Multi-path Routing Deployment Method Based on SRv6

Lihua Song1, Yuzhu Jin1, Pengyu Wang1, Dongchao Ma1,*, Wenlong Chen2, Laizhong Cui3
1. College of Information Science, North China University of Technology, Beijing 100000, China
2. College of Information Engineering, Capital Normal University, Beijing 100000, China
3. College of Computer Science and Software Engineering, Shenzhen University, Shenzhen 518029, China
Corresponding Author: Dongchao Ma
Email: madongchao1980@wo.cn

Abstract—The deployment of Software Defined Wide Area Network (SD-WAN) architecture cannot be accomplished overnight. It is a good attempt to combine Segment Routing (SR) with the existing routing protocol. Multi-path routing can better meet the diverse needs of users. However, deploying multi-path routing will increase the number of Forwarding Information Base (FIB). This paper proposes a method for deploying TD routing in combination with SRv6 (TDSR), which can disperse FIB in some entry routers. Secondly, to address the bandwidth waste caused by the segment lists in the SRv6 header, we propose a compression algorithm (HCMD), based on the comparison of the Shortest Path First (SPF) and SRv6 path. Thirdly, we optimize the algorithm under various scenarios. The experimental results show that TDSR can reduce the number of TD Routing Information Base (RIB) by 69%, and the average compression rate of HCMD can reach 70%. Moreover, HCMD can also be combined with various existing compression methods to comprehensively improve the effect.

Keywords—SD-WAN, Multi-path Routing, Two-Dimensional Routing, SRv6, Segment List Compression

I. INTRODUCTION

Segment Routing (SR) is an important technology in Software Defined Wide Area Network (SD-WAN). The control plane takes the SD-WAN controller as the core for centralized management. The data plane uses Multi-Protocol Label Switching (MPLS) or SRv6. SD-WAN makes hardware universal and simple. However, the network development follows the evolutionary principle, and the deployment of SD-WAN architecture cannot be accomplished overnight. At present, some researches and applications attempt to combine SR with existing distributed router architecture, such as SR+OSPF, SR+BGP, etc. [1][2]

The network layer of the traditional Internet has been using single-path routing for many years, that is, routers complete distributed data forwarding based on the destination IP address hop by hop [32]. This method is more prone to congestion and fault. Therefore, it is not conducive to the rational use of network links. Qiu et al. [31] proposed a novel energy-aware fault tolerance mechanism for WSN, called Informer Homed Routing (IHR). In the IHR, Non Cluster Head (NCH) nodes select a limited number of targets in the data transmission. Therefore it consumes less energy.

With the development of network architecture, multi-path routing has been paid more attention [33][34], whether it is security requirements [35][36], trust link requirements [37], Quality of Service (QoS) requirements, traffic engineering [38], congestion avoidance, etc. [3][4]. Among multi-path routing scheme, i.e.,policy routing,MPLS-related technologies, etc. [5][6], the Two-Dimensional routing (TD routing) that introduces source prefixes has become a research hotspot. The reason is that the development of the network should follow the principle of evolution. TD routing does not change the distributed architecture of the network, and it only makes minor changes to the existing protocols. Therefore, it has received attention in both academia and industry. Especially the standardization of IETF has also made substantial progress [7].

Today’s mainstream TD routing research and IETF drafts are still based on distributed systems. It is concentrated in the AS, with OSPF expansion as the main evolution method [8][9]. The applications are mainly oriented to congestion avoidance, fast rerouting based on backup paths, differentiated services, and hierarchical trusted forwarding, etc. [9][10]. As we all know, with the prosperity of the Internet economy, the number of existing Routing Information Base (RIB) has exploded [11][12] and reached one million in 2021. Many of them may be injected into the routers in the AS through the AS-External-LSAs of iBGP or OSPF. Even the routes within the AS domain have reached tens of thousands. The scale of Forwarding Information Base (FIB) will significantly affect the forwarding speed [13]. Therefore, the addition of TD routing poses a severe challenge to the capabilities of the data plane. Especially in some special applications, the scale of FIB is more sensitive. Although some experts have proposed many methods to improve memory performance, thereby reducing memory power consumption and energy cost[39-41], but facing some scenarios, there are still problems. For example, the cost of some network devices is limited and there are no high-speed FPGAs. Another example is the Low Earth Orbit Satellite Network, which can’t use Three-state Content Addressing Memory (TCAM) because of the power limit. So it usually uses the Double Data Rate (DDR) RAM to store FIB, which makes it sensitive to the scale.

