Received 9 December 2023, accepted 12 February 2024, date of publication 22 February 2024, date of current version 6 March 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3368909

RESEARCH ARTICLE

An SDN-Based Flow Table Encoding Approach for Resource and Efficiency Optimization in Topic-Based Pub/Sub Systems

YU ZHOU[1] **AND YANG ZHANG**[2]
[1]College of Computer, Beijing University of Posts and Telecommunications, Beijing 100876, China
[2]State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China
Corresponding author: Yang Zhang (YangZhang@bupt.edu.cn)

ABSTRACT With the rapid development of software-defined networking (SDN), SDN-Based multi-level flow table architectures are always employed to address issues such as QoS, security policies, and matching efficiency. The rise of semantic communication has also sparked researchers' interest in semantic information and its utilization. Many studies on semantic representation and semantic summarization have emerged. This study takes a new perspective to reduce the number of table entries by utilizing the semantic relationships implied in the topic tree in the topic-based pub/sub systems and introduces the concept of semantic aggregation. Semantic aggregation of table entries can work with multi-level flow table architecture to reduce the number of table entries while ensuring the correct delivery of streams. We propose a semantic-based table entry encoding algorithm to implement our idea and conduct several experiments to examine its performance. The experiment results demonstrate that our algorithm can achieve high space and efficiency optimization rate in a short encoding time.

INDEX TERMS Semantic aggregation, multi-level, SDN, flow table.

I. INTRODUCTION

Due to the high availability and efficiency, topic-based pub/sub systems are still heavily used in areas such as cloud computing and IoT. However, since messages on the same topic published by different publishers in real networks are different with each other, it is necessary for the subscribers to receive all the topic messages published by different source nodes. So, we can uniquely identify a particular stream by the publisher and the topic it belongs to and select the flow table rule with these two fields in packets as the matching fields. When there are a significant number of topics and multiple publish sources for each topic, it often leads to a great number of flow table entries and packet matching counts.

In recent years, existing network architectures have faced challenges due to the increasing diversity of user requirements and complexity of the network situation. SDN is widely used as the next-generation network architecture owing to its support for centralized network control and programmability, which can provide a flexible network configuration and diversified transport services. SDN technology allows us to control the switch pipeline as well as the flow table architecture. In many areas, it's prevalent to optimize look-ups using multi-level tables or indexes, such as multi-level page tables, tree indexes in database storage engines and so on [1]. Similarly, multi-level flow tables are often used to optimize efficiency during data forwarding in networks [2], [3], [4]. The nature of SDN makes the implementation of multi-level flow tables easy. Therefore, we choose to optimize the efficiency of topic-based pub/sub systems by adopting the multi-level flow table architecture based on SDN.

Although the multi-level flow table architecture can improve the matching efficiency, massive flow table entries will still occupy the limited space of the switches and consume considerable resources for maintenance. Therefore, reducing the number or size of table entries is essential. With ongoing research on semantics and the upsurge of semantic

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

communication [5], [6], [7], optimizing pub/sub systems by focusing on and exploiting semantics is a hot research topic. Here, semantic publication/subscription [8], [9], semantic summarization [10], [11] and semantic extraction or compression [12] are its common ways, which are usually used to improve the user experience or reduce the resource consumption. Fully considering the characteristics of topic-based pub/sub systems, we propose the idea of topic-based semantic aggregation to reduce the number of table entries in the data plane. Semantic aggregation aims to merge the table entries with identical or virtually identical sets of forwarding ports.

In this paper, we propose an encoding method which is based on the semantics of topics to reduce the number of table entries and build a multi-level flow table architecture. We encode topics with IPv6 multicast addresses and reflect the semantic relationships of different topics in the topic tree through their prefix encoding. So, the topics will inherit the non-zero prefixes of their parent topics. At the same time, the topic fields in the flow rules are prefix-matched by the length of their non-zero parts, which means that in some cases, a stream of someone topic may hit the same table entry as its parent topic. Based on this idea, we designed several detailed algorithms to reduce the table entries by semantic aggregation and organize them into multilevel stream table structures.

Overall, our paper provides the following contributions:

- Semantic aggregation is proposed as an idea to optimize topic-based pub/sub systems by merging the forwarding of different streams. Compared to the previous optimization approaches, it starts from the semantic view of the topics and reduces space occupation by shrinking the number of match fields and table entries, providing a creative strategy to decrease resource consumption and perform latency optimization during the transmission.
- We design a P4-based multi-level flow table architecture where different flow tables jump to the next level based on semantics. We dynamically generate a multi-level flow table structure with different depths based on jumping delay, the size of flow tables, and predicted traffic conditions to optimize the number of table entries and the matching counts.
- Based on the experiments for several metrics, the results demonstrate that this study's algorithm slashes the total number of table entries and the average matching time.

The rest of this article is structured as follows. Section II presents the related works. Section III describes the encoding way of the topic field and introduces the idea of semantic aggregation on table entries in detail. Section IV presents the multi-level flow table architecture in our study. Section V presents the details of the proposed algorithm. Section VI discusses the experiment results. Section VII concludes the paper.

II. RELATED WORK

SDN has evolved significantly over time, with the most common technologies including OpenFlow, P4, and SD-WAN. In addition to some research on the SDN ecosystem and its security and performance issues, most studies are committed to addressing issues such as system performance, service, and security. SDN has been researched and applied in many scenarios, including cloud computing, the Internet of Things, and data centers. In pub/sub systems, SDN is often used as the means to address issues such as latency, QoS, and network resources [13], [14], [15], [16].

For optimization on the flow table side, some SDN-Based studies focus on reducing the number of table entries and the packet matching time. By constructing the multilevel table architectures, the proposed frameworks in some previous studies [2], [3], [4] have achieved better results in matching time optimization. Moreover, in some special scenarios, the multi-level table structure can be used to address some concurrent access issues [17]. In addition, some studies have been conducted to reduce the number of table entries by combining SDN with other technologies. Based on the network programmability that is provided by OpenFlow, Huang et al. [18] propose a novel segment routing architecture with improved data plane to reduce the overhead of additional flow entries. In addition to optimizing the number of flow table entries based on SDN, some researches [19], [20] have applied several other techniques to reduce the number of table entries, such as segment routing and traffic engineering. Moreover, the entry size is also been considered as a point of optimization. In work [2], an approach of bloom filter is used to achieve high space optimization rate. Here, the bloom filter is used to compress and limit the memory occupation of each flow entry.

Envisioned as one of the critical technologies in future communications, research on semantic communication is mushrooming. Most of them revolve around semantic representation to shrink the amount of information for efficient transmission. Knowledge graphs (KL) [21], [22], information-theoretic approaches, and machine/deep learning (DL) [23] are often used as the methods to construct and compress semantic information. In addition to text transmission, it also achieved better results in the efficient transmission of data such as images [24], [25], speech [26], and video [27]. It aims to express semantic information of the user's concern with less data while ensuring the recovery accuracy.

