S & M 3988

Improving the Scalability of Data Center Networking with Protocol-independent Source Routing

Shih-Pang Tseng,[1,2] Yang Yu,[3] and Chien-Min Chen[4,5*]

[1]School of Information Science and Technology, Sanda University, Shanghai,
No. 2727 Jinhai Road, Pudong District, Shanghai, 201209 China
[2]School of Software and Big Data, Changzhou College of Information Technology, Changzhou,
No. 22, Mingxin Middle Road, Changzhou, Jiangsu, 213164 China
[3]Jiangsu University of Technology, No. 1801 Zhongwu Road, Changzhou, Jiangsu, 213001 China
[4]Quanzhou University of Information Engineering,
No. 249, Bodong Road, Fengze District, Quanzhou, Fujian, 362000 China
[5]International School of Finance, Fudan University, No. 220, Handan Road, Shanghai, 200433 China

(Received December 18, 2023; accepted June 24, 2024)

Keywords: SDN, control plane, source routing, protocol-oblivious forwarding, table item

Sensor networks generate large volumes of data that require robust processing, storage, and analysis capabilities provided by data centers. Software-defined networking, as a new network technique for designing and building networks, provides new concepts and solutions for the current data center system construction. However, the software-defined networking (SDN) framework based on the OpenFlow protocol cannot support new protocols or actively add unknown protocols, since they cause the control plane protocol to become more bloated. It is difficult to solve the problem of network scalability because components are being continuously redesigned. Therefore, an optimized source routing technology is proposed in combination with the protocol-oblivious forwarding technology. Furthermore, a low-cost and extensible network switch is designed to realize the independent data forwarding mechanism that effectively reduces the amount of network control signaling. The experimental results show that the scheme can significantly reduce the scale of flow table items and improve the hardware forwarding rates as well as the flexibility of forwarding rules. Moreover, the design scheme has a good application prospect.

1. Introduction

To gain more information for better awareness of the environment or to monitor system parameters, more sensors distributed over a geographic area are needed to construct a sensor network.(1) These sensors can detect physical phenomena such as temperature, humidity, light, pressure, motion, and sound. Sensor networks are used in various applications, including environmental monitoring, industrial automation, healthcare, and smart cities. The sensors transmit the collected data to a central node or gateway, which processes and transmits the information to a remote server or data center for further analysis and decision-making.

*Corresponding author: e-mail: hally888@gmail.com
https://doi.org/10.18494/SAM4828

A data center is a dedicated facility used to house computer systems and associated components, such as telecommunications and storage systems. It includes the physical infrastructure for computing, storage, and networking hardware, as well as the software and services required to support the critical applications and data. Data centers can manage huge amounts of data generated by sensor networks, enabling real-time analytics, data storage, and the implementation of various applications driven by sensor data.

The integration of sensor networks with data centers is crucial for leveraging the full potential of Internet of Things (IoT) applications. Sensor networks generate large volumes of data that require the robust processing, storage, and analysis capabilities provided by data centers. This integration supports real-time monitoring, predictive analytics, and efficient management of resources, leading to enhanced operational efficiency, informed decision-making, and improved outcomes in various sectors. The data exchange efficiency of the data center has become the development bottleneck of the sensor network scalability.

Software-defined data centers (SDDC) have realized abstraction, pooling, and the automated deployment and management of all physical computing, storage, and network resources through virtualization and software to meet higher agility business demands and user experience.(2,3) Among these technologies, software-defined networking (SDN) architecture separates the control plane used for network decision from the data plane for data forwarding and provides network programmability to accelerate network innovation.(4,5) The application of SDN to build the data center can effectively simplify the network management and achieve precise scheduling of data flows, further enhancing the service carrying capacity.

At present, the processing mechanism and traffic characteristics of the flow table are limited in terms of flow table space and energy consumption in data centers. The OpenFlow switches usually use ternary content addressable memory (TCAM) devices for item storage,(6) yet the high cost and power consumption, along with the limited capacity, make the switch structure inflexible and inefficient, and it is prone to causing network delays and increasing hardware storage pressure. Meanwhile, the SDN architecture based on the OpenFlow protocol only supports the existing data forwarding rules. Therefore, new protocols cannot be added proactively as these protocols would make the OpenFlow protocol increasingly complicated because the device code is modified. To solve the problems above, the Open Networking Foundation (ONF) has proposed the protocol-independent forwarding (PIF) model(7) along with protocol-aware forwarding(8) and P4 technology,(9) which are applied to define the underlying original instruction set and to implement the packet handler. By using a uniform protocol-independent instruction set, the protocols can achieve data plane matching forwarding without perception, which completely decouples the data plane and control plane, as well as support for any new network protocols. With the continuous expansion of the data center network and the diversification of service types, the longest matching algorithm based on IP address used in the traditional routing mechanism leads to the complexity of the forwarding unit and the expansion of the routing table. The network performance is degraded in addition to the serious consumption of the network bandwidth.(10)

