BIG DATA MINING AND ANALYTICS
ISSN 2096-0654 06/10 pp55–71
Volume 6, Number 1, March 2023
DOI: 10.26599/BDMA.2022.9020018

Intelligent Segment Routing: Toward Load Balancing with Limited Control Overheads

Shu Yang, Ruiyu Chen, Laizhong Cui*, and Xiaolei Chang

Abstract: Segment routing has been a novel architecture for traffic engineering in recent years. However, segment routing brings control overheads, i.e., additional packets headers should be inserted. The overheads can greatly reduce the forwarding efficiency for a large network, when segment headers become too long. To achieve the best of two targets, we propose the intelligent routing scheme for traffic engineering (IRTE), which can achieve load balancing with limited control overheads. To achieve optimal performance, we first formulate the problem as a mapping problem that maps different flows to key diversion points. Second, we prove the problem is nondeterministic polynomial (NP)-hard by reducing it to a k-dense subgraph problem. To solve this problem, we develop an ant colony optimization algorithm as improved ant colony optimization (IACO), which is widely used in network optimization problems. We also design the load balancing algorithm with diversion routing (LBA-DR), and analyze its theoretical performance. Finally, we evaluate the IRTE in different real-world topologies, and the results show that the IRTE outperforms traditional algorithms, e.g., the maximum bandwidth is 24.6% lower than that of traditional algorithms when evaluating on BellCanada topology.

Key words: traffic engineering; segment routing; bandwidth load balancing; ant colony optimization

1 Introduction

Traffic engineering (TE) has always attracted much research attention. Traditional TE concentrated on IP routing protocols[1], routing optimization problems[2], overlaying in an IP network[3], etc. Most of these studies were conducted in traditional IP networks. With the advent of the software defined network (SDN), researchers began to focus on TE issues in the SDN, including traffic splitting and SDN protocol design[4]. The SDN can help us achieve efficient network management, which can solve massive TE issues that are difficult to realize in traditional networks. However, the SDN faces great challenges, e.g., scalability issues that limit its application scope[5, 6]. In addition, segment routing (SR) has advantages in network structure that can help solve these problems in the SDN. Therefore, many scholars began to explore the possibility of combining the SDN with segment routing[7].

Segment routing is a novel network architecture that has realized further control of SDN in recent years. SR can achieve compatibility with the SDN, and it has become an implementation solution to some TE problems in the SDN[8]. SR can realize directional data transmission from the source node to the destination node[9]. However, the control overhead in SR is also unlimited in many different scenarios, which will causes inefficient data transmission[10]. Meanwhile, existing solutions ignore the problem of an overlarge packet header in SR[11]. Therefore, a routing scheme must be explored to achieve load balancing and consider the limited packet length in SR.

Control overhead has an important problem in

• Shu Yang, Ruiyu Chen, and Laizhong Cui are with the College of Computer Science and Software Engineering, Shenzhen University, Shenzhen 518000, China. E-mail: yang.shu@szu.edu.cn; 2070276056@email.szu.edu.cn; cuilz@szu.edu.cn.
• Xiaolei Chang is with the Tsinghua Shenzhen International Graduate School, Tsinghua University, Shenzhen 518071, China. E-mail: changxl@mail.tsinghua.edu.cn.
∗To whom correspondence should be addressed.
Manuscript received: 2021-12-18; revised: 2022-06-22; accepted: 2022-06-24

© The author(s) 2023. The articles published in this open access journal are distributed under the terms of the Creative Commons Attribution 4.0 International License (http://creativecommons.org/licenses/by/4.0/).

TE. Appropriate schemes for controlling overhead can optimize network transmission performance. In recent years, with the development of new network paradigms, i.e., the SDN, SR, and blockchain, control overhead optimization has been used in different frameworks[12, 13]. However, control overhead optimization is neglected in the SDN and SR. An unlimited control overhead causes an overlarge SR packet length, and decreases data transmission efficiency. The length of an SR packet header increases as the routing length grows, which deteriorates the control overhead situation in SR. Researchers did not optimize network performance from the aspect of limited control overhead when solving the issue of bandwidth load balancing in SR[14]. Therefore, the TE scheme in SR that combines bandwidth load balancing and limited control overhead merits deeper exploration.

In this paper, we propose an intelligent routing scheme for TE (IRTE) in an SR environment. The IRTE not only achieves bandwidth load balancing in an SR environment, but also considers packets lengths for limited control overhead. We propose the diversion routing method to achieve our bandwidth load balancing target. We design the IRTE architecture based on SR controller and deploy diversion routing capable nodes. The controller collects the network status and generates the information matrix in each time interval. After receiving the data transmission demand from source hosts, the controller computes the diversion routing paths for them according to load balancing algorithms and the network situation. We design two novel load balancing and overload optimization algorithms for the IRTE, including the load balancing algorithm with diversion routing (LBA-DR) and improved ant colony optimization (IACO). The LBA-DR generates the diversion routing path for each source node through linear programming and the randomized rounding method. IACO analyzes this matrix and provides a diversion routing path, which is generally different from the traditional shortest routing path. On the one hand, this diversion routing path from these two algorithms considers the bandwidth load balancing target, and decreases the maximum bandwidth use in the network. On the other hand, it controls the SR packet header length, and avoids the forwarding overload situation. Later, the source establishes data flow according to this diversion routing path and transmits data to the destination. Finally, the nodes on this routing report the latest network status to the SR controller to update the network information matrix.

We implement our IRTE architecture, and evaluate it under real-world network topologies. We compare the designed algorithms with traditional algorithms. We compare their performances in six typical topologies with different numbers of flows. We record the control overhead of each algorithm and verify that our simulation is a test-bed experiment. We analyze the changing trend of these two algorithms and compare their load balancing performances. We also explore the relationship between algorithm performance and the number of nodes, degrees, and parameters of IACO. The experimental results show that in different topologies, the IRTE realizes the bandwidth load balancing objective. The maximum bandwidth use in most networks using the LBA-DR or IACO is reduced by 2.45% to 24.6% compared with previous algorithms. With increasing flow numbers, IACO achieves a performance advantage of 1.3% to 22.5%, while the LBA-DR reaches 2.1% to 17%. Additionally, with a change in α and β in IACO, the load balancing gap between IACO and a traditional algorithm improves from 14.5% to 22.7% and from 9.34% to 29.3%, respectively. Meanwhile, the packet header length is a significant constraint in each experiment.

The main contributions are as follows.

- We design the diversion routing architecture IRTE in the SR environment for TE. We complete the function and application of the SR controller and diversion routing capable nodes in the network. The network status is timely recorded in the controller at each time interval.
- We formulate the bandwidth load balancing problem in the SR environment, and design the LBA-DR algorithm and IACO algorithm to realize the load balancing objective with the help of SR. We define the forwarding overload situation in the SR environment, and control the SR packet header length to avoid forwarding overload.
- We implement and compare our IRTE architecture in different topologies. The experimental results show that the LBA-DR and IACO of the IRTE significantly achieve bandwidth load balancing and avoid forwarding overload in data transmission.

The remaining of paper is organized as follows. Section 2 introduces the related work. Section 3 presents the IRTE architecture and the functions and details of each entity. Section 4 discusses our problem formulation, and proves the time complexity of our problem. Section 5 introduces the two algorithms for improving control

overheads in the IRTE. The experimental results are presented in Section 6. Finally, Section 7 gives the conclusion.

2 Related Work

2.1 Traffic engineering

TE has been proposed for several decades. In Ref. [15], the authors discussed TE with multi-protocol label switching (MPLS). Fortz et al.[16] introduced traditional TE protocols such as the open shortest path first (OSPF) and system-intermediate system. In the meantime, Elwalid et al.[17] introduced an adaptive algorithm named MATE to optimize TE in an MPLS environment. The authors of Ref. [18] introduced a flow controller to configuration commands in an ISP network. Kandual et al.[19] presented a novel protocol named TeXCP that realized online TE and load balance.

With the development of the SDN, many researchers concentrate on the TE of the SDN. In Ref. [20], the authors showed that TE had methods for addressing the problems in the integrated architecture of the SDN and the exiting network. The authors of Ref. [21] introduced the primary challenges of TE, including protocols, traffic control, and management in the SDN. In Ref. [22], the researchers improved the performance of a flow classifier for heterogeneous traffic. In Ref. [4], the authors summarized several contributions about TE in the SDN, including SDN protocols for TE and novel SDN architecture such as the path computation element. In addition, Chiesa et al.[23] demonstrated the shortcomings of equal-cost-multipath (ECMP) because of the large flow and presented a novel algorithm to improve the performance of ECMP. Meanwhile, Bahnasse et al.[24] contributed to an SDN architecture aiming at improving the quality of service and bandwidth allocation for TE. The authors of Ref. [25] introduced a novel traffic monitoring framework via spark streaming. In Ref. [26], researchers presented several theories and mathematical demonstrations about node-constrained TE. Thus, we discover several possible improvement issues of TE by optimizing bandwidth use.

2.2 Segment routing

SR is a recently proposed novel transmission network architecture. The researchers of Refs. [27, 28] introduced the concepts and construction of SR, and improved the traffic matrix algorithm. Authors in Refs. [8, 9] presented several surveys about SR architecture and its details. In Ref. [29], the TE advantages of SR are shown through reasonable deployments. Hereafter, many studies of TE on SR have followed. For example, the authors of Ref. [30] presented an optimized routing algorithm for bandwidth balancing and SR header control. The authors of Ref. [14] introduced an incremental method for optimizing link utilization for a shifty network environment. In Ref. [31] the authors considered decision load balancing in IPv6 and proposed a novel framework for applications to reduce network overhead. Additionally, the authors of Ref. [32] obtained an optimization solution for topology scalability through the column generation method and uesd the SR adjacency segment adequately. Moreover, the researchers of Ref. [33] presented an effective failure resiliency framework and controlled the number of tunnels for TE in SR. These articles reflect the potential to improve network performance for TE in SR. Thus, the issue of network resource scheduling motivates us to design an effective solution for optimizing link bandwidth use.