As for the application of semantics on pub/sub systems, the existing researches mainly utilize the semantics in the publishing/subscription phase and data representation. Some existing pub/sub systems implement semantic publication/subscriptions in ways which typically include SPARQL [8], graph subscription/publication [9], and key-value pairs. The common examples are content-based and graph-based pub/sub systems. Besides, by providing high-level interpretation of entities through semantic summarization, some pub/sub systems [10], [11] improve the readability of messages and make the system more user-friendly. Like the existing semantic communication approaches, semantic extraction and compression aims to avoid the transmission of redundant information and optimize the network resources occupation [12]. Unlike applying semantics in the matching phase and the message representation phase, our study performs semantic aggregation on table entries at the switches to optimize the number of table entries during the delivery of messages. This is a new perspective on the use of semantics.

III. SEMANTIC AGGREGATION

A. PREPARATION

In a topic-based pub/sub system, messages are categorized by topics, and users can subscribe to messages on any topic. In some topic-based pub/sub systems, topics are organized by their semantic relationships and are available for users to publish and subscribe in the topic tree. The mapping of the message's necessary publish source and topic information to the packet's ipv6 header is shown in Fig. 1.

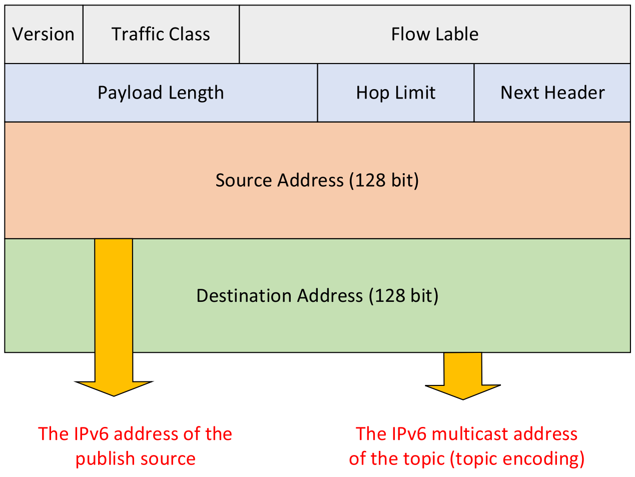
Version
Traffic Class
Flow Lable
Payload Length
Hop Limit
Next Header
Source Address (128 bit)
Destination Address (128 bit)
The IPv6 address of the publish source
The IPv6 multicast address of the topic (topic encoding)

FIGURE 1. Mapping of necessary attributes of the message to the ipv6 header of the packet.

As shown in Fig. 1, the source address and destination address in the ipv6 header are used to identify the publishing source and topic of the message. In the data plane, switches can choose specific paths for different data streams based on these two fields.

Semantic aggregation aims to aggregate the forwarding table entries of multiple streams, whose topics often have parent-child relationships or belong to the same topic, into a single table entry. Such an approach produces fewer table entries and requires less memory space.

To realize semantic-based table entry aggregation, we first need to specify the matching fields of the table entries. For any independent users, streams on the same topic from different publishers are distinct and required to be received by them. Therefore, only the combination of source and topic can uniquely identify a stream. For a table entry that has not performed any aggregation operations, its match fields should contain both the publish source and topic identifier.

In addition, we need to design a topic field encoding method that reflects the semantic relationships among topics.

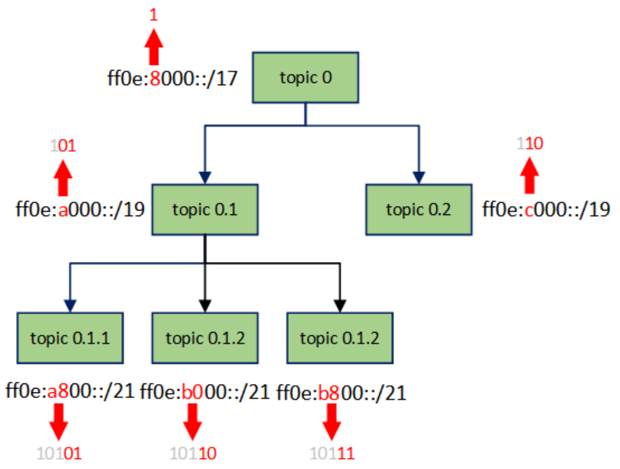

FIGURE 2. An example for topic encoding.

So, we adopt the prefix encoding shown in Fig. 2. We can see that a non-zero encoding prefix of someone's topic is also the encoding prefix of all its child topics. For the topic field in the table entry, we only need to set the match length to the length of the non-zero prefix so that the table entry can accurately match the messages belonging to a specific topic. If we need to aggregate forwarding entries of these topics and have messages belonging to different topics forwarded from the same port collection, we only need to use a new table entry that takes the encoding of the nearest public ancestor topic of all the aggregated topics as the value of the topic field.

B. TABLE-ENTRY SEMANTIC AGGREGATION

Before specifying the idea of semantic aggregation of table entries in this paper, we first introduce the aggregation of several arbitrary table entries. As shown in Fig. 3, $S1$, $S2$, ... are the ipv6 addresses of different publisher node. In (a), table aggregation shrinks the size and number of flow table entries by removing the source ipv6 address as a match field and combining table entries with the same topic into a single table entry. In (b), by modifying the matching field and the prefix match length, we can aggregate table entries with different topics but with the same forwarding ports into a single table entry, further reducing the number of flow table entries. Regardless of which type of table entry aggregation is implemented, the data streams that need to be matched by the table entries before aggregation can hit the table entry after aggregation, ensuring at least their correct propagation along the pre-planned paths. It should be noted that such table entry aggregation may also cause traffic to propagate to links outside of the plan, requiring us to intercept it with appropriate methods.