To summarize, in this paper, we present a new solution based on the protocol-oblivious forwarding technique, termed source routing protocol-oblivious forwarding (SRPOF). The data plane forwarding device does not have any perception of the routing protocol and forwarding process owing to a more streamlined source routing mechanism. Allied with the unification of the source address instruction tag, the strategy can support any forwarding protocol and packet data format. Furthermore, the network behavior is completely defined by the control plane that can solve the scalability problems for data centers effectively. We apply the self-developed and programmable network switch for data plane forwarding. Moreover, the flow-table-independent packet forwarding operation is realized by identifying the address instruction set controlled by the source side so as to ensure the hardware forwarding rate and improve the flexibility of the identification and forwarding rules.

2. Related Works

The data center network (DCN) architecture connects computing and storage resources for providing data access capabilities to users in the form of services. With the rise of XaaS service providers and the continuous virtualization of data centers, data center networks are developing rapidly. Traditional TCP/IP-based network systems cannot meet the commands of rapid internet development owing to problems such as complex management and low utilization. Therefore, operation and maintenance teams of the network architecture have begun to study high-performance and highly reliable network systems.(11) The introduction of SDN technology to build data center networks can effectively realize the automated management of programmable infrastructure.(12) The OpenFlow protocol, as its core technology,(13) separates the control plane and the data plane to achieve the flexible control of network traffic. However, its processing mechanism for storing and forwarding data information relies on the flow table. This approach is limited by the table flow space and energy consumption of the network switches. The scalability problems in the data plane are difficult to solve. In addition, the OpenFlow protocol cannot support the new protocols; thus, the only way to increase the support of the new protocol is to modify the switch hardware structure and device code, which makes the protocol increasingly complicated and limits the programmable features and flexibility in SDN architecture.(14) Huawei Company proposed a solution named protocol-oblivious forwarding (POF) technology, which is an improvement of the OpenFlow protocol to resolve the issues.(15–17) The data plane uses the triplet form such as {type, offset, length} to identify the protocol field, which does not need to grasp the format or content of the specific protocol. Meanwhile, complete decoupling between the control plane and the data plane is realized by using the unified protocol-independent instruction set so that any new protocol can be supported without modifying the data plane.

The source routing technology means that the network host adds routing information through the packet header to specify some or all of the forwarding devices that the data packet passes through, that is, the forwarding path is planned for it beforehand. There are many technological schemes of source routing based on the OpenFlow protocol.(18–20) The typical algorithm based on the source routing mechanism is the MPLS protocol.(21) Each MPLS label represents one-hop routing information. However, the limiting factor is that the number of MPLS labels supported by the data plane is limited. Thus, it is not applicable or scalable in large-scale networks. In Ref. 22, Ventre *et al.* adopted a mask that supports arbitrary bit lengths to implement field matching operations and Apple IPv6 fields to store per-hop routing information. However, the redundant field of the scheme increases the protocol length, that is, if the number of ports of a switch is 256, the 128-bit source IP address can only support 16-hop data forwarding operations. Thus, the network scalability problem is still difficult to solve. Sourcey,(23) as a typical source routing protocol in a data center network, provides topology discovery policies based on the network host. However, all hosts need to detect and master global network topology information. Such a solution causes a large number of redundant detection packets along with the serious reduction of network performance.

3. Routing Mechanism Based on POF in Data Center Networks

3.1 Data center network architecture based on POF

The POF protocol implements data matching and corresponding operations through two parameters: offset and length. The content of the related protocol is written by the controller. Thus, the switch does not need to append any entries or searching cost. The protocol-independent data forwarding process can be completed by applying the unified instruction set. The specific POF-based data center network architecture is shown in Fig. 1. The switch can complete the data processing procedure without interpreting the protocol content. That is, there is no need to upgrade the network switch or replace it with the new device when a new protocol is appended. The control plane controls the forwarding behavior of the network switch through the flow table of the extended OpenFlow protocol that expedites the network innovation proceedings.