2.3 Control overhead optimization

How to realize overhead control optimization in data transmission is a potential research area in TE. In recent years, researchers have designed many methods to achieve control overhead. The authors of Ref. [34] proposed a data structure to control the synchronization packet length. Wang et al.[35] introduced an analysis scheme for balancing run-time overhead. Freschi and Lattanzi[36] presented a novel architecture to explore the mathematical relationship between packet length and node reliability. Meanwhile, the authors of Ref. [37] proposed a novel rebroadcast strategy to improve the performance of routing overhead. In Ref. [38], the authors designed a data aggregation algorithm to improve the overhead situation by reducing the packet loss rate and latency. Additionally, the authors of Ref. [39] optimized the technique of hybrid automatic repeat requests for packet control overhead. However, these works ignore the possibility of controlling packet header length for control overhead. Therefore, we must consider the limited control overhead for bandwidth load balancing in SR environment.

3 IRTE Design

In this section, we introduce our overall system design and the details of our architecture. We first propose an overview of our framework, and discuss instances in our framework. Then we introduce the details of the entities

in our system.

3.1 Overall architecture

In this paper, we propose IRTE as an SR environment.

Figure 1 shows the total architecture of the IRTE. The IRTE mainly comprises four components: routers, hosts, links, and an SR controller. Generally, the source and destination of each flow are hosts. Routers and links form paths between hosts. There are two types of hosts, deployed diversion capable hosts and traditional routing hosts. Diversion capable hosts can select the diversion routing path or the shortest path to transmit packets, while traditional routing hosts can only use the latter path. As the major module of the IRTE, the SR controller collects the bandwidth status of each link and generates and maintains a bandwidth utilization matrix.

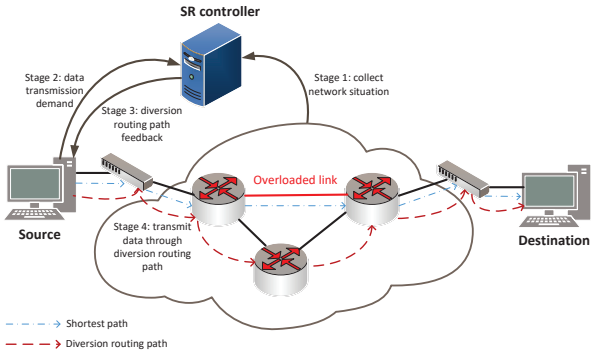


Fig. 1 IRTE mechanism.

In Fig. 2, we use a case study to illustrate the data transmission mechanism of diversion routing. Assume that Host1 requires communication with Host5 to establish data flow. We define the shortest routing path from Host1 to Host5 as $P_s = \{\text{Host1, R1, R4, R7, R9, R11, Host5}\}$. Additionally, we provide two alternative routing paths for this flow, including $P_1 = \{\text{Host1, R1, R3, R7, R9, R11, Host5}\}$ and $P_2 = \{\text{Host1, R1, R2, R4, R7, R10, R9, R11, Host5}\}$. The routers represented in red in the SR header are the active segments, and it will direct the current router to transfer packets to the next segment router. We use a diversion routing label to represent the segment in the packet header for each path. We can see that the number of diversion routing labels for P_1 and P_2 is 2 and 4, respectively. Therefore, different diversion routing paths bring different numbers of diversion routing labels. This effect will cause differences in router forwarding load performance. Consequently, a limitation of packet length must be designed to avoid packet forwarding overload for our diversion routing method.

In Fig. 1, the IRTE process can be described as follows.

Stage 1. Initially, the SR controller collects the current network situation such as the overloaded link and congestion information. Via this information, the SR controller generates the network status information of

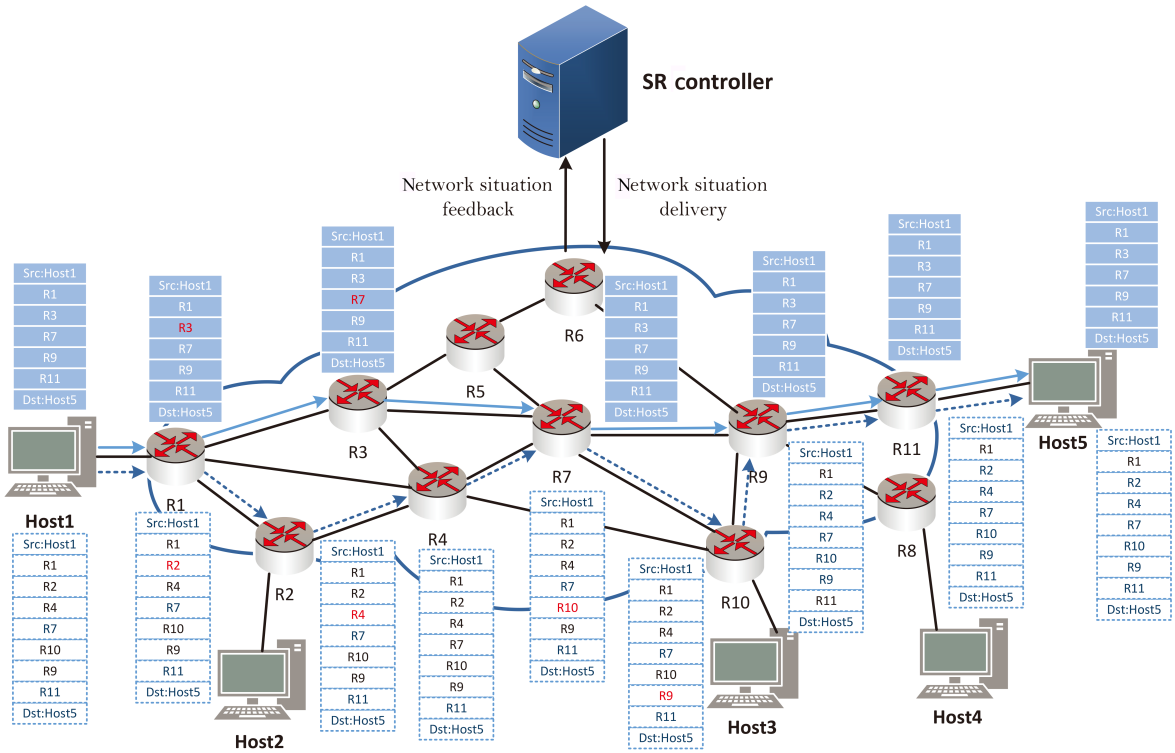


Fig. 2 Case study.

each link in this network and saves it in memory space.

Stage 2. When one diversion capable source host is prepared to transmit data to another host, it sends a request to the SR controller to obtain the diversion routing path of the transmission.

Stage 3. After receiving data transmission demand, the controller analyzes and computes the diversion routing path for source through network status information and load balancing algorithms of the IRTE, and delivers this path to source.

Stage 4. According to the diversion routing path from the controller, the source host establishes data flow to the destination host through this diversion routing path to avoid transferring data on overloaded links and realizes the load balancing objective.

In general, the advantages of the IRTE can be described as follows.

• **Timely updating of network status.** In this framework, the SR controller periodically collects bandwidth use. Therefore, the controller records any change in network status and timely updates the status matrix. Exploring an optimal routing path benefits new transmission requirements.

• **Overall balancing bandwidth use.** According to the intelligence of the IRTE routing selection, the controller can collect network information and transfer it to required hosts. Hosts use the IRTE to explore the diversion routing path and achieve the bandwidth load balancing objective.

• **Optimizing packet lengths.** Except for reduced maximum bandwidth use, we consider the adverse influence of unlimited packet length due to increasing diversion routing labels. Therefore, controller determines the maximum number of diversion routing labels as a significant threshold to optimize packet lengths and reduce transmission overhead.

3.2 SR controller

The SR controller is the kernel of the IRTE in our framework, and it belongs to the control plane of SR. In this paper, the SR controller collects the network status of links, timely updates the network information and assists hosts in discovering the optimal routing path. The major functions of the SR controller are as follows.

• **Status collection.** The controller periodically collects some necessary network status, such as bandwidth use of links, diversion routing labels, and efficiency of data transmission. Meanwhile, it timely updates status matrix of the network to satisfy the load balancing objective at any time.

• **Packet length control.** The controller limits the packet length to avoid the overload of data forwarding while using the diversion routing path. It can change the threshold of forwarding labels according to the network status it collected previously.

• **Strategy optimization.** Except for controlling labels, the controller can help hosts to determine and optimize the diversion routing path to achieve the load balancing objective. It improves the total performance of the network.

According to these functions, the controller can coordinate the bandwidth use of links and SR forwarding labels of packets and optimize the total load balancing situation of this network.

3.3 Hosts and routers

Hosts play the role of producing the data transmission requirement in this network. First, each host and router detects the network status of links connected to them, and transfers this status to the SR controller. Then, if the diversion capable hosts have the demand of constructing data flow, this demand will require the controller to generate a diversion routing path for the flow. When receiving the path from the controller, the source host establishes the flow according to this path. After establishing the data flow, the hosts and routers of this path will also transfer the latest network status to the SR controller, to maintain the newest information matrix in the network controller.

4 Problem Formulation

4.1 Problem description

We show the notation list in Table 1. Let $G = (V, E)$ be a network, where V is the set of nodes and E is the set of links. Assuming that n traffic flows in our network, let $F = \{f_1, f_2, \ldots, f_n\}$ denote the set of traffic flows, and $d(f_i)$ denotes the traffic demand of f_i. For each network flow f_i, $s(f_i)$ and $d(f_i)$ represent the source and destination of traffic flow f_i, respectively. For each f_i, the source $s(f_i)$ will probably travel on the shortest path according to the traditional routing operation, and we call it S_i. We use $S_i = \{s(f_i), x^1_{s_i}, x^2_{s_i}, \ldots, x^k_{s_i}, \ldots, d(f_i)\}$ to denote the shortest path S_i for f_i. In addition, we use $Div_i = \{s(f_i), x^1_{d_i}, x^2_{d_i}, \ldots, x^k_{d_i}, \ldots, d(f_i)\}$ to denote the diversion routing path for flow f_i. Let μ_i denote the node number of Div_i. Diversion routing path Div_i

Table 1 Notation list.