Here, we divide the procedure of semantic aggregations into two steps, which are the aggregation of table entries with the same topic and the aggregation of table entries with different topics. The target table entry takes the union of the forwarding ports sets of the original table entries as

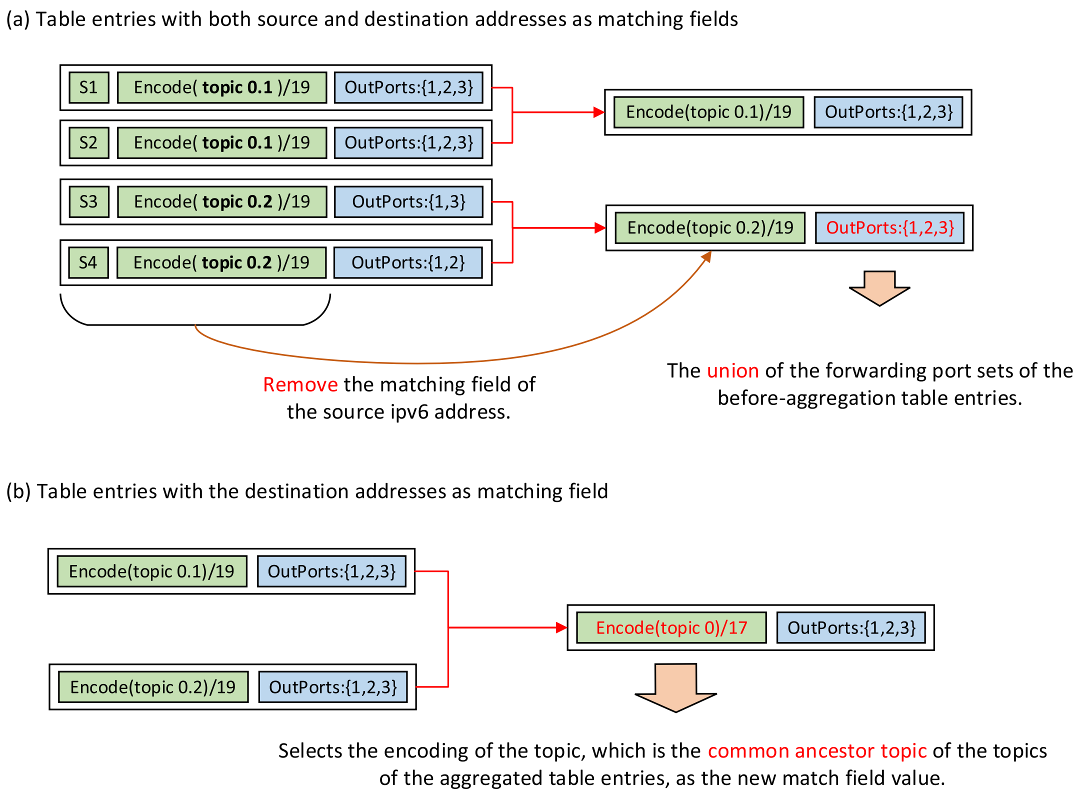
(a) Table entries with both source and destination addresses as matching fields
S1
Encode(topic 0.1)/19
OutPorts:{1,2,3}
S2
Encode(topic 0.1)/19
OutPorts:{1,2,3}
S3
Encode(topic 0.2)/19
OutPorts:{1,3}
S4
Encode(topic 0.2)/19
OutPorts:{1,2}
Encode(topic 0.1)/19
OutPorts:{1,2,3}
Encode(topic 0.2)/19
OutPorts:{1,2,3}
Remove the matching field of the source ipv6 address.
The union of the forwarding port sets of the before-aggregation table entries.
(b) Table entries with the destination addresses as matching field
Encode(topic 0.1)/19
OutPorts:{1,2,3}
Encode(topic 0.2)/19
OutPorts:{1,2,3}
Encode(topic 0)/17
OutPorts:{1,2,3}
Selects the encoding of the topic, which is the common ancestor topic of the topics of the aggregated table entries, as the new match field value.

FIGURE 3. **An example of aggregations on any table entries.**

its set of forwarding ports. Semantic aggregation of table entries with the same topic is well understood. Streams with the same topic tend to reach the same destinations and thus have nearly the same forwarding behavior on some nodes. It makes the forwarding ports of the streams after aggregation nearly identical to the original ones, which is advantageous for the entries' aggregation. Meanwhile, after the semantic aggregation of entries on the same topic, the source domain will no longer be used as a match field in the table entries.

As for the semantic aggregation for entries with different topics, it is hard to guarantee that they have relatively similar forwarding ports. However, theoretically, when a user semantically subscribes to a topic, the descendant topics of this topic are usually also considered as the user's subscribed topics, and the streams belonging to these descendant topics need to be routed and delivered to that user as well. Thus, it is easy to have some streams share part of the forwarding routes and forwarding behaviors of streams belonging to its parent topic. So, we usually aggregate table entries whose topics possess ancestor-descendant relationships.

To make the forwarding behaviors of streams with different topics more similar, we make the forwarding paths more centralized by minimizing resource consumption. In the route planning phase, we divide the topics' routes into two types, namely, aggregated routes and non-aggregated routes. The specific algorithm will not be clearly described here due to space constraints. We simply denote this algorithm as the RP (Route Planing) algorithm, which computes routes based on Dijkstra's algorithm and the idea of backtracking and pruning. One of its inputs is a topic t_i (when there is only one topic as input, the output route is called as a non-aggregate route) or a sub-tree $SubTree(t_i)$ in the topic-tree which is rooted at t_i. It aims to find a shared route that covers the forwarding paths of all the streams of input topics based on the bandwidth resources available, the cost of each link, and the size of the bandwidth resource requested by the stream while somehow minimizing link usage and maximizing forwarding paths' overlap. We identify the final aggregated routes as well as the non-aggregated routes by doing the following:

Traverse the topics in the topic tree in a post-order. For the current topic t_i:

1) Compute the aggregated route $RP(SubTree(t_i))$ for t_i and its descendant topics using the RP algorithm;

2) Denote the costs of the links that $RP(SubTree(t_i))$ occupies as c_1.

3) Compute the route $RP(i)$ for topic t_i;

4) Denote the aggregate link cost of $RP(i)$ and the existing routes of t_i 's descendant topics as c_2.

5) Finally, compare c_1 and c_2. If c_2 is more than c_1, then t_i and its descendant topics share the aggregated route $RP(SubTree(t_i))$; Otherwise, the routes of t_i's descendant topics remain unchanged, and t_i uses the non-aggregated