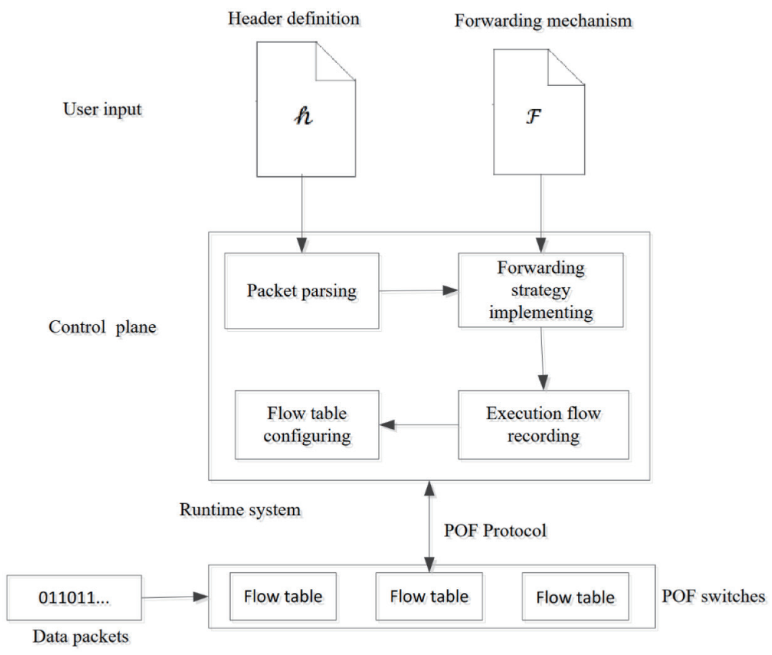

Fig. 1. POF forwarding model.

In the POF-based data center network architecture, the control plane mainly includes modules such as the MAC address learning, topology discovery, and communication engines; these modules are used to implement packet parsing and routing protocols. The processing of data packets includes the insertion, deletion, and modification of the bit string at a specific position of the message. Thus, the network forwarding device does not need to be aware of any knowledge related to a specific protocol; it only needs to perform a simple atomic operation function to efficiently perform stream instruction forwarding. All protocol-related processing flows are defined by the network controller. Users write a network algorithm to invoke the API provided by the POF network environment to obtain the network topology and data packets. The entire process has no further use for interaction with underlying concepts such as messages and flow tables.

3.2 Source routing and forwarding mechanism based on POF

The fundamental cause of the complexity of the source routing technology based on OpenFlow is that it cannot support a custom network protocol. In this paper, a source routing scheme based on the POF mechanism (SRPOF) is proposed. The source control idea is applied to make the forwarding hardware device have no perception of the network protocol and the forwarding procedure. The network behavior is completely defined and supported by the control plane that can support an arbitrary forwarding protocol and packet data format. The specific forwarding process design is shown in Fig. 2. The switch pre-allocates a local serial value (port index, PI) for all its ports. Thus, the sequence formed by sequentially combining the local port numbers of the switches on the communication path forms a source address instruction set. The instruction set uses a relative address coding method.

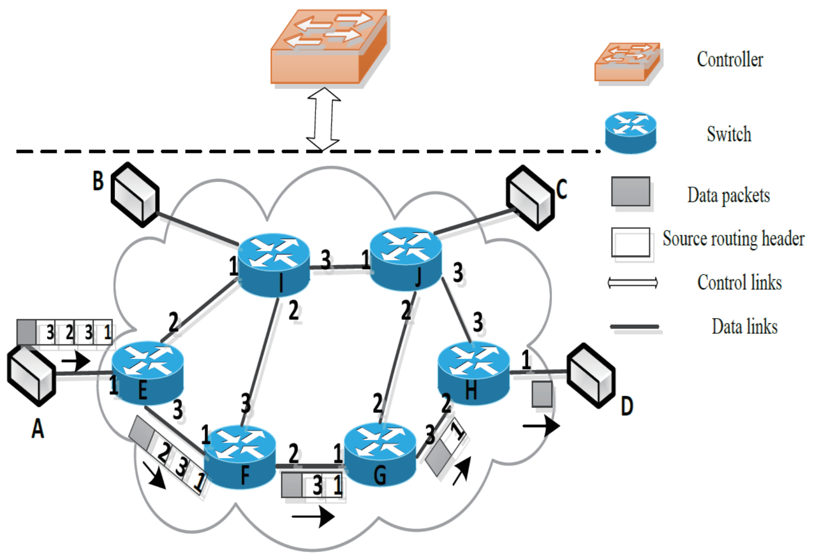

Fig. 2. (Color online) Source routing forwarding model.