SR [14] is a new technology that can implement multi-path control. It can push intermediate segment lists on the header of the message by stack. For convenience, “segment list” will be referred to “hop” in the following. In particular, SR supports discontinuous intermediate path designation. However, the deployment of SR requires a central controller, which is different from the existing network architecture. When deploying TD routing in combination with SRv6, routes are still spread in a distributed architecture. Data is only pushed on the stack at the entry node and is forwarded according to the segment list. This method of routing in combination with is not only in line with the evolution of the network architecture, but also can reduce the number of FIDs. However, in the SRv6 extension header, the IPv6 address is longer, and the stack will occupy a larger space, which will affect the link utilization. Led by China Mobile, operators and manufacturers such as China Telecom, China Unicom, and ZTE have already proposed the SRv6 header compression method G-SID and promoted standardization in the IETF [15].

Based on the above analysis, this paper presents a method to address the increase of FIB in TD routing deployment.
Then we focus on solving the problem of SRv6 intermediate hops occupying communication bandwidth. The specific contributions are as follows:

1) For the first time, we propose a method for deploying TD routing in combination with SRv6 (TDSR). This method only needs the entry node to overlay the TD routes, and the intermediate routes forward according to the segment list. This method can effectively reduce the number of FIBs on the premise that the forwarding path meets the requirements.

2) To address the bandwidth waste caused by the segment lists in the SRv6 header, we propose a Hop Compression Algorithm based on Multi-Dimensional Routing Difference (HCMD) under the coexistence of one-dimensional routing (OD routing) and TD routing. Under the premise of meeting the consistency of the forwarding path, HCMD obtained the minimum set that must be pushed on the stack with lower computing overhead. To reduce the computing cost further, we focus on the original intention of multi-path routing applications. We sort out several typical applications and discuss several lower-cost “inaccurate path” calculation methods.

It should be pointed out that although this paper is based on the coexistence of OD routing and TD routing, the compression method can be extended to other SR application scenarios. For details, please refer to the motivation in Section II. The rest of this paper is organized as follows: Section II analyzes the related works. Section III introduces the compression algorithm and optimization ideas, then discusses low-overhead algorithms for calculating “inaccurate paths”; Section IV analyzes the optimization effect of TDSR and the performance of HCMD through simulation.

II. RELATED WORKS AND MOTIVATIONS

In recent years, since the TD routing was proposed, it has received extensive attention. TD routing is a strategy based on source and destination prefixes. It is an emerging multi-path strategy [16]. Yang et al. [5] proposed a TD routing based on a new FIB structure FISE, which solves the TCAM space explosion problem which is caused by TD forwarding matching. Chen et al. [9] proposed a differentiated forwarding service and optimization for Internet traffic in the autonomous system based on TD forwarding. The border router obtains the optimal path from the access router to itself. It sends information along the path in the reverse direction to establish a TD forwarding path. TD routing information is released on specific paths, which reduces resource consumption in the control plane and data plane. Chen et al. [10] proposed a congestion avoidance forwarding mechanism based on OD routing. When a link is congested, a congestion avoidance path (CA-Path) will be constructed based on TD forwarding. They also optimized CA-Path to reduce the deployment of FIB.

SR is a source routing technology that can combine multiple scenarios such as VNF, SDN, etc. [17][18]. which simplifies the network. However, there are also problems such as large label stacks, packet header length, and limited segment list depth (SLD). There have been some studies on how to solve those problems.

Lee et al. [18] proposed an algorithm based on SR + SDN. It reduces the packet header size by limiting the number of hops between the source and destination. However, constraints may reduce the quality of the selected path. Anix et al. [19] proposed a method to partition AS into subdomains. The path is segmented stored to solve the problem of limited SLD. However, the method requires many logical SIDs to represent the path. Therefore, space is consumed in the segment list to store logical SID. Teng et al. [20] proposed the SRTE-L model and a traffic scheduling scheme. They limit the length of the SR path and the number of intermediate hops by setting variable L and a path constraint. This method reduces the number of hops and increases link utilization, but it needs excessive computing overhead. Li et al. [21] proposed the K-sMILP algorithm to find a K-segment path to carry all flows, thereby minimizing the maximum link utilization. K-sMILP also controls the K-segment paths to prevent excessive divergence or too long paths. Cheng et al. [22] proposed a G-SID header compression scheme based on common prefix stitching, which reduces the length of the packet header.