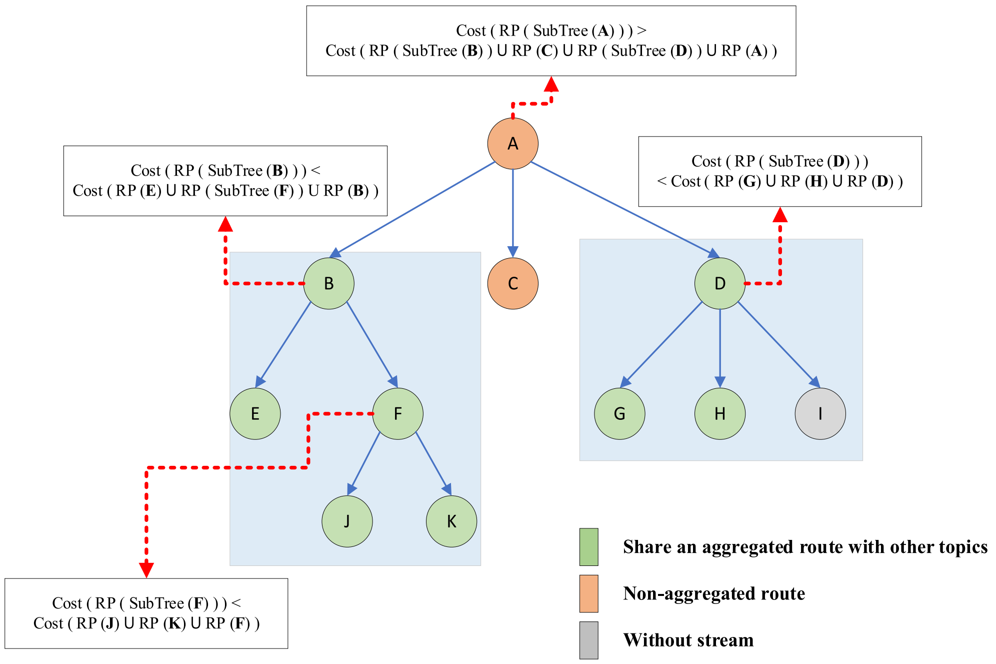
Cost (RP (SubTree (A))) >
Cost (RP (SubTree (B)) U RP (C) U RP (SubTree (D)) U RP (A))
Cost (RP (SubTree (B))) <
Cost (RP (E) U RP (SubTree (F)) U RP (B))
Cost (RP (SubTree (D)))
< Cost (RP (G) U RP (H) U RP (D))
Cost (RP (SubTree (F))) <
Cost (RP (J) U RP (K) U RP (F))
A
B
C
D
E
F
G
H
I
J
K
Share an aggregated route with other topics
Non-aggregated route
Without stream

FIGURE 4. An example of aggregated routes and non-aggregated routes.

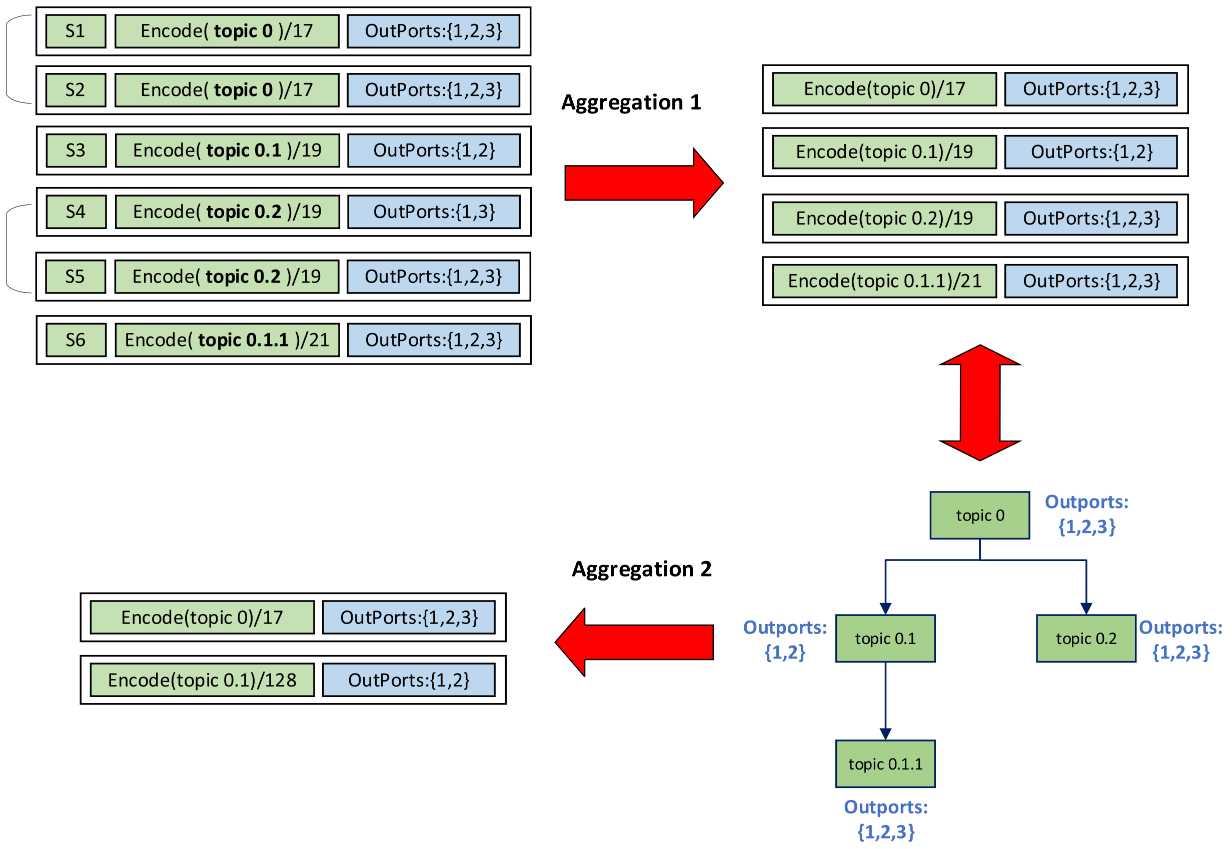
S1 Encode(topic 0)/17 OutPorts:{1,2,3}
S2 Encode(topic 0)/17 OutPorts:{1,2,3}
S3 Encode(topic 0.1)/19 OutPorts:{1,2}
S4 Encode(topic 0.2)/19 OutPorts:{1,3}
S5 Encode(topic 0.2)/19 OutPorts:{1,2,3}
S6 Encode(topic 0.1.1)/21 OutPorts:{1,2,3}
Aggregation 1
Encode(topic 0)/17 OutPorts:{1,2,3}
Encode(topic 0.1)/19 OutPorts:{1,2}
Encode(topic 0.2)/19 OutPorts:{1,2,3}
Encode(topic 0.1.1)/21 OutPorts:{1,2,3}
topic 0
Outports: {1,2,3}
topic 0.1
Outports: {1,2}
topic 0.2
Outports: {1,2,3}
topic 0.1.1
Outports: {1,2,3}
Aggregation 2
Encode(topic 0)/17 OutPorts:{1,2,3}
Encode(topic 0.1)/128 OutPorts:{1,2}

FIGURE 5. An example of semantic aggregations on table entries at a switch.

route $RP(i)$. Fig. 4 gives a simple example of solving for aggregated routes.