In the data forwarding procedure above, it is assumed that the communication path calculated from the end system A to D is A → E → F → G → H → D, and the output port numbers correspond sequentially to E: 3, F: 2, G: 3, H: 1. Thus, the sequence obtained by combining these port numbers sequentially is the source address instruction set from A to D, where the specific numerical list is 3231, and we named it Path_{ad}. The above sequence can be expressed in binary code as {11, 10, 11, 1} and further integrated into {1110111}. This address instruction independently and completely identifies a communication path from A to D, and we regard Path_{ad} as routing information from A to D. The number of address bits of each switch is configured in accordance with the number of local ports. The source routing and forwarding scheme based on the POF technology proposed in this paper is the packet switching process with the above source address instruction set as the data forwarding address. The protocol format of the source routing scheme studied in this paper and the existing source routing protocol are shown in Fig. 3.

The data packets required for traditional source routing are shown in Fig. 3(a). The IPv4 Options field contains five complete IP addresses on the communication path. Figure 3(b) shows the data packet of the component when using OpenFlow source routing. Host A writes all the routing ports in different bits of the source address of the IPv6 protocol, and each hop route on the planned path matches different bits in the field. As shown in Fig. 3(c), in the source routing scheme designed in this study, host A does not need any redundant protocols or fields. The forwarding device on the path only needs to read the current address component in accordance with the pre-assigned local port number and to perform matching and forwarding data packets in accordance with the source address instruction set. The POF-based source routing scheme can support the characteristics of any protocol and implement a simple source routing mechanism.

3.3 Design of POF switch

To support the above source routing and forwarding strategy, we modify the hardware structure of the switch and append the parsing of the source address instruction set. The specific

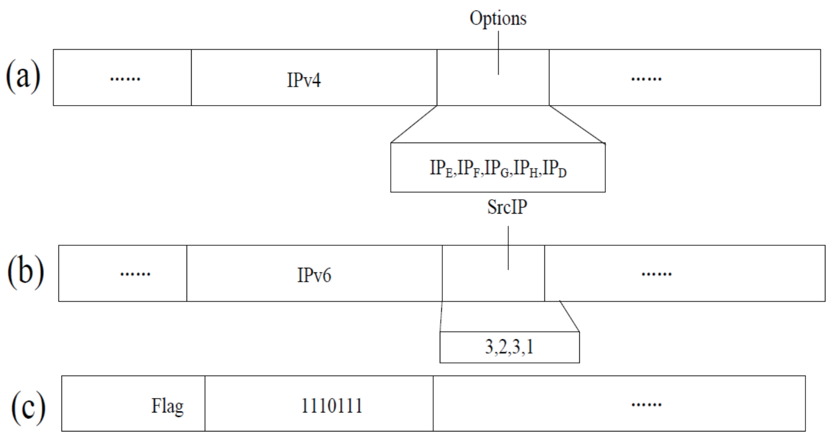

Fig. 3. Source routing packet with different formats.

workflow is shown in Fig. 4. After receiving a new data packet, we first analyze the ethernet header and source address data in accordance with the source routing address parsing protocol. When sending data packets on the output port, the ethernet header is encapsulated in the data in accordance with the address resolution protocol.

The above operation facilitates retaining the existing ethernet switches applied to the protocol-oblivious source routing and forwarding protocol. When a switch receives a data packet, it resolves first the packet header, which includes identifying the code of the frame preamble, and then the delimiter of the start and the end of the frame, and subsequently matches the MAC address of the destination node and packet type so as to transfer the data packet to the corresponding operation module. Considering these tasks, we design and implement the POF switch on the NetFPGA experimental platform, which can realize the data exchange function of four gigabit interfaces. The specific design of the hardware structure is shown in Fig. 5.

When the physical network port receives the data packet, it first performs a packet header parsing operation to extract the current corresponding to the PI component in the switching device, then the scheduling mechanism implements switching array scheduling in accordance with the current PI component prompt. After processing is completed, data packets are respectively entered into the buffering queue (0–3) implemented by the on-chip broadcast recognition access method (BRAM).(24) BRAM is an access protocol designed for regulating internode communication in either a wired or wireless channel-based network system. Thus, the switching device no longer uses static random access memory (SRAM) or off-chip dynamic random access memory (DRAM), effectively reducing the resource overhead of off-chip storage and the cost.