As mentioned in Section I, TDSR is a multi-path deployment method. The advantage is: routing propagation is a distributed architecture, data is pushed on the stack at the entry node, and intermediate hops are forwarded according to the addresses in the stack. This not only conforms to the evolution of the network architecture, but also alleviates the problem of the number of FIB. However, in the SRv6 extension header, the IPv6 address is longer, and the address stack will occupy a larger space, which will affect the bandwidth utilization.

This paper focuses on the problem that the SR address stack occupies too much space and proposes an algorithm to reduce the number of intermediate hops. It needs to be pointed out that this paper is limited by space. It uses the coexistence of OD and TD routing as the background and TDSR as the deployment method to explain. But the proposed SR compression method is not limited to this scenario. It is also applicable to other multi-path methods, especially traffic engineering and network planning such as SD-WAN+SR, RSVP+MPLS, etc. [23][24]. The common feature of these deployment schemes is that OD routing is the basic guarantee. On this basis, various optimization goals of QoS link weights or secure hierarchical topology are added [25][26].

III. SRV6 COMPRESSION ALGORITHM

The meaning of terms is explained in Table I. Multi-path routing based on OD routing has many practical applications, such as congestion avoidance forwarding [10] and bypassing low-security ASs [9], etc. The basic idea of these applications is to select the optimal path based on the OD logical topology.

Here is an example of congestion avoidance forwarding [10]. As shown in Fig. 1(a), three data flows sent from sources $A, B, C$ need to be forwarded to $E$, use OSPF. In OD logical topology, the optimal path is $\{D, E\}$. This causes $\{D, E\}$ congestion, and data sent from $C$ is hoped to avoid this congestion, as shown in Fig. 1(b). According to the literature [10], TD routing Link State Advertisement (TD-LSA) will be advertised through OSPF. After $D$ receives it, a TD logical topology of $C \rightarrow D$ is generated, as shown in Fig. 1(c). Based on the basic principles of the OSPF extended draft [draft], $C$ runs the SPF algorithm on the TD logical topology, and the optimal path is $\{D, F, E\}$. In the data plane, routers $D, F, E$ need to store, maintain, and lookup TD FIB. After overlaying the original OD RIB and FIB, the overhead of these routers will increase significantly.
Table I. Key Terms

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entry Node</td>
<td>Ingress node for specific traffic</td>
</tr>
<tr>
<td>OD Routing</td>
<td>The traditional routing that uses destination prefix as the table lookup index, is the basic reachability guarantee</td>
</tr>
<tr>
<td>TD Routing</td>
<td>For a certain quality of service or information security requirements, routing with source and destination prefixes as table lookup indexes (a type of multi-path routing)</td>
</tr>
<tr>
<td>OD Logical Topology</td>
<td>To ensure reachability, a graph showing the weights of hops and links on the physical topology (such as the basic topology of OSPF)</td>
</tr>
<tr>
<td>TD Logical Topology</td>
<td>For a certain quality of service or information security requirements, the link weights are marked on the physical topology after considering the link status and other factors.</td>
</tr>
<tr>
<td>TD Forwarding Table</td>
<td>The routing directory that stores TD routing.</td>
</tr>
<tr>
<td>OD SPF Path</td>
<td>The shortest path calculated by the Dijkstra algorithm in OD logical topology (hereinafter referred to as SPF path)</td>
</tr>
<tr>
<td>TD SPF Path</td>
<td>The shortest path calculated by the Dijkstra algorithm in the TD logical topology (hereinafter referred to as the SRv6 path)</td>
</tr>
<tr>
<td>Compressed Path</td>
<td>The path where the message is forwarded according to the hop sequence (a subset of the complete hop sequence) compressed by the HCMD algorithm</td>
</tr>
</tbody>
</table>

![Fig. 1. Example of TD routing](image)

If TDPSR is deployed in the actual application, only the hops of the TD routing are pushed on the address stack. Only C (the entry node responsible for pushing the stack) is required to store, maintain, and view the TDFIB, and intermediate hops can directly forward. However, the entry node pushes \( \{C, D, F, E\} \) on the stack. Since IPv6 addresses are long, there are more routers in a large network, and the address stack occupies a lot of space, hops compression can be attempted.

A. Compression Algorithm

The notations of this paper are defined in Table II. The goal of HCMD is to calculate the minimum set of hops that must be pushed on the address stack from the difference between the SPF and SRv6 path. The path that hops forwards data according to the minimum is consistent with the full set. The basic idea is: Starting from the end of the SRv6 path, compare the SPF and SRv6 paths in reverse order hop by hop. If they are inconsistent, push the hop on the stack. If they are consistent, skip the current hop until finding an inconsistent hop.

