Flow-level state transition as a new switch primitive for SDN," in *Proc. 3rd Workshop Hot Topics Softw. Defined Netw.*, 2014, pp. 61–66.

[32] C. Cascone, D. Sanvito, L. Pollini, A. Capone, and B. Sansò, "Fast failure detection and recovery in SDN with stateful data plane," *Int. J. Netw. Manage.*, vol. 27, no. 2, p. e1957, Mar. 2017.

[33] A. Capone, C. Cascone, A. Q. T. Nguyen, and B. Sansò, "Detour planning for fast and reliable failure recovery in SDN with OpenState," in *Proc. 11th Int. Conf. Design Reliable Commun. Netw. (DRCN)*, Mar. 2015, pp. 25–32.

[34] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese, and D. Walker, "P4: Programming protocol-independent packet processors," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 3, pp. 87–95, 2014.

[35] Y.-S. Lim, Y.-C. Chen, E. M. Nahum, D. Towsley, and K.-W. Lee, "Cross-layer path management in multi-path transport protocol for mobile devices," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Toronto, ON, Canada, Apr. 2014, pp. 1815–1823.

[36] M. Shreedhar and G. Varghese, "Efficient fair queuing using deficit round-robin," *IEEE/ACM Trans. Netw.*, vol. 4, no. 3, pp. 375–385, Jun. 1996.

[37] M. Chiesa, G. Kindler, and M. Schapira, "Traffic engineering with equal-cost-multipath: An algorithmic perspective," in *Proc. IEEE INFOCOM*, Apr./May 2014, pp. 1590–1598.

[38] S. Kandula, D. Katabi, S. Sinha, and A. Berger, "Dynamic load balancing without packet reordering," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 37, no. 2, pp. 53–62, Apr. 2007.

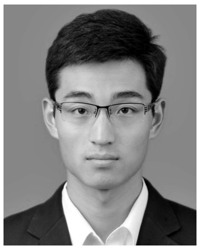
[39] (Nov. 2016). *The P4 Language Specification Version 1.0.3.* [Online]. Available: http://p4.org/wp-content/uploads/2016/11/p4-spec-latest.pdf

[40] W. Quan, N. Cheng, M. Qin, H. Zhang, H. A. Chan, and X. Shen, "Adaptive transmission control for software defined vehicular networks," *IEEE Wireless Commun. Lett.*, vol. 8, no. 3, pp. 653–656, Jun. 2019.

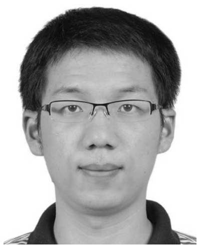
[41] (Mar. 2013). *Mininet: An Instant Virtual Network on Your Laptop (or Other PC).* [Online]. Available: http://mininet.org/

[42] R. L. S. de Oliveira, C. M. Schweitzer, A. A. Shinoda, and L. R. Prete, "Using mininet for emulation and prototyping software-defined networks," in *Proc. IEEE Colombian Conf. Commun. Comput. (COLCOM)*, Bogota, Colombia, Jun. 2014, pp. 1–6.

[43] J. Li, T. Zhang, J. Jin, Y. Yang, D. Yuan, and L. Gao, "Latency estimation for fog-based Internet of Things," in *Proc. 27th Int. Telecommun. Netw. Appl. Conf. (ITNAC)*, Melbourne, VIC, Australia, Nov. 2017, pp. 1–6.



JINYU SHI received the B.Eng. degree in communication engineering from Yanshan University, China, in 2017. He is currently pursuing the master's degree with the Department of Electronic and Information Engineering, Beijing Jiaotong University, Beijing, China. His current research interests include software defined networking (SDN), concurrent multipath transfer (CMT), and programmable data plane.

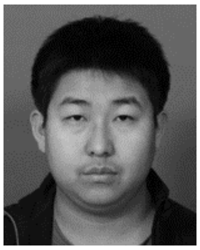


WEI QUAN (Member, IEEE) received the Ph.D. degree in communication and information system from the Beijing University of Posts and Telecommunications (BUPT), Beijing, China, in 2014. He is currently an Associate Professor with the School of Electronic and Information Engineering, Beijing Jiaotong University (BJTU). He has published more than 20 articles in prestigious international journals and conferences, including the *IEEE Communications Magazine*, the IEEE Wireless Communications, the IEEE Network, the IEEE Transactions on Vehicular Technology, the IEEE Communications Letters, IFIP Networking, the IEEE ICC, and the IEEE GLOBECOM. His research interests include key technologies for network analytics, future Internet, 5G networks, and vehicular networks. He is also a member of ACM and a Senior Member of the Chinese Association of Artificial Intelligence (CAAI). He serves as an Associate Editor for the *Journal of Internet Technology* (JIT), *Peer-to-Peer Networking and Applications (PPNA)*, and *IET Networks*, and as a technical reviewer for many important international journals.

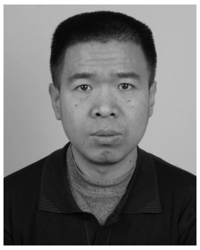
DEYUN GAO (Senior Member, IEEE) received the B.Eng. and M.Eng. degrees in electrical engineering and the Ph.D. degree in computer science from Tianjin University, China, in 1994, 1999, and 2002, respectively. He spent one year as a Research Associate with the Department of Electrical and Electronic Engineering, Hong Kong University of Science and Technology. He then spent three years as a Research Fellow with the School of Computer Engineering, Nanyang Technological University, Singapore. In 2007, he joined the Faculty with the School of Electronics and Information Engineering, Beijing Jiaotong University as an Associate Professor and was promoted to a Full Professor, in 2012. In 2014, he was a Visiting Scholar with the University of California at Berkeley, USA. His research interests include the Internet of Things, vehicular networks, and next-generation Internet.

MINGYUAN LIU received the bachelor's degree in communication engineering, in 2018. He is currently pursuing the Ph.D. degree with the School of Electronic and Information Engineering, Beijing Jiaotong University. His current research interests include future networks, software defined networking (SDN), and cybersecurity.

GANG LIU (Graduate Student Member) was born in Wuhu, Anhui, China, in March 1993. He is currently pursuing the Ph.D. degree with the National Engineering Lab for Next Generation Internet Technologies (NGIT), Beijing Jiaotong University (BJTU), Beijing, China. He is a Student Member of ACM. His current research interests include information centric networking (ICN), software defined networking (SDN), and network function virtualization (NFV).



CHENGXIAO YU (Graduate Student Member) received the B.Eng. degree in communication and information systems from Beijing Jiaotong University, Beijing, China, in 2015. He is currently pursuing the Ph.D. degree with the School of Electronic and Information Engineering, Beijing Jiaotong University. His research interests include multipath transmission, machine learning, and high-speed railway networks.



WEI SU was born in October 1978. He received the Ph.D. degrees in communication and information systems from Beijing Jiaotong University, in January 2008.

He is currently a Teacher with the School of Electronics and Information Engineering, Beijing Jiaotong University. He granted the title of Professor, in November 2015. He is mainly engaged in researching key theories and technologies for the next generation Internet and has taken part in many national projects, such as the National Basic Research Program (also called 973 Program), the Projects of Development Plan of the State High Technology Research, the National Natural Science Foundation of China. He also presides over the research project Fundamental Research on Cognitive Services and Routing of Future Internet and project funded by the National Natural Science Foundation of China.

• • •