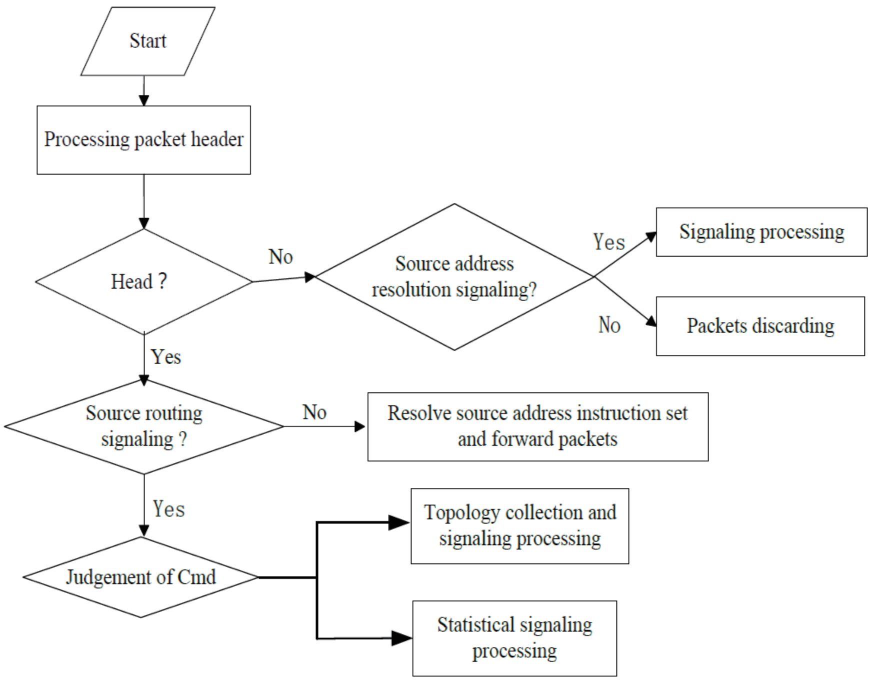
Start
Processing packet header
Head ?
No
Source address resolution signaling?
Yes
Signaling processing
No
Packets discarding
Yes
Source routing signaling ?
No
Resolve source address instruction set and forward packets
Yes
Judgement of Cmd
Topology collection and signaling processing
Statistical signaling processing

Fig. 4. Switch forwarding procedure.

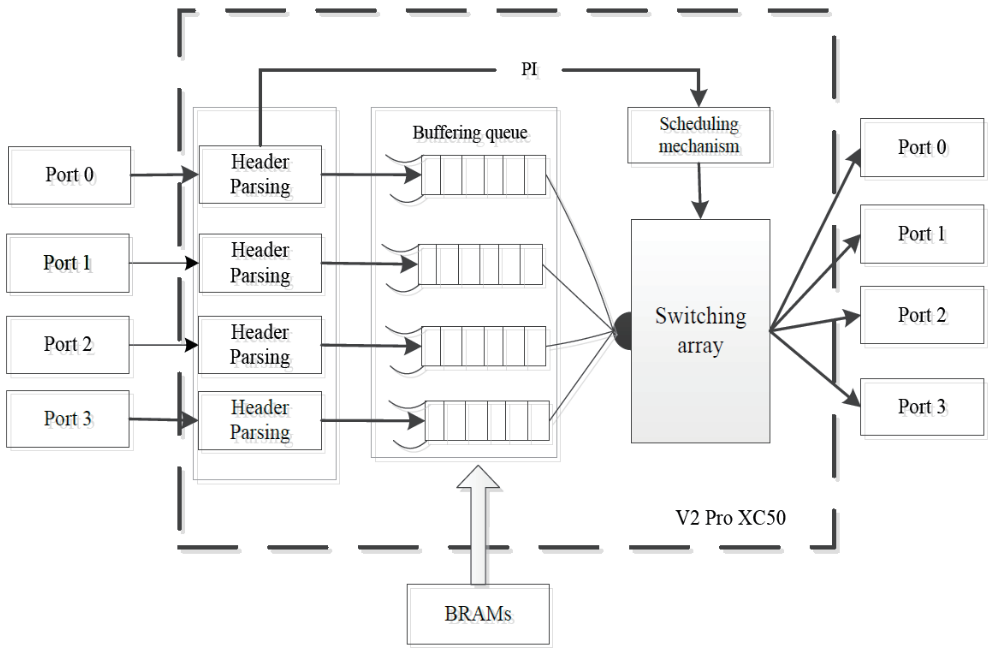

Fig. 5. Design of POF switch.