The core ideas are as follows:

Traversing Path \( S \) in reverse order, initial \( D \) as \( D_f \), calculate from \( D_{S_i-1} \) to \( D_{S_i} \) whether Path \( D_{1-i} \) is same as Path \( S_{i-1} \).

If they are different, it means that under the cost of OD routing, the previous-hop will not forward the data to this hop as required. You need to fill \( D_{S_i-1} \) into List \( Z \), and replace \( D_{S_i-1} \) with the new \( D_f \) to continue traversing.

If they are the same, it means that \( D_{S_i-1} \) will send data to \( D_{S_i} \), but the hop before \( D_{S_i-1} \) may not also forward the data. So the search interval \( J_S \) needs to be further expanded until the path Path \( D_{1-j_S} \) from \( D_{S_i-j_S} \) to \( D_{S_i} \) is different from Path \( S_{i-j_S} \). In the process of searching, it is important to note that the path from \( D_{S_i-j_S} \) to \( D_{S_i} \) meets the requirements of SRv6, which means that as long as \( D_{S_i-j_S} \) forwards the data to \( D_{S_i-j_S+1} \) can ensure that Path \( D_{1-j_S+1} \) is the same as Path \( S_{i-j_S} \). So you need to fill \( D_{S_i-j_S+1} \) into List \( Z \), replace \( D_{S_i-j_S+1} \) with the new \( D_f \), reset \( J_S \) to 1, and continue traversing until \( S \) is \( D_{S_i-j_S} \).

The compression hop can be obtained by Algorithm 1.

Table II. Key Notations

<table>
<thead>
<tr>
<th>Notation</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>( S ) ( . ) ( D )</td>
<td>The source and purpose of the data</td>
</tr>
<tr>
<td>( N ) ( . ) ( M )</td>
<td>The total number of hops in SRv6 sequence, the total number of hops in the same hops sequence</td>
</tr>
<tr>
<td>( D_f )</td>
<td>The endpoint finds by the SPF algorithm</td>
</tr>
<tr>
<td>( D^* )</td>
<td>The latest hop entry into the List ( Z )</td>
</tr>
<tr>
<td>tc</td>
<td>Number of comparisons</td>
</tr>
<tr>
<td>( J_S ) ( \cdot ) ( J_E )</td>
<td>The hop interval of SRv6 sequence, the hop interval of the same hops sequence</td>
</tr>
<tr>
<td>( D_{D_f} ) ( , ) ( D_{S_i} ) ( , ) ( D_{E_j} )</td>
<td>SPF hops sequence, SRv6 hops sequence, identical hops sequence</td>
</tr>
<tr>
<td>Path ( D_{1-i} ) ( , ) Path ( S_{i-1} ) ( , ) Path ( Z_{i-j} )</td>
<td>( S \rightarrow D ), SPF path, SRv6 path, compressed path</td>
</tr>
<tr>
<td>Path ( D_{1-i} ) ( , ) Path ( S_{i-1} ) ( , ) Path ( Z_{i-j} )</td>
<td>Between hop ( i ) and ( j ), SPF path, SRv6 path, compressed path</td>
</tr>
<tr>
<td>List ( Z ) ( , ) List ( Z ) ( , ) List ( Z ) ( , ) List ( Z )</td>
<td>SPF hops sequence, SRv6 hops sequence, same hops sequence, compressed hops sequence</td>
</tr>
</tbody>
</table>
Algorithm 1: HCMD calculation process

Input: Path_D, Path_S
Output: List_z
Traversing Path_S, get N, List_z;
i = N; jS = 1;
while D_D_i !→ S
Traversing Path_D_i−1 from D_D_i−1 to D_S_i;
if Path_S_i−1 = Path_D_i−1 and i > jS
jS = jS + 1;
Traversing Path_D_j−1 from D_S_j−1 to D_S_i;
if Path_S_j−1 = Path_D_j−1 and i > jS
continue;
else if Path_S_i−1 = Path_D_i−1 and i > jS
Fill D_S_i−1 to List_z;
i = i − jS + 1; jS = 1;
else break;
else if Path_D_i−1 = Path_D_i−1 and i > jS
Fill D_S_i−1 to List_z;
i = i − jS; jS = 1;
else break;
The complexity analysis of HCMD: suppose n is the total number of nodes, m is the number of hops in List_z. It can be calculated that the time complexity of HCMD is O(m * n^2). The auxiliary space is of constant order, and the space complexity of HCMD is O(1).

The following example illustrates the operating process of HCMD: Assume that there is a multi-path forwarding requirement between S and D in Fig. 2(a), for S, there are both SPF and SRv6 paths.