As shown in Fig. 4, we can obtain a set of aggregated routes suitable for semantic aggregation of table entries. When a topic's route is non-aggregated, we think its forwarding paths tend to have a low overlap with its sibling topics, ancestor topics, or child topics, which is not feasible for optimization by flow rule aggregation. An aggregated route for the topics

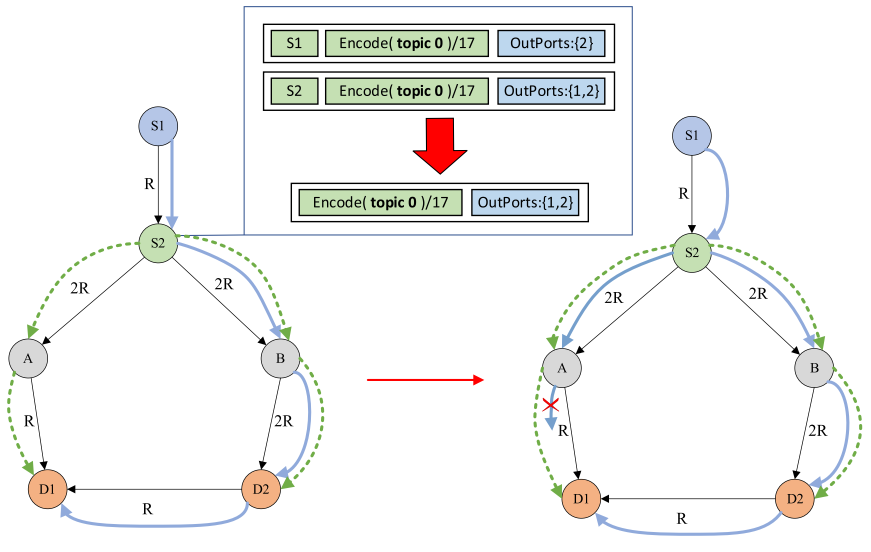
S1 | Encode(topic 0)/17 | OutPorts:{2}
S2 | Encode(topic 0)/17 | OutPorts:{1,2}
Encode(topic 0)/17 | OutPorts:{1,2}
S1
S2
A
B
D1
D2
R
2R

FIGURE 6. An example of redundant traffic.

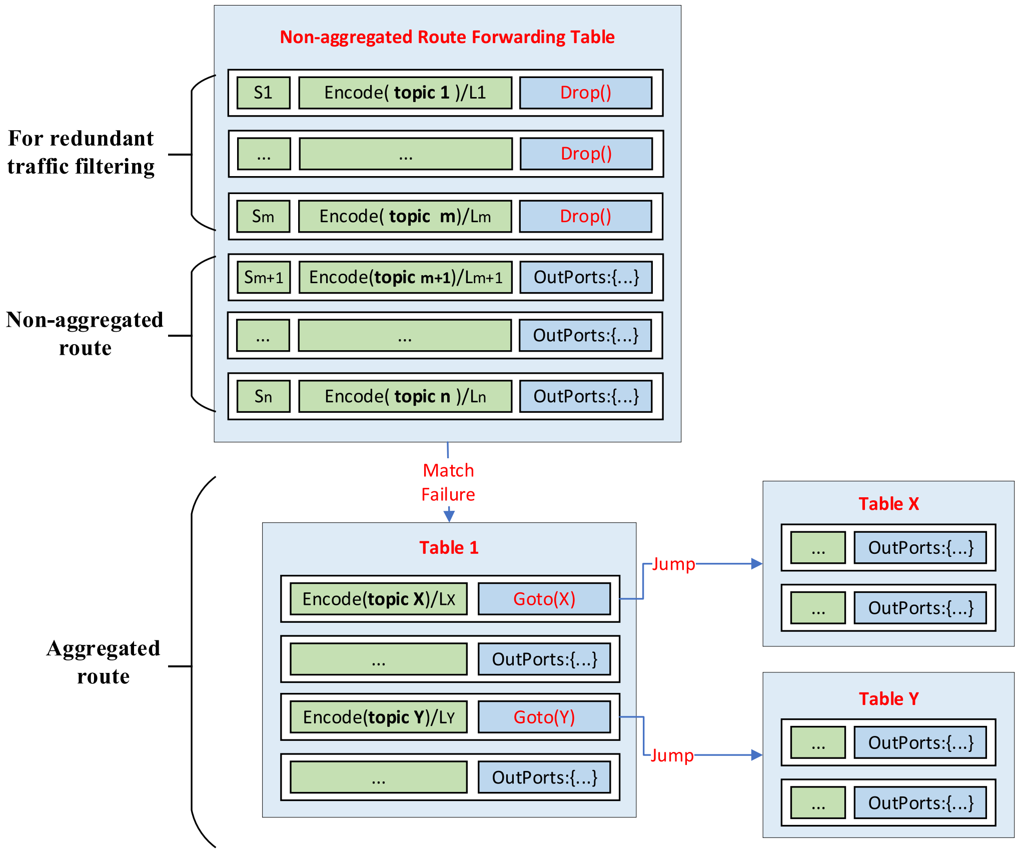
Non-aggregated Route Forwarding Table
S1 | Encode(topic 1)/L1 | Drop()
... | ... | Drop()
Sm | Encode(topic m)/Lm | Drop()
Sm+1 | Encode(topic m+1)/Lm+1 | OutPorts:{...}
... | ... | OutPorts:{...}
Sn | Encode(topic n)/Ln | OutPorts:{...}
For redundant traffic filtering
Non-aggregated route
Match Failure
Table 1
Encode(topic X)/LX | Goto(X)
... | OutPorts:{...}
Encode(topic Y)/LY | Goto(Y)
... | OutPorts:{...}
Jump
Table X
... | OutPorts:{...}
... | OutPorts:{...}
Table Y
... | OutPorts:{...}
... | OutPorts:{...}
Aggregated route

FIGURE 7. The Multi-level Flow Table Architecture.

in sub-tree $SubTree(t_i)$ contains forwarding paths of t_i and all its descendant topics. These forwarding paths have significant overlaps which facilitate the semantic aggregations on the data plane. We perform semantic aggregation of flow table entries only for topics that share the aggregated routes and generate semantically aggregated table entries according to the proposed algorithm at the switch nodes where these streams pass.

There is a simple example of semantic aggregation on table entries in Fig. 5. After aggregating the table entries of different streams with the same topic (Aggregation 1), the number of table entries has been reduced by 2. Only the topic field remains as the match field in table entries. Then, the semantic aggregations for entries with different topics (Aggregation 2) can further reduce their amounts by 2. However, the semantic aggregation in "Aggregation 1" may lead to some downstream nodes receiving redundant traffic, which requires the cooperation of multi-level flow table structure for interception to avoid its flooding in the network.