Notation	Meaning
$G=(V,E)$	Network topology graph
F	Set of traffic flows
$s(f_j), d(f_j)$	Source and destination of flow f_j
S_j	Shortest path of flow f_j
U_i	Use of link e_i
C_i	Capacity of link e_i
B_i	Occupied bandwidth of link e_i
Drn_i	Number of DR labels of flow f_i
Map	Operation of flow f_i
$d(f_i)$	Traffic demand of flow f_i
z_x^i	Path selection parameter of f_i
μ	Node number of Div_i
λ	Comparable parameter
δ	Constraint of Drn

is a pre-defined path for $s(f_i)$ and it normally differs from the shortest path S_i to achieve our load balancing target. Because of the limitation of network bandwidth and storage, let C_i denote the capacity of link e_i, U_i denote the link utilization of link e_i, and B_i denote the occupied bandwidth of e_i. Obviously $B_i \leqslant C_i$. Thus, we can calculate U_i as

$$U_i = \frac{B_i}{C_i} \tag{1}$$

Two routing operations are possible for each flow f_i: ordinary routing operation $Map^1(f_i)$ and diversion routing operation $Map^2(f_i)$. $Map^1(f_i)$ and $Map^2(f_i)$ denote that $s(f_i)$ selects the traditional shortest routing path S_i or diversion routing path Div_i for f_i. We formulate these two operations as follows.

$$Map^1(f_i) = S_i, \forall f_i \in F \tag{2}$$

$$Map^2(f_i) = Div_i, \forall f_i \in F \tag{3}$$

If $s(f_i)$ is diversion routing capable, considering the load balancing of the network, $s(f_i)$ can select $Map^1(f_i)$ or $Map^2(f_i)$ to transmit flow f_i. Otherwise, if $s(f_i)$ is not diversion routing capable, $s(f_i)$ will only perform $Map^1(f_i)$ and select the traditional shortest routing path. We use μ_i to denote the node number of Div_i.

Meanwhile, multiple feasible routing paths may be possible for the diversion routing capable nodes. In our network architecture, we assume that each diversion routing capable node only chooses one diversion routing path to establish a data connection. Let $P(f_i) = \{p_1, p_2, \ldots, p_m\}$ represent the feasible diversion routing path set for flow f_i. We use the parameter z_x^i to denote which diversion routing path the source node selects. z_x^i can be formulated as follows.

$$\forall f_i \in F,\ z_x^i = \begin{cases} 1, \text{if } f_i \text{ selects } p_x \in P(f_i) \\ 0, \text{otherwise} \end{cases} \tag{4}$$

Obviously, because only one routing path is chosen, z_x^i satisfies the following condition.

$$\sum_{x=1}^{m} z_x^i = 1 \tag{5}$$

As for deployed diversion routing capable node v, to transmit flow f_i through Div_i, v will add diversion routing labels in the SR header to direct packets of flow f_i to the given path Div_i. We use Drn_i to denote the number of diversion routing labels of each packet from flow f_i. To calculate Drn_i, let $Snh(f_i,k)$ and $Dnh(f_i,k)$ represent the k-th hop of routing operation $Map^1(f_i)$ and $Map^2(f_i)$, so $k \geqslant 1$ and $k \leqslant \mu_i$. We denote λ_i as a comparable parameter of $Snh(f_i,k)$ and $Dnh(f_i,k)$. Thus, Drn_i can be formulated as follows.

$$\lambda_j = \begin{cases} 1,\ \Big(\forall Dnh(f_i,j) \notin S_i\Big) \text{ or} \\ \quad \Big(\exists Dnh(f_i,j) = Snh(f_i,k), \\ \quad Dnh(f_i,j+1) \neq Snh(f_i,k+1)\Big) \\ 0,\ \text{otherwise} \end{cases} \tag{6}$$

$$Drn_i = \sum_{j=2}^{\mu_i - 1} \lambda_j \tag{7}$$

Additionally, we use δ as the constraint of the maximum Drn of the network. Thus, our load balancing objective can be formulated as follows.

Objective 1. Given $G=(V,E)$ and ω, find an operation $Map(f_i)$ for $\forall f_i \in F$,

$$\text{Min max } U \tag{8}$$

$$\text{s.t.} \begin{cases} Drn_i \leqslant \delta, \forall f_i \in F \\ \sum\limits_{x=1}^{m} z_x^i = 1, \forall f_i \in F \\ \sum\limits_{f \in F} \sum\limits_{p \in P, e \in E} z d(f) \leqslant U_{\max} C_{\max}, \forall e \in E \\ z_x^i \in \{0,1\}, \forall f_i \in F \end{cases} \tag{9}$$

The first inequality denotes that the Drn of each f cannot exceed a network parameter δ. The second inequality denotes that each f only selects one routing path for data transmission. The third inequality denotes that each link load of e will not exceed the maximum link load of the network.

4.2 Problem complexity

To solve our load balancing objective, we need to determine the complexity of our problem. However, no

previous proof exists of the relative complexity based on SR and diversion routing mechanism. Thus, next, we will prove the complexity of this problem.

Theorem 1. The problem of optimal bandwidth balancing is nondeterministic polynomial (NP)-hard.

Proof. Once $G=(V,E)$, ω and operation $Map(f_i)$ are given, our load balancing problem is easily verified in polynomial time. Therefore, finding the decision of $Map(f_i)$ is in the NP-class. We prove that our problem is NP-hard by reducing from the k-dense subgraph problem, which is proven to be NP-complete[40]. The minimum point-to-point connection problem is that, for a given graph $G=(V,E)$, positive integer k, and weight w_i of each $e_i \in E$, find induced subsets V', E' and $|V'|=k$, such that the total weight $W=\sum_{e_i \in E'} w_i$ of E' is minimized.

Suppose a graph $G=(V,E)$, where $V=\{a,b,c,d\}$, $E=\{(a,c),(a,d),(b,c),(b,d),(c,d)\}$ and $\omega=3$. Nodes a,b, and c of G' are diversion routing capable. The source and destination of flow f are a and d, respectively. Diversion routing path Div_f is undetermined, and the shortest routing path is $S_f=(a,d)$. Assume that the link utilization U_{ad} of link (a,d) is the maximum U in graph G. Thus, to achieve the load balancing target, the fewer the packets transmitted through S_f, the more minimized the maximum U of G is. Because nodes a,b, and c connect to node d, if we find the diversion routing path from source a to node b or c, the total diversion routing path Div_f of f is determined. Consequently, according to $k=\omega=3$, we need to choose subsets $V' \subseteq V$ and $E' \subseteq E$, such that the total occupied bandwidth $B=\sum_{e_i \in E'} B_i$ of E' is minimized. This choice also denotes that the shortest routing path $S_f=(a,d) \notin E'$, because U_{ad} is the maximum U in G. Thus, the diversion routing problem for load balancing is equivalent to the minimum point-to-point connection problem. Obviously, this problem is an NP-hard, and we cannot achieve the optimal load balancing in polynomial time. ■

5 Algorithm Design for Improving Control Overheads

In this section, we introduce two novel algorithms for improved control overheads. First, we introduce the idea and analyze the theoretical performance of the LBA-DR. Then, we describe our IACO to achieve our target of bandwidth load balancing and overload control.

5.1 LBA-DR algorithm and analysis

Here, we introduce our algorithm LBA-DR (load balancing algorithm with diversion routing), which is a novel way to achieve the bandwidth load balancing objective and limited control overhead in SR. To achieve the load balancing and forwarding overload optimization objective in Formula (8), we first describe the mechanism and details of our LBA-DR, and then we analyze its approximate performance and compute the approximate ratio of the LBA-DR.

5.1.1 Algorithm description

To achieve the objective of Formula (8), we design the LBA-DR for load balancing, and we show the LBA-DR in Algorithm 1. The LBA-DR includes three steps to achieve bandwidth load balancing and overhead control optimization.

- **Step 1.** After receiving the data transmission demand from source node s, the SR controller computes a suitable routing path set P for s with the diversion routing number constraint δ.
- **Step 2.** From our proof above, Formula (8) with constraint in Formula (9) is NP-hard. To solve the problem in polynomial time, we relax the constraint of $z_x^i \in \{0,1\}$ as $z_x^i \in [0,1]$. This means that flow f can select more than one routing path to construct data transmission. Thus, Formula (8) with constraint in Formula (9) is transformed into a linear programming

Algorithm 1 Load balancing algorithm with diversion routing

```
begin
    SR controller receives d(f_i) from source node s.
    Controller searches for a suitable path set
    P_i = {p_1, p_2, ..., p_m} from source to destination.
    Establish the linear programming from the objective as a
    relaxed scheme.
    Obtain the optimal bandwidth allocation solution
    Opt = {(p_1, b_1), ..., (p_m, b_m)}.
    for i = 1 to m do
        Use Pr(p_i) = b_i / d(f_i) to compute the probability for
        each p_i.
    Randomly select a path p according to the probability
    distribution of Pr(p_i).
    SR controller generates p for f and transmits it to the
    source node to construct data connection.
```

problem as follows.

$$\text{Min } U \tag{10}$$

$$\text{s.t.} \begin{cases} \sum\limits_{x=1}^{m} z_x^i = 1, \forall f_i \in F \\ \sum\limits_{f \in F} \sum\limits_{p \in P, e \in E} z \cdot d(f) \leqslant U \cdot C, \forall e \in E \\ z_x^i \in [0,1], \forall f_i \in F \end{cases} \tag{11}$$

Because the problem involves linear programming, we can solve it in polynomial time and obtain an optimal solution $Opt = \{(p_1, b_1), \ldots, (p_m, b_m)\}$, where (p_i, b_i) denotes the bandwidth allocation b_i on feasible path $p_i \in P$ from a linear programming solver.

- **Step 3.** After obtaining Opt from relaxed Formula (10), we use randomized rouding method[41] to select one path from Opt. We use $Pr(p_i) = \frac{b_i}{d(f_i)}$ to compute each path probability from Opt. Then we randomly choose one path according to the path probability distribution. Then, the SR controller will transmit this path for the source to construct data flow.