![Diagram](image_url)

Fig. 2. Example of HCMD running

First traverse Path_S, you can get the hop sequence List_z as (S, a, f, b, c, h, i, e, D), the next steps are as follows.

<table>
<thead>
<tr>
<th>Find</th>
<th>Result</th>
<th>Judgment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Path_D_1</td>
<td>{e, D}</td>
<td>Consistent with Path_S_1</td>
</tr>
<tr>
<td>Path_D_2</td>
<td>{e, D}</td>
<td>Consistent with Path_S_1</td>
</tr>
<tr>
<td>Path_D_3</td>
<td>{h, e, D}</td>
<td>Path_S_3 is {h, i, e, D}, fill i into List_z</td>
</tr>
<tr>
<td>Path_D_4</td>
<td>{h, e, l}</td>
<td>Path_S_4 is {h, l}, fill h into List_z</td>
</tr>
<tr>
<td>Path_S_5</td>
<td>{c, h}</td>
<td>Consistent with Path_S_5</td>
</tr>
<tr>
<td>Path_D_6</td>
<td>{h, c, h}</td>
<td>Consistent with Path_S_6</td>
</tr>
<tr>
<td>Path_D_7</td>
<td>{f, c, h}</td>
<td>Consistent with Path_S_7</td>
</tr>
<tr>
<td>Path_D_8</td>
<td>{a, c, h}</td>
<td>Path_S_8 is {a, f, c, h}, fill f into List_z</td>
</tr>
<tr>
<td>Path_D_9</td>
<td>{a, f}</td>
<td>Consistent with Path_S_9</td>
</tr>
<tr>
<td>Path_D_10</td>
<td>{S, a, f}</td>
<td>Consistent with Path_D_10</td>
</tr>
</tbody>
</table>

The hops in List_z are f, h, i, as shown in Fig. 2(b), which is reduced by 2/3 compared to List_s. It can be verified that the data is forwarded in the same path as List_z and List_s:

Path_Z_s(S → f is {S, a, f}, consistent with Path_S_s,f);
Path_Z_f,h off → h is {f, b, c, h}, consistent with Path_S_f,h;
Path_Z_h,i of h → i is {h, i}, consistent with Path_S_h,i;
Path_Z,i,D of i → D is {i, e, D}, consistent with Path_S_i,D;
Path_Z is consistent with Path_S.

B. Verify the Consistency between the Compressed Path and the Original Path

From S to D, the hops sequence obtained by the TD logical topology is List_z = {D_S_0, D_S_1, ..., D_S_m}. The hops sequence after HCMD compression is List_z, List_z is empty initially. Path_D_{m} represents the route from D_S_{m} to D_S_0 according to the SPF algorithm. Path_S_{m} represents the route from D_S_{m} to D_S_0 according to SRv6. Path_Z_{m} represents the route from D_S_{m} to D_S_0 according to List_z. Initially t_c = 0, D^t = D.

$$R_{result}(List_z) = \begin{cases} \text{push(D_{S_{m−1}}), } t_c = 0 \\ \text{push(D_{S_{m−1−1}}), } t_c \neq 0 \end{cases}$$

When the hops satisfy (1), bring in (2) for corresponding operations. push(X) represents pushing hop X into the List_z, and D^t = X. t_c = 0. Where i \in [0, N].

Proposition: The path forwarded according to List_z and List_s is consistent, that is, the path consistency of HCMD.

Use mathematical induction to prove:

Suppose the hop sequence compressed by HCMD is List_z = {D_S_1, D_S_2, ..., D_S_m}.

When m = 1
• If t_c = 0, D_S_i is pushed on the stack.
I.e. Path_Z_{S,D_S_1} = Path_S_{S,D_S} = Path_D_{S,D_S_1}
Path_Z_{D,S_1,D} = Path_S_{D,S_1,D}
Then Path_Z = Path_S

• If t_c ≠ 0, D_S_i is pushed on the stack.
I.e. Path_Z_{S,D_S_1} = Path_S_{S,D_S} = Path_D_{S,D_S_1}
Path_Z_{D,S_1,D} = Path_S_{D,S_1,D} = Path_D_{D,S_1,D}
Then Path_Z = Path_S

Assuming m = k(k ≥ 1), the compression is correct.
Then Path_Z = Path_S

When m = k + 1
Path_Z = Path_Z_{S,D,S_k} + Path_Z_{D,S_k}.