IV. MULTI-LEVEL FLOW TABLE ARCHITECTURE

Although we have used designed routing algorithms to plan paths to make our proposed coding algorithm as efficient as possible and minimize the generation of redundant traffic, there is no guarantee that we can avoid the generation of redundant traffic altogether, so we intercept this traffic by adding a filtering flow table. As shown in Fig. 6, we give a simple example to illustrate the generation of redundant traffic and then will analyze the possible solutions based on it. In Fig. 6, there are two publishers of topic0, S_1 and S_2, and two subscribers, D_1 and D_2. Both two streams request a bandwidth resource of size R. When the path shown in Fig.6 is adopted and table entries' aggregation is performed at node B, node C will receive the stream published by source S_2.

To avoid the propagation of redundant traffic as much as possible, we choose to add a flow table rule at node A to discard the stream. Therefore, we designed our multilevel flow tables architecture as shown in Fig. 7.

As shown in Fig. 7, this architecture contains two parts, the redundant traffic filtering and non-aggregated route forwarding (RFNRF) table and several aggregated forwarding tables. The RFNRF table will be placed at the first level of the pipeline, and all the packets must be matched there. Redundant traffic that needs to be filtered will be dropped here, and streams that possess a non-aggregated route are forwarded here. Packets that fail to match at this table will enter the subsequent multi-level flow tables structure for continuing the aggregated route forwarding. The flow tables are associated with each other in a tree-shape jump relationship. Moreover, we have confirmed that the multi-level flow table architecture presented in this paper can be implemented on P4 programmable switches through the P4 language.

V. PROPOSED ALGORITHM

After introducing the idea of semantic aggregation and multi-level flow table architecture in the previous two sections, we propose a semantic-based table entry encoding algorithm (Algorithm 1) to reduce the number of table entries and the matching time. It can generate the flow tables organized in a multi-level structure while semantically aggregating the table entries. Here, we assume that we have computed the table entries of all aggregated routes at individual nodes and performed source aggregation on them. These

Algorithm 1 Semantic-Based Table Entry Encoding Algorithm

```
Input: the topic tree $t$, the node id $i$, $T_1$, the set of
computed aggregated routes $R_a$.
Result: The set of all aggregated forwarding flow
tables at node $i$, $T$.
1  $T \leftarrow \{T_1\}$;
2  while the post-order traversal of $t$ not end do
3      $t_{cur} \leftarrow$ the current topic;
4      if $RP(SubTree(t_{cur}))$ not in $R_a$ or $i$ not in
         $RP(SubTree(t_{cur}))$ then
5          continue;
6
       end
7      $t_{cur} \leftarrow$ the sub-tree rooted at $t_{cur}$ in $t$;
8      $T \leftarrow T \cup MTGSA(t_{cur}, T_1)$;
9
   end
10 $T \leftarrow TSO(T, i)$;
11 return $T$;
```

table entries after the initial aggregation are stored in the currently unique aggregated forwarding flow table T_1 (Of course, the encoding algorithm may subsequently create new flow tables with T_1 as the jump source) and await further semantic aggregation.

As shown in Algorithm 1, we implement the inter-topic semantic aggregation of table entries and the generation and optimization of multi-level flow tables. We traverse all nodes in the topic tree in a post order. When all nodes in a sub-tree rooted at a topic t_{cur} share an aggregated route, we semantically aggregate the corresponding table entries of topics in this sub-tree by the Multi-level Tables Generation and Semantic Aggregation (MTGSA) Algorithm (line 8). Since the multi-level flow tables generated during semantic aggregation may not be able to achieve matching counts optimization, we also invoke the Table Structure Optimization (TSO) Algorithm to optimize the multi-level flow table structure by upward merging (line 10). The MTGSA Algorithm and TSO Algorithm are implemented in Algorithms 2 and 3, respectively, which we will introduce in the following sections.

A. MULTI-LEVEL TABLES GENERATION AND SEMANTIC AGGREGATION ALGORITHM

In Algorithm 1, we sequentially access some specific sub-trees in the topic tree by post-order traversal, where topics in the same sub-tree share an aggregated route. For any such sub-tree, Algorithm 2 performs further aggregation on the table entries, so the streams of several different topics share the same table entry without further introducing redundant traffic. However, since there may be some topics in the sub-tree whose streams do not pass through the current node, no forwarding table entries are generated for the

Algorithm 2 Multi-Level Tables Generation and Semantic Aggregation Algorithm

```
Input: a sub-tree in topic-tree $t_{cur}$, the initial aggregated forwarding table $T_1$.
Result: The set of all aggregated forwarding flow tables after table entry semantic aggregation for topics in $t_{cur}$, $T$.
1  $T \leftarrow \{T_1$;
2  while the bottom-up hierarchical traversal of $t_{cur}$ not end do
3      $t_j \leftarrow$ the current topic;
4      if there is no entry for $t_j$ in $T_1$ and $t_j$ is not a leaf-node of $t_{cur}$ then
5          $e_j \leftarrow$ a virtual table entry with $t_j$ as the topic field;
6          $P \leftarrow$ the set of outports which appears mostly in the entries of $t_j^{\prime s}$ sub topics;
7          Set the outports of $e_j$ as $P$;
8          Add $e_j to T_1$;
9
       end
10
   end
11 while the hierarchical traversal of $t_{cur}$ not end and $t_a$ is not the root of $t_{cur}$; do
12     if most entries in $E_c$ has the same forwarding ports with $e_b$ then
13         if $e_a$ and $e_b$ have the same forwarding ports then
14             Set $e_a$ as a virtual entry;
15             // The streams of topic $t_a$ may be forwarded by $e_b$ and $e_a$ may be aggregated to $e_b$
16
           else
17             Set the match length of topic in $e_a$ to 128; // The entries in $E_c$ may be aggregated to $e_b$
18
           end
19         Set the parent topic of all topics in $T_c$ as $t_b$
20     else if more than half of the entries in $E_c$ have similar forwarding ports then
21         $P \leftarrow$ the set of forwarding ports with the highest frequency of occurrence in $E_c$;
22         if $e_a$ has same forwarding ports as $P$ then
23             continue;
24
           end
25         $e_{new} \leftarrow$ a new virtual table entry with the same match configuration as $e_a$;
26         Set the forwarding ports of $e_{new}$ as $P$;
27         // The entries in $E_c$ may be aggregated to $e_{new}$
28         Set the match length of topic field in $e_a$ to 128;
29     else
           // Difficult for entries in $E_c$ being aggregated to $e_a$ or $e_b$
30         $T_a \rightarrow$ a new and empty flow table;
31         Move all the entries of $t_a^{\prime}$s descendant topics and $e_a$ to $T_a$;
32         $T \leftarrow T \cup \{T_a$;
33         $e_g \leftarrow$ a goto table entry with $t_a$ as the topic field;
34         Set the match length of $e_g$ as the length of $t_a^s$ non-zero encoding prefix;
35         Set the target table of $e_g$ as $T_a$;
36
       end
37
   end
38 return $T$;
```

corresponding topics. So, to facilitate the algorithm's processing, we propose the concept of virtual table entries. Its presence makes the process of table entry aggregation convenient and improves the efficiency of aggregation. Correspondingly, at the end of the algorithm, we remove all the virtual table entries that do not affect the correct forwarding. In other words, a virtual table entry will be deleted if no non-virtual table entry is aggregated to it.