4. Experiment and Verification

We implement Mininet(25) to design and deploy a data center network architecture based on POF and source routing mechanisms. The network is deployed by using a POF controller(26) and a self-developed programmable POF switch. From the results of the experiment, we can mainly verify the following three points: (1) the proposed POF-based source routing scheme can effectively reduce the number of data plane flow entries; (2) the network redundancy for detecting the packet has been considerably decreased during the flow table update phase; (3) the resource consumption and manufacturing costs for switch hardware can be reduced significantly.

4.1 Experimental statistics for the size of flow table

In the test described in this section, we build the Fat-tree(27) network topology and implement the SRPOF. The script is coded for the network host to append the source routing protocol header to the data packet. We set k = 4 (pods) for counting the number of flow entries of each switch in each layer of the network. Meanwhile, we also compare the number of flow entries generated by paired unicast routing with OpenFlow v1.0 protocol.

The number of required flow entries at each level is shown in Table 1. Table 1 shows that the SRPOF scheme can effectively reduce the number of flow entries. Consequently, the SRPOF only needs to match the fields of the source address header, while the paired unicast protocol demands to match multiple protocol fields such as the source address, destination address, and MAC address.

Table 1
Number of flow table items in each scheme.

	SRPOF	OpenFlow v1.0
Core	4	82
Aggregation	4	48
Edge	4	56

The SRPOF strategy proposed herein can effectively reduce the data processing time by simplifying the forwarding processing logic and the number of flow table entries. The data plane is satisfactory to complete the forwarding operation of the data packet through the unified source address instruction label, effectively simplifying the data processing logic and reducing the size of the flow table and thus further improving the network utilization efficiency.

4.2 Routing comparison

In this section, we analyze the statistics of the number of probe packets consumed in different network sizes under the conditions of the same routing information and network topology. It is assumed that the data flows are transmitted through N network nodes.

In the OpenFlow network environment, the number of signals required for the first OpenFlow switch is three during the routing procedure: a PACKET-IN routing request signal sent to the controller, a Modify State Message signal returned by the routing result, and a PACKET-OUT signal sent to itself to trigger the data transmission process. The other $n - 1$ forwarding devices each require one Modify State Message signal. Thus, the total number of probe packets required is $n + 2$. The number of switches is linearly proportional to the number of hops N on the communication path.

In the SRPOF scheme, the data stream transmission routine is determined by the source instruction tag. The source routing address is obtained by the edge gateway which sends a PACKET-IN routing request to the controller and accepts the PACKET-OUT message. The intermediate node on the communication path can accomplish the forwarding procedure without adding a flow entry. Thus, the number of probe packets required is four. The specific trends for the number of probe packets when the number of network nodes changes in the two routing mechanisms are shown in Fig. 6. It can be seen from the figure that as the number of switches increases, the SRPOF scheme proposed in the paper generates far fewer probe packets than the OpenFlow protocol in the same environment. Meanwhile, the routing and flow table updating procedure has been significantly simplified.

4.3 Switch performance test

NetFPGA is a reconfigurable and low-cost hardware platform developed by Stanford University for network researchers. Researchers can apply the platform to build diverse network prototypes and stable network test environments.[28] The core part of the NetFPGA hardware mainly consists of two FPGAs running synchronously at a clock frequency of 125 MHz, where