Path_Z_{S,D,S_k} + Path_Z_{D,S_k}.
• If t_c = 0, both D_S_k and D_S_{k+1} are pushed on the stack, D_S_k and D_S_{k+1} are continuous and at the end of List_s,
Then Path_Z_{D,S_k,D} = Path_S_{D,S_k,D}
• If \( tc = 0 \), \( \mathcal{D}_S \) is pushed on the stack and when \( tc \neq 0 \), \( \mathcal{D}_S \) is pushed on the stack,
\( \mathcal{D}_S \) and \( \mathcal{D}_S \) are adjacent in \( \text{Path}_D \),
\[ \text{Path}_{Z_{D,S_k+d}} = \text{Path}_{D,S_k+d} = \text{Path}_{D,S_k+d} \]
Then \( \text{Path}_{D,S_k+d} = \text{Path}_{D,S_k} \)
• If \( tc \neq 0 \), \( \mathcal{D}_S \) is pushed on the stack and when \( tc = 0 \), \( \mathcal{D}_S \) is pushed on the stack,
\( \mathcal{D}_S \) is the last in \( \text{List}_5 \).
\[ \text{Path}_{Z_{D,S_k}} = \text{Path}_{D,S_k} \]
\[ \text{Path}_{Z_{D,S_k}} = \text{Path}_{D,S_k} \]
\[ \text{Path}_{Z_{D,S_k}} = \text{Path}_{D,S_k} \]
\[ \text{Path}_{Z_{D,S_k}} = \text{Path}_{D,S_k} \]
So the original formula = \( \text{Path}_{Z_{D,S_k}} + \text{Path}_{Z_{D,S_k}} = \text{Path}_{Z_{D,S_k}} + \text{Path}_{Z_{D,S_k}} = \text{Path}_{S} \)

In summary, it can be proved that the forwarding paths according to \( \text{List}_2 \) and \( \text{List}_5 \) are consistent.

C. “Link as the Label” and “Initial Label Discontinuity”

SRv6 has two different situations in practical applications, they are “Link as the label” and “Initial label discontinuity”. Due to limited space, this article only gives examples of these two situations:

1) HCMD is also applicable to label links stored in the label stack, because saving the label link is equivalent to saving the labels of the routers at both ends of the link.

Refer to the example in Fig. 2(a), after traversing the SRv6 path, you can get the link sequence \( \text{List}_5 \) as \{\( S, a \), \{\( a, f \), \( f, b \), \{\( b, c \), \( c, h \), \{\( h, i \), \( i, e \), \{\( e, D \)\}\}\}\}\}\}.

The link \( \{a, f\} \), \( \{h, i\} \) needs to be pushed on the stack, and the algorithm is correct.

2) The situation of "discontinuous label nodes" may come from the user's independent route selection. It only stipulates that the data passes through some designated nodes, and there is no strict requirement on the path.

Refer to the example in Fig. 2(a), and suppose the data flow is required to pass through \( f, c, i \),

The hops pushed on \( \text{List}_2 \) have \( f, i, l \), and the algorithm is correct.

D. HCMD Algorithm Optimization

If the scale of the topology is large, the number of hops will increase significantly. If HCMD is used from the destination of the whole path, the computational overhead is unacceptable. In the existing studies, the TD SPF path and the OD SPF path mostly overlap, and only a few hops are different. For example, common congestion avoidance, link failure, delay limitation, etc. [9][27]. This section further optimizes the algorithm and proposes a simplified HCMD (SHCMD).

SHCMD needs to traverse \( \text{List}_5 \) and \( \text{List}_5 \) to get the same section, so it is only suitable for the case where the SRv6 path is long and hops in \( \text{List}_5 \) is continuous. If the SRv6 path is short, SHCMD will increase the running time; In addition, the discontinuous hops in \( \text{List}_5 \) result in incomplete hops in \( \text{List}_2 \) and the calculated \( \text{Path}_Z \) is inaccurate, so the simplified algorithm is not applicable for all scenarios.

The basic idea of SHCMD is to traverse the SPF and the SRv6 path first, get the same hops on two paths, and only use HCMD between the same hops that are not continuous on the SRv6 path.

The core ideas of SHCMD are as follows:

1) Traverse \( \text{Path}_D \) and \( \text{Path}_S \), get \( \text{List}_5 \) and \( \text{List}_5 \), and store the same hops in \( \text{List}_5 \). Traverse \( \text{List}_5 \) in reverse order, starting from the last identical hop \( E_{S} \), and find \( \mathcal{D}_S \) corresponding to \( E_r \) in \( \text{List}_5 \). If \( E_r \) and \( E_r \) are adjacent in \( \text{List}_5 \), \( E_r \) moves forward until \( E_r \) is not adjacent, let \( E_r \) be the initial \( D_f \), from \( \mathcal{D}_S \) to \( \mathcal{D}_S \) to calculate whether \( \text{Path}_{D_1} \) is the same as \( \text{Path}_{S_1} \).