Algorithm 3 Table Structure Optimization Algorithm

```
Input: the set of aggregated forwarding tables $T$, the node id $i$.
Result: The set of all aggregated forwarding tables after optimization, $T'$.
1  $T' \leftarrow T$;
2  while the bottom-up hierarchical traversal of $T'$ according to the jump relationship not end do
3      $T_{cur} \leftarrow$ the current table;
4      $T_p \leftarrow$ the parent table of $T_{cur}$;
5      if $T_{cur}=T_1$ then
6          break;
7      end
8      $M_{reduced} \leftarrow$ the reduced matching count at node $i$;
9      // Based on the traffic conditions on node $i$
10     if $M_{reduced} \leq 0$ then
11         Merge $T_{cur}$ into $T_p$;
12     end
13 end
14 return $T'$
```

To simplify the writing of the algorithm, we list several variables used in the code from line 11 to line 37 in a separate Table 1.

TABLE 1. The variables used in Algorithm 2.

Symbol	Definition
t_a	The current accessing topic in traversal.
t_b	The parent topic of t_a.
T_c	The set of sub-topics of t_a.
e_a	The forwarding table entry of topic t_a.
e_b	The forwarding table entry of topic t_b.
E_c	$\{e_c \mid e_c$ *is the forwarding table entry of topic* t_c and $t_c \in T_c\}$

As shown in Algorithm 2, we traverse the sub-tree t_{cur} in level-order from the bottom up. For the topic t_j, which has no stream passing through the current node and no table entry exists here, we add a virtual table entry e_j with t_j as the topic field and the length of the non-zero encoding prefix of t_j as the matching length (line 2-10). Most importantly, the set of forwarding ports of e_j will be configured as the set of forwarding ports that appear most frequently in the forwarding table entries of the sub-topics of t_j. Next, we will proceed with different actions based on the following four scenarios:

In Fig. 8 to Fig. 11, the numbers in the circles represent the forwarding ports of the table entries, while the arrows reflect the parent-child relationships among the topic fields of the entries.

(1) Most entries in E_c has the same forwarding ports with e_b, and the forwarding ports of e_a are identical to e_b (line 13-16, line 19):

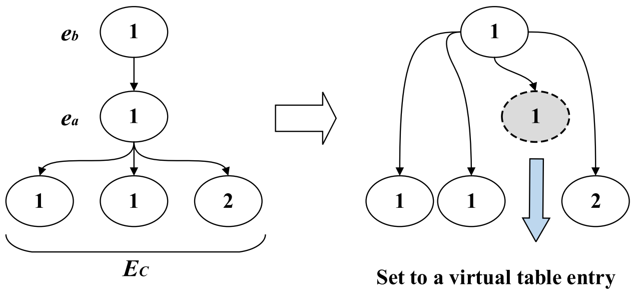

FIGURE 8. An example of case (1).

Here, e_a can be aggregated to e_b, and the entries in E_c may also be aggregated to e_b. Therefore, as shown in Fig. 8, we aggregate e_a to e_b and set e_a as a virtual table entry (for subsequent deletion) to make streams of topic t_a forwarded via e_b. We also temporarily set the parent topic of topics in T_c as t_b to allow more table entries to be aggregated to e_b in the follow-up.

(2) Most entries in E_c has the same forwarding ports with e_b, but the forwarding ports of e_a are different from e_b (line 16-18, line 19):

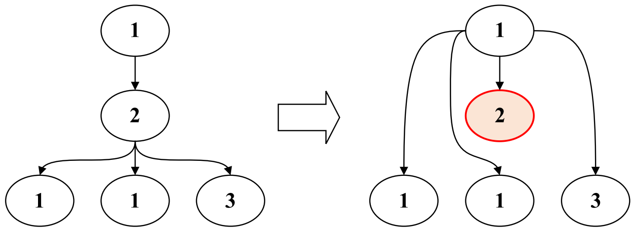

FIGURE 9. An example of case (2).

Here, the table entries in E_c are more likely to be aggregated to e_b. As shown in Fig. 9, we set the match length of the topic field in e_a to 128 so that the entry e_a can only be matched by the streams of topic t_a. Even if the table entries in E_c are aggregated to e_b in the follow-up, streams of topics in T_c will not be forwarded through e_a.

(3) Most entries in E_c has the same forwarding ports (denoted as P), but both the forwarding ports of e_a or e_b are different from P (line 20-28):

As shown in Fig. 10, we set the match length of the topic field in e_a to 128 so that the entry e_a can only be matched by the streams of topic t_a. Moreover, we added a virtual table entry with P as the set of forwarding ports to enable better aggregation of table entries in E_c to e_{new}. e_{new} takes the encoding of t_a as the matching field and the non-zero prefix length of it as the matching length. Thus, after the table entries

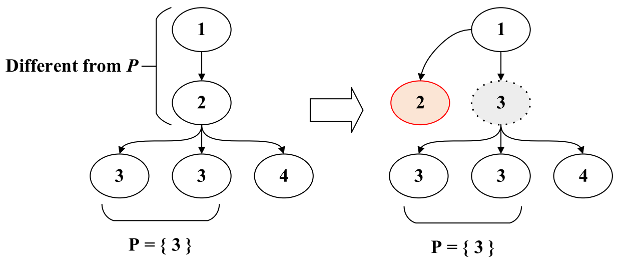

FIGURE 10. An example of case (3).

in E_c are aggregated to e_{new}, e_{new} can prefix match the streams of topics in T_c.

(4) Most entries in E_c has different forwarding ports from each other (line 29-36):

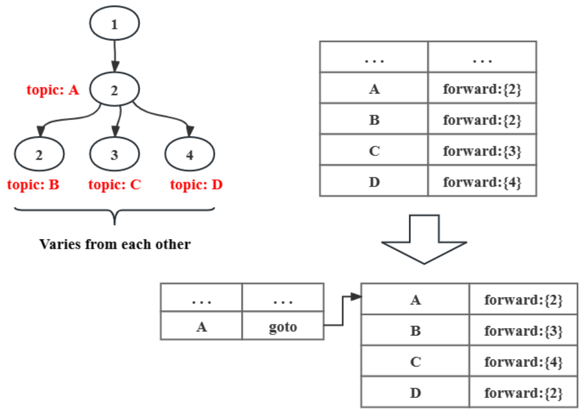

FIGURE 11. An example of case (4).