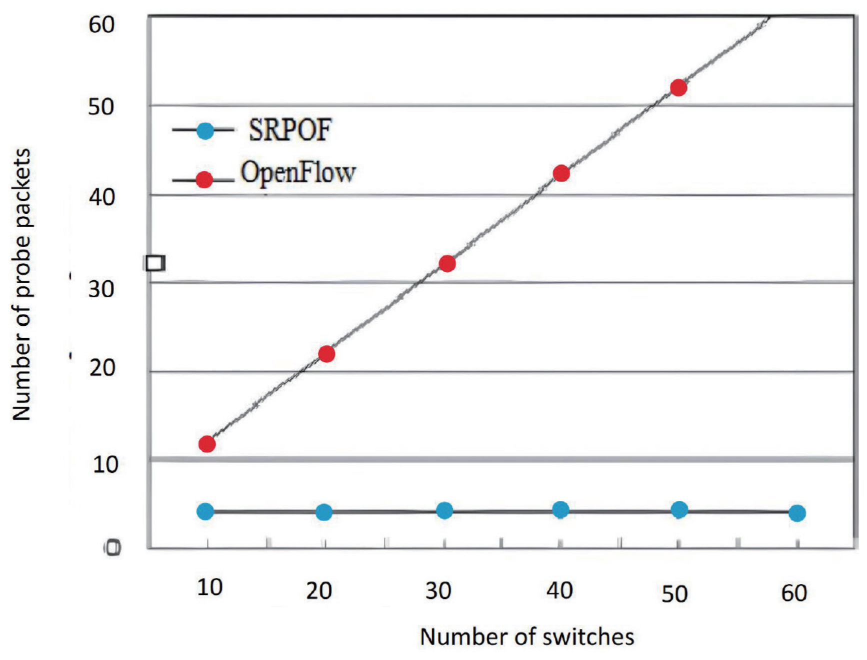

Fig. 6. (Color online) Experimental results on the number of network signals.

one has a Virtex-II Pro series chip added as the main processing chip and the other has a Spartan II series XC2S200-FG456C chip. The onboard resources include the SRAM and a general-purpose JTAG test port. In Ref. 29, Chu *et al.* proposed the implementation of OpenFlow switches on the NetFPGA platform, which utilizes two flow table memory structures including the accurate table lookup strategy based on SRAM and the TCAM-based wildcard table lookup. Meanwhile, NetFPGA developers have also devised two sets of hardware architecture solutions for gigabit Ethernet switches and IPv4 routers. In this section, we analyze the hardware resource consumption statistics of the designed POF switch, including the parameters Slices, LUTs, and BRAMs, and compare them with reference designs such as those of ethernet switches, IPV4 routers, and OpenFlow switches. The specific hardware resource consumption results are shown in Table 2. The POF switch calculates the resources consumed by all modules, while for the other types of switches, only the resources consumed by the lookup module of the output port are considered.

It can be concluded that the POF switch designed in this paper to support the source routing mechanism has no further need for off-chip storage SRAM, while the OpenFlow switch uses SRAM as the flow table memory and requires PCI bus support, whereby each new flow entry consumes at least 12 μs of processing time, severely hindering the rate of flow table updating. At the same time, the POF packet buffer is only 40 kB, which is realized by using an off-chip BRAM, which is different from the off-chip DRAM packet buffer used by OpenFlow switches. Finally, the consumption of hardware resources, such as LUT and DFF, is about 63% less than that of the OpenFlow switches.

Table 2
Statistics of hardware resource consumption for switches based on the NetFPGA.

	POF Switch	Ethernet Switch	IPv4 Router	OpenFlow Switch
Slices	1896	—	—	5878
LUTs	2680	3920	14086	10368
DFFs	2026	1174	3358	6671
BRAMs	26	96	15	14
Off-chip storage	None	—	DRAM, SRAM	DRAM, SRAM

5. Conclusions

We combined POF and source routing mechanism and proposed two key technologies applied in data center networks. Firstly, we put forward the SRPOF scheme based on simple instructions for data forwarding. It supports arbitrary forwarding protocols and data format through the unification of source address instruction labels, effectively reducing the redundancy of network probe packets and the size of flow entries. Secondly, we developed low-cost and scalable POF switches that can implement flow-table-independent packet forwarding operations and effectively improve the flexibility of forwarding rules. Finally, a data center network environment was built on the Mininet platform for testing and comparing related technologies. The experimental results showed that the proposed solution can significantly improve the utilization of the data center network along with providing good flexibility and scalability. The future research contents and directions will be to explore how to optimize the network topology management mechanism and improve the performance of the POF controller.