If the two paths are different, push \( \mathcal{D}_S \) on the address stack, and judge whether \( \mathcal{D}_S \) is \( \mathcal{D}_E \), if not, replace \( \mathcal{D}_S \) with the new \( D_f \) to continue traversing. Otherwise, \( \mathcal{D}_E \) moves forward.

If the two paths are the same, check whether \( \mathcal{D}_S \) is \( \mathcal{D}_E \), if not, further expand the search interval \( J_s \) until the path \( \text{Path}_{D_{1}} \) is different from \( \mathcal{D}_S \) to \( \mathcal{D}_S \), fill \( \mathcal{D}_S \) to \( \mathcal{D}_S \) into \( \text{List}_2 \). Otherwise, \( \mathcal{D}_E \) moves forward.

Traverse \( \text{List}_2 \) in reverse order until \( S \) is \( D_{E_{1}} \).

The compression hop can be obtained by Algorithm 2.

**Algorithm 2: SHCMD calculation process**

**Input:** \( \text{Path}_D \), \( \text{Path}_S \)

**Output:** \( \text{List}_5 \)

1. Traverse \( \text{Path}_D \), get the hop sequence \( \text{List}_5 \).
2. Traverse \( \text{Path}_S \), get the hop sequence \( \text{List}_5 \).
3. Fill the same hop of the two sets of hop sequences into \( \text{List}_2 \).
4. Traverse \( \text{List}_2 \), get \( M \),
5. \( i = M, J_s = 1, J_k = 0 \);
6. **while** \( D_{E_{1}} \) != \( S \) **do**
7. **if** \( E_r \) and \( D_{E_{1}} \) are not continuous in \( \text{List}_5 \)
8. **if** \( D_{E_{1}} > J_s \)
9. Run HCMD between \( D_{E_{1}} \) and \( D_{E_{1}} \)
10. **else** \( i = i - 1, J_s = 1, J_k = 0 \);
11. **else** \( i = i - 1 \);

![Table 1](https://example.com/table1.png)

**Table 1**

<table>
<thead>
<tr>
<th>Find</th>
<th>Result</th>
<th>Judgment</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{Path}<em>{D</em>{1}} )</td>
<td>( i, e, D )</td>
<td>Consistent with ( \text{Path}<em>{S</em>{1}} )</td>
</tr>
<tr>
<td>( \text{Path}<em>{D</em>{2}} )</td>
<td>( c, d, e, D )</td>
<td>( \text{Path}<em>{S</em>{2}} ) if ( (c, h, i, e, D) ), fill ( i ) in ( \text{List}_2 )</td>
</tr>
<tr>
<td>( \text{Path}<em>{D</em>{l}} )</td>
<td>( S, a, f )</td>
<td>Consistent with ( \text{Path}<em>{D</em>{l}} )</td>
</tr>
</tbody>
</table>
The complexity analysis of SHCMD: suppose \( n \) is the total number of identical nodes between a segment that is not consecutive, \( m \) is the number of hops of \( List_e \) in the segment. It can be calculated that the time complexity of SHCMD is \( O(m + n^2) \). However, \( m \) and \( n \) of SHCMD are much smaller than those of HCMD. The space complexity of SHCMD is also \( O(1) \).

2) In multi-path routing applications, such as secure hierarchical trusted forwarding [26], if the boundaries of the areas with different security levels are clear [30], the link can be marked in advance. The key nodes in the links of different security levels are selected and pushed on the stack as representatives. In this way, the forwarding path can meet the demand.

Due to space limitations, this discussion will not be expanded here, and further research is needed in the future.

IV. PERFORMANCE EVALUATION

To demonstrate the optimization effect of TDSR on the number of FIB tables, we publish TD routing requirements in a self-made topology randomly. Then we record the TD routing increments by using TDSR and TD routing.

In order to reflect the effect of HCMD on hops compression, we compare HCMD with G-SID [22] and K-sMILP [21]. After using G-SID and K-sMILP, we can go on using HCMD. However, since there is no mutually exclusive algorithm with HCMD, this paper attempts to compare these algorithms separately.

The basic scheme of the simulation is to use the Python and NetworkX module to generate a 300-hops topology (scale of area). The experimental platform is a 64-bit Win10 server, x64-bit processor, 32G memory, Intel(R) Xeon(R) Silver 4110 CPU, 2.10 GHz. The configuration is shown in Table III.