5.1.2 Approximate ratio analysis

After introducing the LBA-DR, we analyze the approximate ratio of our algorithm. Let E represent the mathematical expectation. First we introduce two important lemmas.

Lemma 1. (Chernoff Bound). Given n independent variables $x_1, x_2, \ldots, x_n$, $\forall x_i \in [0,1]$. Let $\mu = E\left(\sum\limits_{i=1}^{n} x_i\right)$. Then, $Pr\left[\sum\limits_{i=1}^{n} x_i \geqslant (1+\epsilon)\mu\right] \leqslant \mathrm{e}^{\frac{-\epsilon^2\mu}{2+\epsilon}}$, where ϵ is an arbitrarily positive value.

Lemma 2. (Union Bound). Given a countable set of n events $A_1, A_2, \ldots, A_n$, each event A_i happens with probability $Pr(A_i)$. Then $Pr(A_1 \cup A_2 \cup \cdots \cup A_n) \leqslant \sum\limits_{i=1}^{n} Pr(A_i)$.

According to our relaxed Formula (10), the linear programming solver generates an optimal solution and the algorithm selects one path p_{opt} from it by randomized rounding. We use t_f^e to denote the occupied bandwidth from f on edge e. According to the probability distribution of $Pr(p)$, the occupied bandwidth expectation of edge e from f can be formulated as follows.

$$\forall f_i \in F, \forall e \in E, E(t_{f_i}^e) = d(f_i) \cdot Pr(p) \tag{12}$$

Let $U_{\max}$ denote the maximum bandwidth use in the network with the optimal solution. We assume that the capacity of each edge is C. Because $t_{f_1}^e, t_{f_2}^e, \ldots, t_{f_n}^e$ are independent random variables, the occupied bandwidth expectation of e can be computed as follows.

$$E\Big(\sum_{f \in F} t_f^e\Big) = \sum_{f \in F} E(t_f^e) = \sum_{f \in F} \sum_{e \in p, p \in P} d(f) Pr(p) \leqslant U_{\max} C \tag{13}$$

According to Formula (13), we obtain the inequality group as follows.

$$\begin{cases} \frac{t_f^e}{U_{\max} C} \in [0,1] \\ E\left(\sum\limits_{f \in F} \frac{t_f^e}{U_{\max} C}\right) \in [0,1] \end{cases} \tag{14}$$

According to Lemma 1 and Formula (14), we have the formula as follows.

$$Pr\left[\sum_{f \in F} \frac{t_f^e}{U_{\max} C} \geqslant (1+\epsilon)\mu\right] \leqslant \mathrm{e}^{\frac{-\epsilon^2\mu}{2+\epsilon}} \tag{15}$$

From Formula (14) we know that $\mu = \sum\limits_{f \in F} \frac{t_f^e}{U_{\max} C} \in [0,1]$. We use n_E to denote the total number of edges in our network system. Thus the Formula (15) can be transformed as follows.

$$Pr\left[\sum_{f \in F} \frac{t_f^e}{U_{\max} C} \geqslant (1+\epsilon)\right] = Pr\left[\sum_{f \in F} \frac{t_f^e}{U_{\max} C} \geqslant (1+\epsilon)\mu\right] \leqslant \mathrm{e}^{\frac{-\epsilon^2\mu}{2+\epsilon}} \tag{16}$$

From Lemma 2 we know that

$$Pr\left[\sum_{e \in E} \sum_{f \in F} \frac{t_f^e}{U_{\max} C} \geqslant (1+\epsilon)\right] \leqslant \sum_{e \in E} Pr\left[\sum_{f \in F} \frac{t_f^e}{U_{\max} C} \geqslant (1+\epsilon)\right] \tag{17}$$

Therefore

$$Pr\left[\sum_{e \in E} \sum_{f \in F} \frac{t_f^e}{U_{\max} C} \geqslant (1+\epsilon)\right] \leqslant \sum_{e \in E} \mathrm{e}^{\frac{-\epsilon^2\mu}{2+\epsilon}} = n_E \mathrm{e}^{\frac{-\epsilon^2\mu}{2+\epsilon}} \leqslant n_E \mathrm{e}^{\frac{-\epsilon^2}{2+\epsilon}} \tag{18}$$

Letting $\epsilon = 2\ln n_E + 2$, from inequality 18 we obtain

$$Pr\left[\sum_{e \in E} \sum_{f \in F} \frac{t_f^e}{U_{\max} C_{\max}} \geqslant (2\ln n_E + 3)\right] \leqslant n_E \mathrm{e}^{\frac{-\epsilon^2}{2+\epsilon}} = \frac{1}{n_E} \cdot \mathrm{e}^{-\frac{2}{\ln n_E + 1}} \leqslant \frac{1}{n_E} \tag{19}$$

Thus, we obtain the approximate ratio of our LBA-DR as $(2\ln n_E + 3)$, which means that the maximum bandwidth use of the LBA-DR approximates the optimal solution of Formula (8) with $(2\ln n_E + 3)$.

5.2 IACO algorithm for load balancing

Here we will present our bandwidth balancing and forwarding overload optimization algorithm based on IACO. First, we introduce the classical ant colony algorithm, and then we discuss our novel optimization strategy that balances bandwidth use and optimizing a forward overload situation based on IACO.

5.2.1 Traditional ant colony algorithm

The traditional ant colony algorithm is a classical heuristic method that is suitable for the traveling salesman problem (TSP). This algorithm imitates the working mechanism and physiological features of ant colony for a local optimal solution in each iteration to approximate a global optimal solution. Initially, the ant colony dispatches a certain number of ants to explore the possible routing paths between the source and destination. In the process of searching paths to the destination, the artificial ants generate a substance, called pheromone, to guide other ants to the destination. The higher the pheromone concentration is, the greater the attraction to other ants. This degree of attraction can be expressed as follows.

$$P_{ij}^{k}(t) = \begin{cases} \dfrac{\tau_{ij}^{\alpha}(t)\cdot\eta_{ij}^{\beta}(t)}{\sum_{s\in L}\tau_{ij}^{\alpha}(t)\cdot\eta_{ij}^{\beta}(t)}, & j \in R_k \\ 0, & \text{otherwise} \end{cases} \tag{20}$$

where $P_{ij}^{k}(t)$ denotes the probability of ant k selecting a path from node i to neighbor j at time t, R_k denotes the residual node set for k to visit, $\tau_{ij}(t)$ denotes the pheromone concentration between i and j at time t, $\eta_{ij}(t)$ denotes the heuristic function between i and j at time t, and α and β denote the dependence degree of the pheromone and the heuristic factor for a new path, respectively. In the TSP, η can be formulated as

$$\eta_{ij}(t) = \frac{1}{d_{ij}} \tag{21}$$

where d_i denotes the physical distance between i and j. This equation reflects that the shorter the path between i and j, the higher the attractiveness to the ants.

After discovering a certain path from source to destination, the ants release pheromone on this path. Similar to a natural environment, the artificial pheromone evaporates with time to avoid trapping into local optimization. The evaporation function can be formulated as follows.

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \Delta\tau_{ij}(t) \tag{22}$$

where ρ is the pheromone evaporation factor and $\Delta\tau_{ij}(t)$ denotes the total pheromone quantity between i and j at time t. Obviously, $\Delta\tau_{ij}(t)$ can be computed as follows.

$$\Delta\tau_{ij}(t) = \sum_{k=1}^{n}\tau_{ij}^{k}(t) \tag{23}$$

where n denotes the total number of artificial ants.

There are several methods for computing $\tau_{ij}^{k}(t)$. Here, we introduce the classical ant-cycle model, which updates the global pheromone after an iteration of all ants. The computation method of $\tau_{ij}^{k}(t)$ can be expressed as follows.

$$\tau_{ij}^{k}(t) = \frac{Q}{L^{k}} \tag{24}$$

where $Q > 0$ denotes the pheromone increase factor and L^k denotes the length of ant k's path in this iteration.

Because of the attraction of ants to pheromone and the evaporation of pheromone, the ants generate a local optimization in each iteration. After a certain number of iterations, the local optimization will gradually approximates the global optimization.

5.2.2 IACO algorithm

The section above shows that the artificial ants are driven through the objective of searching for the shortest path from source to destination in the traditional ant colony algorithm. Meanwhile, ants explore the optimal path under no limitation, which probably results in an overload of packet forwarding. Considering our objective of balancing bandwidth use and limiting the diversion routing number for each flow f, we propose IACO which not only prompts ants to search for a path of low bandwidth use, but also controls the forward labels to avoid an overload of packet forwarding.

5.2.3 Optimizing packet forwarding

Before introducing our algorithm of bandwidth load balancing, we first present our method for optimizing the packet length.

Generally, the data flow f from source s to destination d is transmitted through traditional routing path S_f according to the Dijkstra algorithm. Assume that $S_f = \{s, x_1, x_2, \ldots, x_m, d\}$ and $Div_f = \{s, y_1, y_2, \ldots, y_n, d\}$, where $x_1, x_2, \ldots, x_m$ denotes the intermediate nodes between s and d in S_f, and $y_1, y_2, \ldots, y_n$ denotes the intermediate nodes in Div_f. From Eqs. (6) and (7), we know that while hosts transmit data through Div_f, the source node needs to append diversion routing labels into the SR header to direct the

next hop of packets. We use Drn_f to count the number of diversion routing labels of flow f. Therefore, if Drn_f of f is overlarge, it will decrease the forwarding rate and cause an overload of packet forwarding during data transmission.

To achieve the constraint of Drn_f, first, we use α to limit the maximum Drn_f in Eq. (9). Now we define the situation of packet forwarding overload. Intuitively, Drn_f is considered overlarge when it exceeds the node number of S_f too much or a certain percentage of the total number of nodes in the network $G = (V, E)$.