References

1 D. Kandris, C. Nakas, D. Vomvas, and G. Koulouras: Appl. Syst. Innov. **3** (2020) 14. https://doi.org/10.3390/asi3010014
2 W. Lin, Y. Wu, and N. Jiao: Mob. Inf. Syst. **2022** (2022) 9139257. https://doi.org/10.1155/2022/9139257
3 L. A. Barroso and J. Clidaras: The datacenter as a computer: An introduction to the design of warehouse-scale machines, (Springer Nature, 2022).
4 M. Karakus and A. Durresi: Comput. Networks **112** (2017) 279. https://doi.org/10.1016/j.comnet.2016.11.017
5 S. Ahmad and A. Hussain: J. Netw. Syst. Manage. **29** (2021) 1. https://doi.org/10.1007/s10922-020-09575-4
6 M. Shahbaz, S. Choi, B. Pfaff, C. Kim, N. Feamster, N. McKeown, and J. Rexford: Proc. 2016 ACM SIGCOMM Conf. (2016) 525–538.
7 D. Hu, S. Li, N. Xue, C. Chen, S. Ma, W. Fang, and Z. Zhu: Proc. 2015 IEEE Global Communications Conf. (IEEE, 2015) 1–6.
8 C. Yoon, T. Park, S. Lee, H. Kang, S. Shin, and Z. Zhang: Comput. Networks **85** (2015) 19. https://doi.org/10.1016/j.comnet.2015.05.005
9 B. Dai, G. Xu, B. Huang, P. Qin, and Y. Xu: J. Netw. Comput. Appl. **94** (2017) 33. https://doi.org/10.1016/j.jnca.2017.07.004
10 O. Rottenstreich and J. Tapolcai: IEEE/ACM Trans. Networking **25** (2017) 864. https://doi.org/10.1109/TNET.2016.2611482
11 C. C. Udeze, K. C. Okafor, C. C. Okezie, I. O. Okeke, and C. G. Ezekwe: Proc. 2014 IEEE 6th Int. Conf. Adaptive Science and Technology (IEEE, 2014) 1–12.
12 Y. Zhao, R. He, H. Chen, J. Zhang, Y. Ji, H. Zheng, Y. Lin, and X. Wang: Opt. Express **22** (2014) 9538. https://doi.org/10.1364/OE.22.009538
13 K. Sharma and R. N. Yadav: Comput. Networks **175** (2020) 107235. https://doi.org/10.1016/j.comnet.2020.107235

14 S. A. Rofie, I. Ramli, K. N. Redzwan, S. M. Hassan, and M. S. B. Ibrahim: Adv. Sci. Lett. **24** (2018) 1210. https://doi.org/10.1166/asl.2018.10718
15 H. Song: Proc. 2013 ACM SIGCOMM Conf. (2013) 127–132.
16 X. Tang, X. Zeng, and L. Song: IEEE Trans. Network and Service Manage. **20** (2022) 578. https://doi.org/10.1109/TNSM.2022.3207227
17 Q. Zhang, A. Ansari, and Z. Zhu: IEEE Internet of Things J. **10** (2022) 7303. https://doi.org/10.1109/JIOT.2022.3228796
18 R. M. Ramos, M. Martinello, and C. E. Rothenberg: Proc. 2013 IEEE Local Computer Networks (IEEE, 2013) 606–613.
19 A. Ishimori, E. Cerqueira, and A. Abelém: Proc. 2017 IFIP/IEEE Symp. Integrated Network and Service Manage. (IEEE, 2017) 923–928.
20 S. Li, K. Han, N. Ansari, Q. Bao, D. Hu, J. Liu, S. Yu, and Z. Zhu: IEEE Trans. Network and Service Manage. **15** (2017) 275. https://doi.org/10.1109/TNSM.2017.2766159
21 L. Huang, Q. Shen, F. Zhou, W. Shao, and X. Cui: Trans. Emerging Telecommun. Technol. **29** (2018) e3286. https://doi.org/10.1002/ett.3286
22 P. L. Ventre, M. M. Tajiki, S. Salsano, and C. Filsfils: IEEE Trans. Network Serv. Manage. **15** (2018) 1378. https://doi.org/10.1109/TNSM.2018.2876251
23 Y. Zhang, D. Li, Z. Sun, F. Zhao, J. Su, and X. Lu: IEEE Trans. Cloud Comput. **6** (2018) 464. https://doi.org/10.1109/TCC.2015.2440242
24 I. Chlamtac, W. R. Franta, and K. D. Levin: IEEE Trans. Commun. **27** (1979) 1183. https://doi.org/10.1109/TCOM.1979.1094529
25 Q. Li, X. Zou, Q. Huang, J. Zheng, and P. P. Lee: IEEE Trans. Dependable Secure Comput. **16** (2019) 915. https://doi.org/10.1109/TDSC.2018.2810880
26 P. S. Priya and B. Bandyopadhyay: Eur. J. Control **33** (2017) 52. https://doi.org/10.1016/j.ejcon.2016.08.001
27 Z. Guo, J. Duan, and Y. Yang: Proc. 2013 IEEE 27th Int. Symp. Parallel and Distributed Processing (IEEE, 2013) 589–600.
28 N. Zilberman, Y. Audzevich, G. A. Covington, and A. W. Moore: IEEE Micro **34** (2014) 32. https://doi.org/10.1109/MM.2014.61
29 T. W. Chu, C. A. Shen, C. W. Wu: Multimedia Tools Appl. **77** (2017) 1. https://doi.org/10.1007/s11042-017-4806-7