![Fig. 3. Example of SHCMD running](image)

**TABLE III. PARAMETER SETTING**

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of nodes</td>
<td>300</td>
</tr>
<tr>
<td>Number of SRv6 hops</td>
<td>5~50</td>
</tr>
<tr>
<td>Cost of working port</td>
<td>1~10</td>
</tr>
<tr>
<td>Cost of congesting port</td>
<td>100</td>
</tr>
</tbody>
</table>

A. The Optimization Effect of TDSR

To demonstrate the optimization effect of TDSR on reducing router FIB tables, we publish TD routing requirements in the simulation topology and recorded the TD routing increments. As shown in Fig. 4, the published requirement is increased by 5 each time, and the number of the TD routing increase by about 10-30. Due to the randomness of requirement, the growth trend is not regular, but overall, the number of FIB has increased significantly. The increment is less than 10 by using TDSR. This means that when there are a large number of multi-path routing requirements, TDSR can significantly reduce router FIB tables and can be distributed in different entry nodes.

![Fig. 4. Increment of the TD routing](image)
B. Bandwidth Gains of HCMD

To reflect the compression effect of HCMD, we run HCMD, G-SID, and K-sMILP on the same SRv6 path. In Fig.5 and Fig.6, hops are increased by 5 each time, and the data is increased by 80B each time. HCMD is less affected by the number of hops, and the compression effect is significant. The compressibility is up to 70.4%. 50% of the compressed data don’t exceed 100 B, 40% is between 100-200 B, and the maximum is only 220B. It shows that HCMD can be applied to networks of different scales and alleviate bandwidth waste effectively. In the comparison method: the compression effect of G-SID depends on the proportion of continuous same prefixes. If the prefixes change frequently, the compressibility will be poor. In this experiment, the compressibility of G-SID is not higher than 50%, and the data size is close to 400B. K-sMILP don’t compress hops when the Path_S is short and the number of segment lists don’t exceed the upper limit. The compressibility is up to 60% in large-scale networks, and the compressed data is about 250B. However, the intermediate hops need to access the TD FIB table, and there are still many TD routing.

After the above two methods are compressed, HCMD is used again for compression, which can significantly improve the result. The compressibility of G-SID in combination with HCMD is increased by 40%, and the compressed data is decreased by 300B. The compressibility of K-sMILP in combination with HCMD reaches 90%, saving about 200B. This shows that HCMD can combine with other hop compression methods, and increase optimization effects accordingly.

C. Overhead of HCMD and SHCMD

To reflect the optimization of SHCMD over HCMD, in this section we test the running time and memory overhead. As shown in Fig. 7, the running time of HCMD increased with the number of SRv6 hops. When the number of hops is less than 10, it is less than 1ms. With each additional hop, it increases by 2ms-4ms. When the number of hops reaches 50, it is close to 100ms. When SHCMD have fewer hops, compression time is close to that of HCMD. With the increase of hops, it shows that 50% of the running time can be saved. In terms of computing overhead, as shown in Fig. 8, when hops are less than 15, the computation only uses about 1MB. As the number of hops increases, it increases significantly, up to 10MB. The optimization effect of SHCMD increases with the number of hops, which can save 3MB-5MB on average. This indicates that HCMD has good performance, but the increase of hops will impact running time and computing overhead. SHCMD can reduce these effects and improve the capabilities significantly. The more hops, the more optimization effect.

V. CONCLUSION AND PROSPECT

To solve the problem of increasing the FIB table in multi-path routing deployment, this paper takes TD routing as an example and presents a deployment method combining SRv6. It can reduce the number of routes by about 69.2%. To reduce the bandwidth occupancy of SRv6, this paper proposes a hop compression algorithm based on the difference between the SPF path and the SRv6 path (HCMD). The simulation result shows that HCMD can compress 70.4% of hops, and can also be combined with existing compression methods. Because the SRV6 path is long and has many repeated hops, we propose simplified HCMD (SHCMD). The experimental result shows that SHCMD reduces 50% running time and 40% computing overhead.

In the future, we will focus on the original intention of multi-path routing, reduce the scale of problems and further simplify the compression algorithm.

ACKNOWLEDGMENT

This work is supported by the National Key Research and Development Program of China under Grant 2018YFB1800302, 2020YFE0200500 and Beijing Natural Science Foundation (4212018, KZ201810009011), and in part
by the Basic Scientific Research Projects of Beijing Municipal Commission of Education.

REFERENCES