Definition 1. If $Drn_f \geqslant \zeta \times n_1$ or $Drn_f \geqslant \xi \times n_2$, then Div_f will cause an overload of packet forwarding, where n_1 denotes the number of total nodes in network $G = (V, E)$, n_2 denotes the number of S_f, $\zeta \in (0, 1)$, and $\xi > 1$ is the parameter.

According to this definition, we design Algorithm 2 to avoid the overload of packet forwarding when Algorithm 3 generates the current best path T.

In Algorithm 2, from lines 2 to 3, we first obtain the trail T that is optimized through Algorithm 3 and S_f from the SR controller. In line 4, we initialize Drn and the counting label $judge$. From line 5 to 10, we check each intermediate node from T and compute Drn. In line 10, we judge the quality of T according to Definition 1 above. Therefore, through Algorithm 2, we achieve our constraint of the number of diversion routing labels and avoid packet overload in data transmission.

5.2.4 Balanced bandwidth use

According to our target of balancing bandwidth use, we aim to minimize the maximum bandwidth of the network in Formula (8). To satisfy our load balancing demand,

Algorithm 2 Optimizing packet forwarding

```
1  begin
2      Obtain a trail T;
3      Obtain S_f from SR controller.
4      Initialize Drn = 0 and judge = false.
5      for i = 1 to T.size do
6          for j = 1 to S_f.size do
7              if T[i] ≠ S_f[j] or T[i] = S_f[j] but
               T[i + 1] ≠ S_f[j + 1] then
8                  judge = true.
9          if judge = true then
10             Drn = Drn + 1.
11     if Drn ⩾ ζ × n_1 or Drn ⩾ ξ × n_2 then
12         Confirm best path.
```

Algorithm 3 Bandwidth load balancing

```
1  begin
2      SR controller collects bandwidth situation C and O^t and
       generate U^t = {u_12(t), ..., u_ij(t), ..., u_n(n-1)(t)}.
3      Generate transmission requirement from source to
       destination.
4      Initialize ants and pheromones.
5      For each ant
6      for k = 0 to n do
7          Compute P_{x_{k-1}x_k}(t).
8          if ∃P_{x_{k-1}x_k}(t) > 0 then
9              Generate next node x_k.
10             if x_k = j then
11                 Collect x_k into queue Q_k.
12                 Break.
13             else
14                 Select x_k as the next node.
15                 Collect x_k into queue Q_k.
16         else
17             Build trail fail.
18     Generate the current best trail T through Algorithm 2.
19     Update pheromones.
20 Generate path Div and save.
21 Node i transmits data to j through Div.
```

the SR controller will first collect the whole network bandwidth condition, and generate two basic matrices, as follow.

- **Bandwidth capacity matrix C:** The size of C is $N \times N$, where N denotes the number of nodes in network $G = (V, E)$. $\forall c_{ij} \in C$ represents the bandwidth capacity of node i to j. If i and j are disconnected, we set $c_{ij} = 0$.
- **Occupied bandwidth matrix O^t:** The size of O^t is also $N \times N$, where N denotes the number of nodes in this network at time t. $\forall o_{ij}(t) \in O^t$ represents the occupied bandwidth of node i to j. If i and j are disconnected, we set $o_{ij}(t) = 0$.

After receiving bandwidth conditions from each node, the SR controller generates the third matrix U^t through C and O^t. U^t represents the bandwidth use matrix, which is $N \times N$ and N is the number of nodes. $\forall u_{ij}(t) \in U^t$ denotes the bandwidth use of node i to j. Apparently, $u_{ij}(t)$ can be computed as follows:

$$u_{ij}(t) = \begin{cases} \frac{o_{ij}(t)}{c_{ij}}, & c_{ij} \neq 0 \\ \infty, & \text{otherwise} \end{cases} \tag{25}$$

Next, the SR controller transmits U^t to each node to replace the adjacent matrix of the traditional ant

colony algorithm. Therefore, we propose an improved probability equation that actuates ants to explore the link of lower bandwidth use as follow:

$$P_{ij}^{k}(t)=\begin{cases}\dfrac{\tau_{ij}^{\alpha}(t)\cdot\eta_{ij}^{\beta}(t)}{\sum_{s\in L}\tau_{ij}^{\alpha}(t)\cdot\eta_{ij}^{\beta}(t)}, & j\in R_k\\ 0, & \text{otherwise}\end{cases} \tag{26}$$

where η is expressed as follows:

$$\eta_{ij}(t)=\frac{1}{u_{ij}(t)} \tag{27}$$

The guidance of η for ants to explore possible paths actuates ants to be inclined to search the path of lower bandwidth use. Therefore, the paths that ants explore satisfy our objective of bandwidth load balancing.

Assume that one ant has generated a path $p=\{x_1,x_2,\ldots,x_n\}$ from source x_1 to destination x_n. Different from the traditional algorithm that computes the length p through $p=\sum_{k=1}^{n-1}d_k$, where d_k denotes the weight of the edge between x_k and x_{k+1}, we propose a novel method for computing the path quality to satisfy our objective of bandwidth load balancing. We use $u_{ij}(t)$ to represent the edge weight between i and j. Obviously, accumulating each $u(t)$ in p is unsuitable for evaluating the quality of p. In our problem, the quality of p is determined by $u_{p_{\max}}(t)$, which represents the maximum $u(t)\in p$. This is because $u_{p_{\max}}(t)$ is the benchmark in data flow transmission. The higher $u_{p_{\max}}(t)$ is, the lower the quality of p. Additionally, this p is exactly what we need to avoid selecting according to our target 8. Therefore, we formulate the quality of path p as follows.

$$q_p=\max\{u_{x_1}(t),u_{x_2}(t),\ldots,u_{x_n}(t)\} \tag{28}$$

In general, we design Algorithm 3 to achieve bandwidth load balancing. Algorithm 3 is the main process for ants to discover each possible path from the source node to the destination node in this network. First, the SR controller obtains network situation matrix C and O^t, and generates bandwidth utilization matrix U^t through Eq. (25). In line 3, node i as the source generates the transmission requirement and selects j that satisfies $i\neq j$ as the destination to discover the diversion routing path Div. After determining source i and destination j, in line 4, matrix $ants$ and $pheromones$ are initialized. From lines 5 to 17, each artificial ant explores the optimal path from i to j. In line 7, ant computes the probability from the current node x_{k-1} to other possible nodes x_k. If the probability $P_{x_{k-1}x_k}(t)$ is executable for the ant to select the next node x_k, in line 10 ant will recognize if x_k is the destination. If x_k is the destination, this ant will finish its mission and record the trail into queue Q_k. Otherwise, from line 14 to line 15, the ant selects x_k as the next node and moves to x_k, then x_k is written into Q_k . If the probability $P_{x_{k-1}x_k}(t)$ is not executable, the mission of this ant is judged as a failure, and the next iteration of another ant begins in line 17. After finishing one iteration of all $ants$ and saving these paths into Q_k, from line 18 to line 19, Algorithm 3 computes the quality of all paths according to Eq. (28), and generates the current best trail T from Q_k according to Algorithm 2, and updates matrix $pheromones$ through Eq. (22). Then in line 20, Algorithm 3 will generate the optimal path Div, namely the diversion routing path from source i to j after finishing all iterations and saving it into the SR controller. Consequently, in line 21, i transmit data to j through Div to achieve our bandwidth load balancing objective.

6 Performance Evaluation

6.1 Simulation setups

We evaluate the performance of our bandwidth load balancing and packets length optimizing algorithm, including the LBA-DR and IACO. We use the Internet Topology Zoo to perform the simulation[42]. The topologies we choose include Abilene, Abvt, Aconet, Agis, Ai3, Airtel, BellCanada, BellSouth, Bics, and Ion. We assume that all links are all undirected, and all nodes are diversion routing capable in these topologies. The details of each topology are shown in Table 2.

To assess the performance of load balancing with limited control overhead, we select two algorithms for

Table 2 Network details list.

Topology	Number of nodes	Number of edges
Abilene	11	14
Abvt	23	31
Aconet	23	55
Agis	25	70
Ans	25	47
Ai3	25	75
Airtel	25	93
Bics	32	48
BellCanada	48	64
BellSouth	51	66
Ion	125	146

comparison with our IRTE algorithms as follows.

• **Traditional algorithm.** To achieve the objective of load balancing, traditional algorithm such as OSPF guides the data flow to transmit through the shortest path by the Dijkstra algorithm.

• **OFLoad[43].** This algorithm performs data transfer by distinguishing small and large flows. The large flows transmit data through the shortest path, while the small flows transmit data through several permissible paths through the weight of paths.

• **D-LBAH[44].** The D-LBAH algorithm achieves load balancing based on the SDN architecture. In D-LBAH, the controller sets a threshold value and computes a path set for each flow in a network. The controller uses a threshold value to limit the number of flows passing through each link. If the number of flows traversed by one link does not exceed the threshold, the controller directs the source node to transmit data according to the default path. Otherwise, the controller selects the idlest path from the path set for the source node to transmit data.

• **Optimal solution.** We use the method for traversing all traffic assignment paths to obtain the optimal solution. Because of its computation and running time, we only compare it with our algorithms regarding CPU use to achieve the test-bed experiment.

For each topology, we first use the Dijkstra algorithm to generate the shortest routing path from any source nodes to any destination nodes for comparison. Then we randomly generate the source node and destination node of a flow, and establish a constant data transmission through this flow. For simplicity, we assume that each link capacity of the topologies is 100 MB, and the bandwidth of each flow is randomly from 3 MB to 8 MB. For the OFLoad algorithm, we define flow of more than 6 MB as a large flow, and the other as a small flow. For the D-LBAH algorithm, we define the threshold of each link as 0.35, which means that the number of flows passing through each link does not exceed 35% of the maximum number of flows in the network. Then we operate our bandwidth load balancing algorithms of the IRTE. Meanwhile, we consider the number of diversion routing labels to each flow. In this experiment, we assume $\zeta = 0.5$ and $\xi = 4$ to control the packet length in a reasonable area for IACO. When we generate the diversion routing path of a flow, we transfer data on it and the shortest routing path. After the data transmission task finishes, we compare the performance of these five algorithms.

We first exhibit the maximum bandwidth use of these five algorithms in six topologies, and then we compare the performance of load balancing through these algorithms as the number of flow increases. Meanwhile, we record the CPU use for the LBA-DR and IACO to compare the control overhead of the optimal solution. For IACO, we compare the change in maximum bandwidth use in different algorithms. Then we compare the maximum bandwidth use difference of each topology. Meanwhile, we evaluate the maximum bandwidth use situation with the changing number of nodes and edges. Finally, we modify the parameters for our bandwidth load balancing algorithm in each topology, including α and β in Eq. (26).

6.2 Experiment results

6.2.1 Bandwidth use comparison

We select six topologies from the Internet Topology Zoo, including Abvt, Aconet, Ans, BellCanada, BellSouth, and Ion. The number of nodes of these six topologies increases from 23 to 125 to test IRTE performance from a small network to a large network. Figure 3 shows the load balancing performance of maximum bandwidth use for five algorithms in these topologies. We randomly generate 25 flows with a size of 3 MB to 8 MB in these six network topologies. We record the maximum bandwidth use of these networks after all flows are assigned to the suitable routing paths. We set $\alpha = 3$ and $\beta = 3$ for IACO to compare its performance in different topologies. Meanwhile, we control Drn to 7 in each network. Figure 3 shows that the maximum performance advantage of IACO reaches

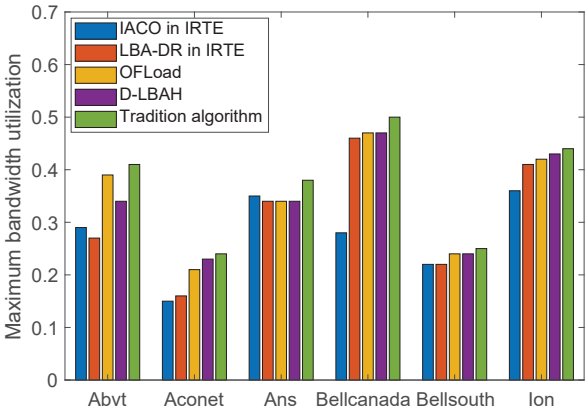


Fig. 3 Maximum bandwidth use in different topologies.

24.6% in BellCanada compared with the other three algorithms. There are also 2.45% to 11.6% performance gains in other topologies compared with the traditional algorithm, OFLoad and D-LBAH. As for LBA-DR, its performance is slightly inferior to IACO in some topologies, but it leads OFLoad by 3.2% to 12%, D-LBAH by 0 to 5.3%, and the traditional algorithm by 2.6% to 15.3%. However, in Ans, we can see that the load balancing performance of LBA-DR, OFLoad, and D-LBAH is close, and has an approximately 1% performance advantage over IACO. This advantage is due to the Drn not being the optimal value for IACO convergence. Therefore, the result obtained by IACO exceeds the value of Drn and the controller automatically selects the shortest path, which leads to slightly worse network load balancing.

6.2.2 CPU use for IACO and LBA-DR

To verify that our experiments meet the test-bed standards, we record the CPU use per second while running our algorithms. Our computer configuration is an Intel Core i5-9750H with 16 GB RAM. We compare our algorithms, including IACO and LBA-DR with optimal solution on BellSouth, which represents a medium-sized network. In Fig. 4, the change in LBA-DR is relatively stable and fluctuates between 15.3% and 16.2%. The change in IACO is slightly larger than LBA-DR, fluctuating between 14.5% and 16.8%. Moreover, the maximum and minimum CPU use of the optimal solution is 23.4% to 24.5%, respectively, which exceeds IACO and LBA-DR by approximately 7%. This comparison means that our algorithms are suitable for operating on the SR controller, and the evaluation complies with the working standard of the test-bed experiment.

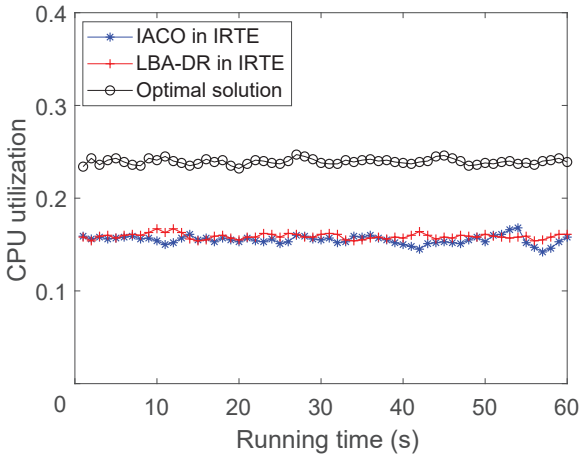


Fig. 4 CPU utilization in Bellsouth.

6.2.3 Bandwidth use with increasing flow number

To inspect the load balancing performance with increasing flow numbers, we select three topologies, Aconet, BellSouth, and Ion, which represent a small network, medium network, and big network, respectively. We control the value of Drn to 6, 8, and 10, respectively, to establish the load balancing performance in these three scale networks. Figures 5–7 show the change in the maximum bandwidth use for each algorithm in Aconet, BellSouth, and Ion. We randomly increment the number of flows from 10 to 30 in Aconet, Bellsouth, and Ion. As we can see, with increasing flow number, the link utilization growth of IACO and LBA-DR is flatter than that of the traditional algorithm, OFLoad and D-LBAH. This is because with increasing traffic demand, which causes network congestion, IACO and LBA-DR in the IRTE can help the hosts to transmit data more effectively. The difference between these two algorithms does not exceed 1.85% in Aconet and BellSouth. For

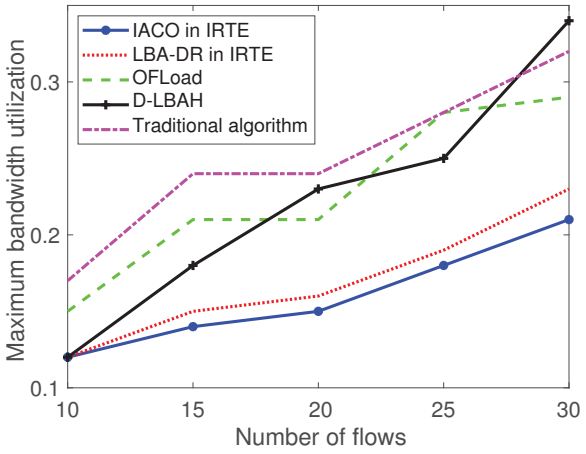


Fig. 5 Increasing flow number in Aconet.

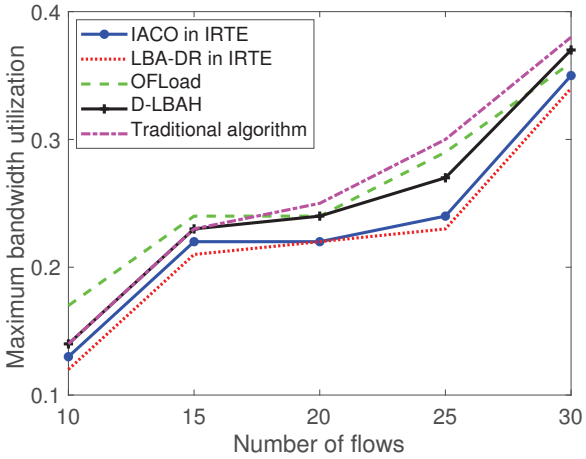


Fig. 6 Increasing flow number in Bellsouth.

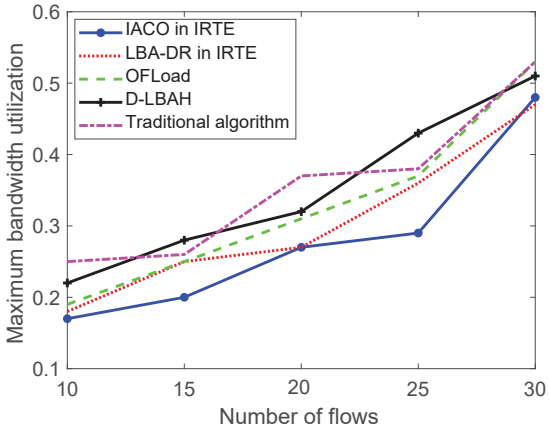


Fig. 7 Increasing flow number in Ion.

example, Fig. 7 shows that the total load balancing situation of IACO is better than that of the LBA-DR. Moreover, we can see that the performance gap between the two algorithms of the IRTE, the traditional algorithm, OFLoad and D-LBAH remains above 1.3% to 10.3%. This result shows the load balancing advantages of IACO and the LBA-DR in the situation of increasing flow number.

6.2.4 Nodes and degrees for IACO

Figures 8 and 9 represent the maximum bandwidth use differences between the traditional algorithm and our bandwidth load balancing algorithm. In Figs. 8 and 9, the number of nodes and degrees of these topologies on the x-axis gradually increase. Figure 8 shows that with increasing nodes, the bandwidth difference of these two algorithms does not show strong regularity. However, Fig. 9 shows that the load balancing performance

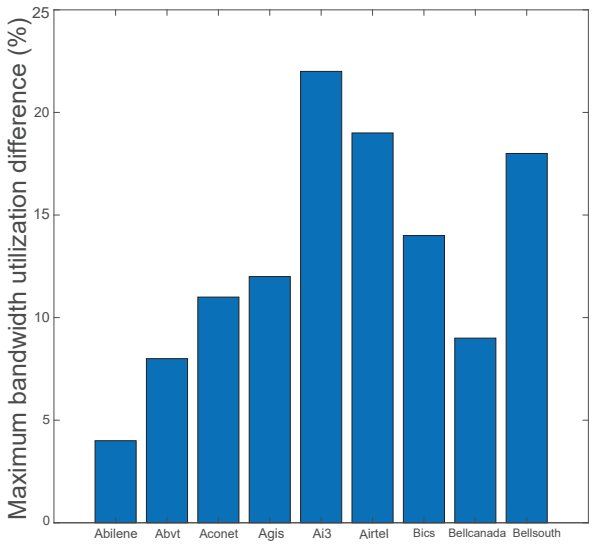


Fig. 8 Bandwidth difference for changing nodes.

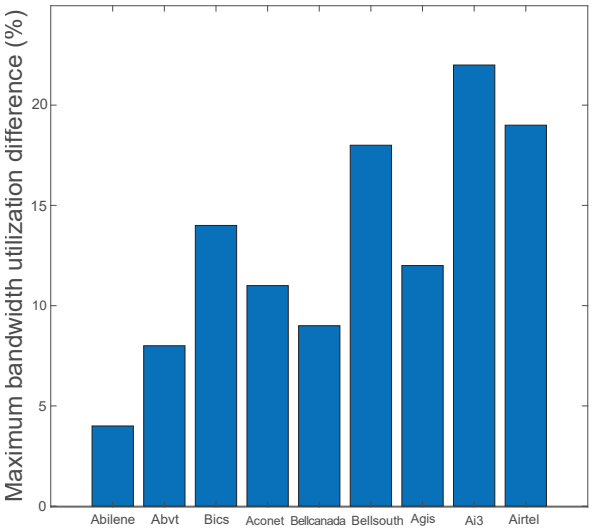


Fig. 9 Bandwidth difference for changing degrees.

generally improves as the degree increases. Combined with the observation of Fig. 8, we see the better the optimization effect of the topology graph with fewer nodes and more degrees. For example, the number of nodes is similar in Abvt, Aconet, Agis, Ai3, and Airtel. However, the performance of load balancing is better in Ai3 and Airtel than in other topologies. Meanwhile, the degrees of Ai3 and Airtel exceed those of the other three topologies. This is because if the number of nodes in the topology is smaller, the degree is greater, so each node has more edges connected. Therefore, the artificial ants of IACO can explore more possible routing paths to achieve the objective of bandwidth load balancing, and it improves the performance of our algorithm.

6.2.5 Dependence degree of pheromones α

Figure 10 shows the changes in α in our IACO

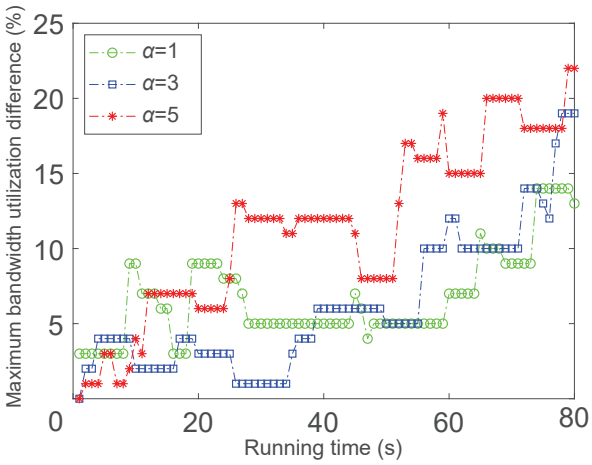


Fig. 10 Bandwidth difference for changing α.

algorithm. For simplicity, we select topology Ai3 to achieve our experimental target. Additionally, we set $\beta = 3$, $\rho = 0.2$, and $Q = 2.5$ to test the performance of our algorithm. Figure 10 shows that at the beginning of establishing data communication, the maximum bandwidth difference between these two algorithms is small. For example, when the number of iterations is smaller than 20, the difference between the two algorithms is less than 20%. However, as the number of iterations increases, the experiment with $\alpha = 5$ gradually showes performance advantages. Above 20 iterations, the bandwidth difference of $\alpha = 5$ approximately exceeds the value of $\alpha = 1$ and $\alpha = 3$. This is because α increases the possibility of ants searching a path along the direction of high pheromone concentration. At the initial stage of the iterations the advantages of the high-pheromone path are not sufficiently obvious because the overall bandwidth use of the network is relatively low. After a certain number of iterations, the overall bandwidth use of links in the network increases. Therefore, high pheromone concentrations with a low bandwidth use path reflects the advantages of load balance performance. It will direct the ants to the lower bandwidth use routing path, which shows the advantage of $\alpha = 5$ compared with $\alpha = 1$ and $\alpha = 3$.

6.2.6 Heuristic factor β

Figure 11 shows the changes in β in our IACO algorithm. To show the connection between α and β, we still select topology Ai3 to finish our experiment. We set $\alpha = 3$, $\rho = 0.2$, and $Q = 2.5$ to test the performance of our algorithm. Figure 11 shows that with an increasing number of iterations, the polyline of $\beta = 3$ shows the performance advantage of bandwidth load balancing. The performance gap between the other two polylines is relatively close. After 40 iterations, the load balancing optimization with $\beta = 3$ is significantly better than the case with $\beta = 1$ or $\beta = 5$. This result is obtained because β influences the probability of artificial ants searching a new routing path, except for following the path of high pheromone concentration. Therefore, the polyline of $\beta = 3$ will stimulate the possibility of ants searching for new paths while retaining the possibility of tracing a path with high pheromone concentrations. As for $\beta = 1$, an undersized β will make ants cling to paths with high pheromone concentration and reduce the possibility of discovering paths with lower bandwidth use. In contrast, an oversized β, such as $\beta = 5$, will over-encourage ants to explore new paths, while ignoring the importance of paths with high pheromone concentration to a certain extent.

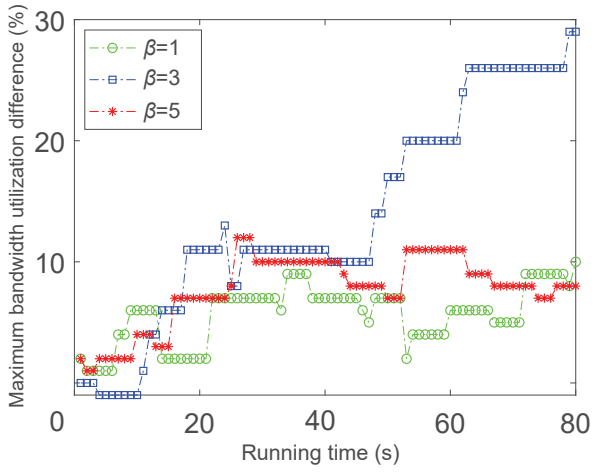


Fig. 11 Bandwidth difference for changing β.

7 Conclusion

In this paper, we propose the IRTE to achieve load balancing with limited control overheads in the SR environment. We introduce the function of the SR controller and hosts in our IRTE system design. We propose the idea of diversion routing, and formulate the mapping problem in data flow transmission. We prove that the diversion routing problem is NP-hard. Additionally, we design the LBA-DR and IACO algorithms to realize the objective of bandwidth load balancing and limited control overheads. We evaluate the IRTE in different real-world topologies and compare the difference in performance between the IRTE and the traditional routing algorithm. The results show that the load balancing performance of the IRTE is better than that of the traditional algorithm and effectively achieves limited control overheads in SR.

Acknowledgment

This work was supported in part by the National Natural Science Foundation of China (Nos. 61772345 and 61902258), the Major Fundamental Research Project in the Science and Technology Plan of Shenzhen (Nos. JCYJ20190808142207420, GJHZ20190822095416463, and RCYX20200714114645048), the Natural Science Foundation of Guangdong Basic and Applied Basic Research (No. 2021A1515011857), and the Pearl River Young Scholars Funding of Shenzhen University.

References

[1] D. O. Awduche and B. Jabbari, Internet traffic engineering using multi-protocol label switching (MPLS), *Comput. Netw.*, vol. 40, no. 1, pp. 111–129, 2002.

[2] N. Wang, K. H. Ho, G. Pavlou, and M. Howarth, An overview of routing optimization for internet traffic engineering, *IEEE Commun. Surv. Tut.*, vol. 10, no. 1, pp. 36–56, 2008.

[3] Y. Wang, Z. Wang, and L. Zhang, Internet traffic engineering without full mesh overlaying, in *Proc. IEEE INFOCOM 2001. Conf. Computer Communications. Twentieth Annual Joint Conf. IEEE Computer and Communications Society* (*Cat. No. 01CH37213*), Anchorage, AK, USA, 2001, pp. 565–571.

[4] A. Mendiola, J. Astorga, E. Jacob, and M. Higuero, A survey on the contributions of software-defined networking to traffic engineering, *IEEE Commun. Surv. Tut.*, vol. 19, no. 2, pp. 918–953, 2017.

[5] M. Karakus and A. Durresi, A survey: Control plane scalability issues and approaches in software-defined networking (SDN), *Comput. Netw.*, vol. 112, pp. 279–293, 2017.

[6] S. H. Yeganeh, A. Tootoonchian, and Y. Ganjali, On scalability of software-defined networking, *IEEE Commun. Mag.*, vol. 51, no. 2, pp. 136–141, 2013.

[7] P. L. Ventre, M. M. Tajiki, S. Salsano, and C. Filsfils, SDN architecture and southbound APIs for IPv6 segment routing enabled wide area networks, *IEEE Trans. Netw. Serv. Manag.*, vol. 15, no. 4, pp. 1378–1392, 2018.

[8] Z. N. Abdullah, I. Ahmad, and I. Hussain, Segment routing in software defined networks: A survey, *IEEE Commun. Surv. Tut.*, vol. 21, no. 1, pp. 464–486, 2019.

[9] P. L. Ventre, S. Salsano, M. Polverini, A. Cianfrani, A. Abdelsalam, C. Filsfils, P. Camarillo, and F. Clad, Segment routing: A comprehensive survey of research activities, standardization efforts, and implementation results, *IEEE Commun. Surv. Tut.*, vol. 23, no. 1, pp. 182–221, 2021.

[10] J. Zhang and C. Zhao, Q-SR: An extensible optimization framework for segment routing, *Comput. Netw.*, vol. 200, p. 108517, 2021.

[11] X. Li and K. L. Yeung, Bandwidth-efficient network monitoring algorithms based on segment routing, *Comput. Netw.*, vol. 147, pp. 236–245, 2018.

[12] X. Jia, Q. Li, Y. Jiang, Z. Guo, and J. Sun, A low overhead flow-holding algorithm in software-defined networks, *Comput. Netw.*, vol. 124, pp. 170–180, 2017.

[13] R. Banerjee and S. Das Bit, Low-overhead video compression combining partial discrete cosine transform and compressed sensing in WMSNs, *Wirel. Netw.*, vol. 25, no. 8, pp. 5113–5135, 2019.

[14] A. Cianfrani, M. Listanti, and M. Polverini, Incremental deployment of segment routing into an ISP network: A traffic engineering perspective, *IEEE/ACM Trans. Netw.*, vol. 25, no. 5, pp. 3146–3160, 2017.

[15] X. Xiao, A. Hannan, B. Bailey, and L. M. Ni, Traffic engineering with MPLS in the internet, *IEEE Netw.*, vol. 14, no. 2, pp. 28–33, 2000.

[16] B. Fortz, J. Rexford, and M. Thorup, Traffic engineering with traditional IP routing protocols, *IEEE Commun. Mag.*, vol. 40, no. 10, pp. 118–124, 2002.

[17] A. Elwalid, C. Jin, S. Low, and I. Widjaja, Mate: MPLS adaptive traffic engineering, in *Proc. IEEE INFOCOM 2001, Conf. Computer Communications, Twentieth Annual Joint Conf. IEEE Computer and Communications Societies*, Anchorage, AK, USA, 2001, pp. 1300–1309.

[18] A. Feldmann and J. Rexford, IP network configuration for intradomain traffic engineering, *IEEE Netw.*, vol. 15, no. 5, pp. 46–57, 2001.

[19] S. Kandula, D. Katabi, B. Davie, and A. Charny, Walking the tightrope: Responsive yet stable traffic engineering, *ACM SIGCOMM Comput. Commun. Rev.*, vol. 35, no. 4, pp. 253–264, 2005.

[20] S. Agarwal, M. Kodialam, and T. V. Lakshman, Traffic engineering in software defined networks, in *2013 Proc. IEEE INFOCOM*, Turin, Italy, 2013, pp. 2211–2219.

[21] I. F. Akyildiz, A. Lee, P. Wang, M. Luo, and W. Chou, Research challenges for traffic engineering in software defined networks, *IEEE Netw.*, vol. 30, no. 3, pp. 52–58, 2016.

[22] A. Guezzaz, Y. Asimi, M. Azrour, and A. Asimi, Mathematical validation of proposed machine learning classifier for heterogeneous traffic and anomaly detection, *Big Data Mining and Analytics*, vol. 4, no. 1, pp. 18–24, 2021.

[23] M. Chiesa, G. Kindler, and M. Schapira, Traffic engineering with equal-cost-MultiPath: An algorithmic perspective, *IEEE/ACM Trans. Netw.*, vol. 25, no. 2, pp. 779–792, 2017.

[24] A. Bahnasse, F. E. Louhab, H. A. Oulahyane, M. Talea, and A. Bakali, Novel SDN architecture for smart MPLS Traffic engineering-DiffServ Aware management, *Future Gener. Comput. Syst.*, vol. 87, pp. 115–126, 2018.

[25] B. Zhou, J. Li, X. Wang, Y. Gu, L. Xu, Y. Hu, and L. Zhu, Online internet traffic monitoring system using spark streaming, *Big Data Mining and Analytics*, vol. 1, no. 1, pp. 47–56, 2018.

[26] G. Trimponias, Y. Xiao, X. Wu, H. Xu, and Y. Geng, Node-constrained traffic engineering: Theory and applications, *IEEE/ACM Trans. Netw.*, vol. 27, no. 4, pp. 1344–1358, 2019.

[27] C. Filsfils, N. K. Nainar, C. Pignataro, J. C. Cardona, and P. Francois, The segment routing architecture, presented at the 2015 IEEE Global Communications Conf. (GLOBECOM), San Diego, CA, USA, 2015, pp. 1–6.

[28] R. Bhatia, F. Hao, M. Kodialam, and T. V. Lakshman, Optimized network traffic engineering using segment routing, presented at the 2015 IEEE Conf. Computer Communications (INFOCOM), Hong Kong, China, 2015, pp. 657–665.

[29] T. Schüller, N. Aschenbruck, M. Chimani, M. Horneffer, and S. Schnitter, Traffic engineering using segment routing and considering requirements of a carrier IP network, *IEEE/ACM Trans. Netw.*, vol. 26, no. 4, pp. 1851–1864, 2018.

[30] M. C. Lee and J. P. Sheu, An efficient routing algorithm based on segment routing in software-defined networking, *Comput. Netw.*, vol. 103, pp. 44–55, 2016.

[31] Y. Desmouceaux, P. Pfister, J. Tollet, M. Townsley, and T. Clausen, 6LB: Scalable and application-aware load balancing with segment routing, *IEEE/ACM Trans. Netw.*, vol. 26, no. 2, pp. 819–834, 2018.

[32] M. Jadin, F. Aubry, P. Schaus, and O. Bonaventure, CG4SR: Near optimal traffic engineering for segment routing with column generation, presented at the IEEE INFOCOM 2019 - IEEE Conf. Computer Communications, Paris, France, 2019, pp. 1333–1341.

[33] T. Schüller, N. Aschenbruck, M. Chimani, and M. Horneffer, Failure resiliency with only a few tunnels-enabling segment routing for traffic engineering, *IEEE/ACM Trans. Netw.*, vol. 29, no. 1, pp. 262–274, 2021.

[34] J. Haxhibeqiri, I. Moerman, and J. Hoebeke, Low overhead scheduling of LoRa transmissions for improved scalability, *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 3097–3109, 2019.

[35] G. Wang, S. Chattopadhyay, I. Gotovchits, T. Mitra, and A. Roychoudhury, oo7: Low-overhead defense against spectre attacks via program analysis, *IEEE Trans. Softw. Eng.*, vol. 47, no. 11, pp. 2504–2519, 2021.

[36] V. Freschi and E. Lattanzi, A study on the impact of packet length on communication in low power wireless sensor networks under interference, *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 3820–3830, 2019.

[37] Y. H. Robinson, R. S. Krishnan, E. G. Julie, R. Kumar, L. H. Son, and P. H. Thong, Neighbor knowledge-based rebroadcast algorithm for minimizing the routing overhead in mobile ad-hoc networks, *Ad Hoc Netw.*, vol. 93, p. 101896, 2019.

[38] V. S. Devi, T. Ravi, and S. B. Priya, Cluster based data aggregation scheme for latency and packet loss reduction in WSN, *Comput. Commun.*, vol. 149, pp. 36–43, 2020.

[39] H. Zhu, B. Smida, and D. J. Love, Optimization of two-way network coded HARQ with overhead, *IEEE Trans. Commun.*, vol. 68, no. 6, pp. 3602–3613, 2020.

[40] U. Feige, D. Peleg, and G. Kortsarz, The dense k-subgraph problem, *Algorithmica*, vol. 29, no. 3, pp. 410–421, 2001.

[41] P. Raghavan and C. D. Tompson, Randomized rounding: A technique for provably good algorithms and algorithmic proofs, *Combinatorica*, vol. 7, no. 4, pp. 365–374, 1987.

[42] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan, The internet topology zoo, *IEEE J. Sel. Areas Commun.*, vol. 29, no. 9, pp. 1765–1775, 2011.

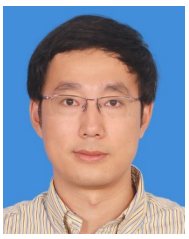
[43] R. Trestian, K. Katrinis, and G. M. Muntean, OFLoad: An OpenFlow-based dynamic load balancing strategy for datacenter networks, *IEEE Trans. Netw. Serv. Manag.*, vol. 14, no. 4, pp. 792–803, 2017.

[44] C. S. Mbacke Babou, D. Fall, S. Kashihara, Y. Taenaka, M. H. Bhuyan, I. Niang, I. Diané, and Y. Kadobayashi, D-LBAH: Dynamic load balancing algorithm for HEC-SDN systems, presented at the 2021 8th Int. Conf. Future Internet of Things and Cloud (FiCloud), Rome, Italy, 2021, pp. 304–310.

Shu Yang received the BS degree from the Beijing University of Posts and Telecommunications, Beijing, China, in 2009, and the PhD degree from Tsinghua University, Beijing, China, in 2014. He is currently an associate researcher with the College of Computer Science and Software Engineering, Shenzhen University, Shenzhen, China. His research interests include network architecture, edge computing and high performance router associations.

Ruiyu Chen received the BS degree from South China Normal University, Guangzhou, China, in 2019. He is pursuing the MS degree at Shenzhen University. His research interests include software-defined network, segment routing, and traffic engineering.

Xiaolei Chang received the BS and MS degrees from Tsinghua University, Beijing, China, in 1999 and 2002. He is pursuing the PhD degree at Tsinghua University. His research interests include edge computing, cloud computing, and data center energy efficiency.



Laizhong Cui is currently a professor in the College of Computer Science and Software Engineering at Shenzhen University, China. He received the BS degree from Jilin University, Changchun, China, in 2007 and the PhD degree in computer science and technology from Tsinghua University, Beijing, China, in 2012. His research interests include future internet architecture and protocols, edge computing, multimedia systems and applications, blockchain, Internet of Things, cloud and big data computing, computational intelligence and machine learning. He led more than ten scientific research projects, including the National Key Research and Development Plan of China, the National Natural Science Foundation of China, the Guangdong Natural Science Foundation of China, and the Shenzhen Basic Research Plan. He has published more than 100 papers in journals such as *IEEE JSAC*, *IEEE TC*, *IEE TKDE*, *IEEE TMM*, *IEEE IoT Journal*, *IEEE TII*, *IEEE TVT*, *IEEE TNSM*, *ACM TOIT*, *IEEE TCBB*, *IEEE Network*, *IEEE INFOCOM*, and *ACM MM*. He serves as an associate editor or a member of editorial board for several international journals, including *IEEE IoT Journal*, *IEEE Transactions on Network and Service Management*, and *International Journal of Machine Learning and Cybernetics*. He is a senior member of the IEEE and the CCF.