As shown in Fig. 11, for this case where it is hard to implement table entry aggregation, we try to reduce the number of matches by multi-level flow tables. However, this does not necessarily mean that the matching time can be optimized, so we also need Algorithm 3 to implement the optimization of the multi-level table structure.

B. TABLE STRUCTURE OPTIMIZATION ALGORITHM

Notably, since we have considered the effect of jumping between tables on the efficiency of the multi-level architecture in our study, the multi-level flow table structure generated by the MTGSA Algorithm may not reduce the packet matching time. In this algorithm, a sub-table will not be divided out as one of the next-level flow tables when the number of reduced table entries matches does not offset the time consumption caused by the jump between tables. Here, we denote the time to make a table jump as J and the average time to do a table entry matching as M. Then, we denote the variable K as J/M. Besides, to know whether the number of matches decreases, we need to identify the distribution of the packet amounts of different streams on the switch. However, traffic in real networks tends to change all the time. So, we assume that the average number of a stream's messages published in a unit moment is proportional to the bandwidth resources requested by the publishing source, and we denote the ratio as L. For any two aggregated forwarding tables T_1 and T_2, streams that fail to match at T_1 will continue to be matched at T_2. It is assumed that the sum of resources requested by the matched streams in the two tables is R_1 and R_2, respectively, the reduced number of matches can be expressed as follows:

$$(R_1 \times L - R_2 \times L) \times |T_2| - R_2 \times L \times K$$

From the expression, we can see that L does not affect the positivity or negativity of the results, the resource allocation is known to the system, and K is related to the specific device. Based on the above expression, we implement the Table Structure Optimization Algorithm as follows to achieve optimization of the multi-level flow table structure and ensure that the aggregate matching time is reduced.

As shown in Algorithm 3, for all flow tables that have a parent table in $\boldsymbol{T}$, if the reduced matching counts is negative due to the presence of the jump delay after dividing the current flow table $\boldsymbol{T_{cur}}$ from the parent flow table $\boldsymbol{T_p}$, $\boldsymbol{T_{cur}}$ will be merged into $\boldsymbol{T_p}$ (line 8-12).

In Fig. 12, we merge a flow table 'Table Y' into its parent table 'Table 1' if the expected matching time doesn't decrease. And the corresponding 'goto' table entry in 'Table 1' will be deleted and all flow table entries in 'Table Y' will be moved into 'Table 1'.

VI. EVALUATION

Before experimenting, we need to confirm the value of K on the tested device. Therefore, we measured the time to jump to different depths and the matching time for different numbers of table entries on a switch equipped with the Tofino programming chip. Our experiments are conducted using P4 programmable switches, and the specific parameters of the switches used in the tests are detailed in Table 2. The results are shown in Figs. 13 and 14.

TABLE 2. Configuration parameters of P4 switches.

Parameter	Configuration
Manufacturer	POLARISDN
Model Number	Flnet S9180-32X
Switching Chip	3.2Tbps Tofino BFN-T10-032D
Forwarding Rate	2500Mpps
Processor	Intel Xeon D-1527 CPU
Memory	8GB DDR4 ECC
Port	32 X 100/50/40GbE QSFP28

We can see that on the tested hardware, the time it takes to perform a flow table jump is approximately equal to the time it takes on 100 table entry matches. Therefore, we set

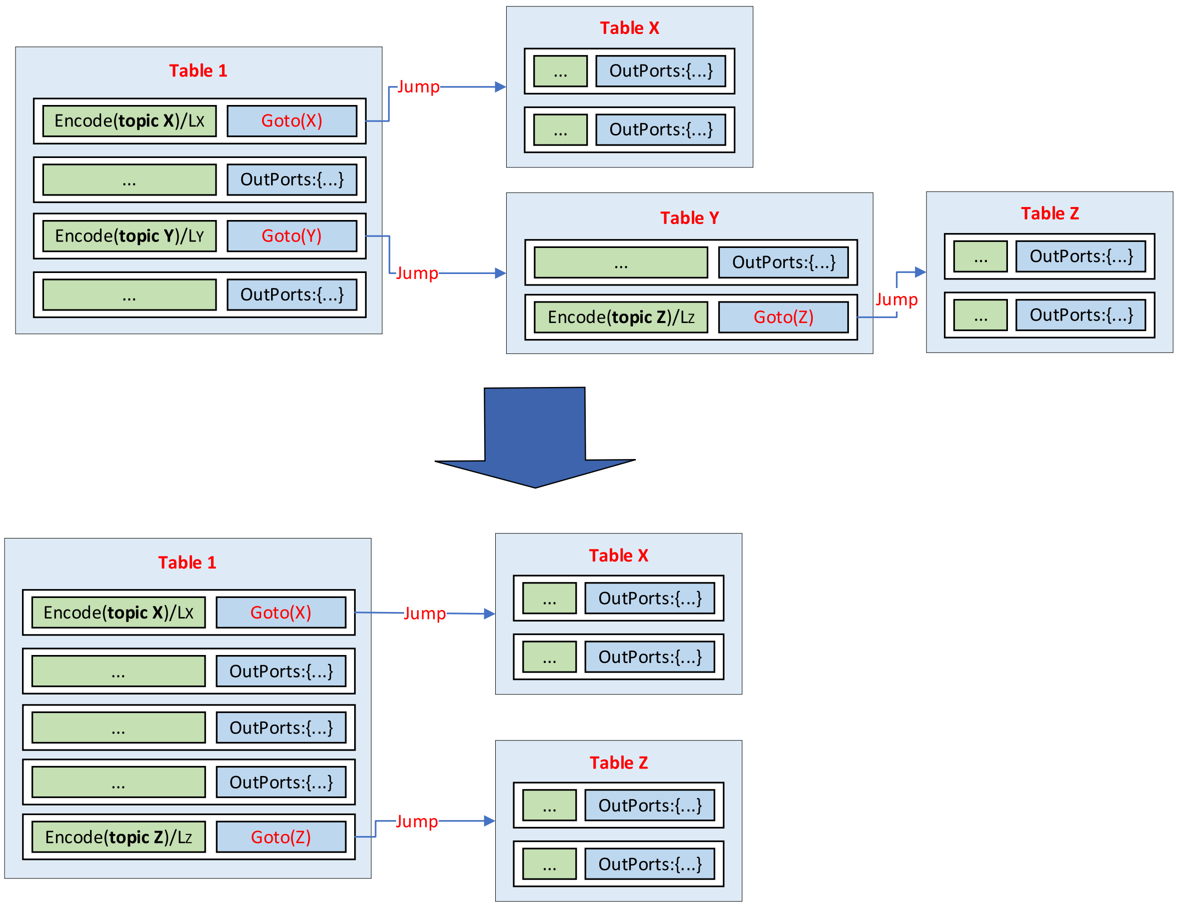

FIGURE 12. An example of flow table structure optimization.

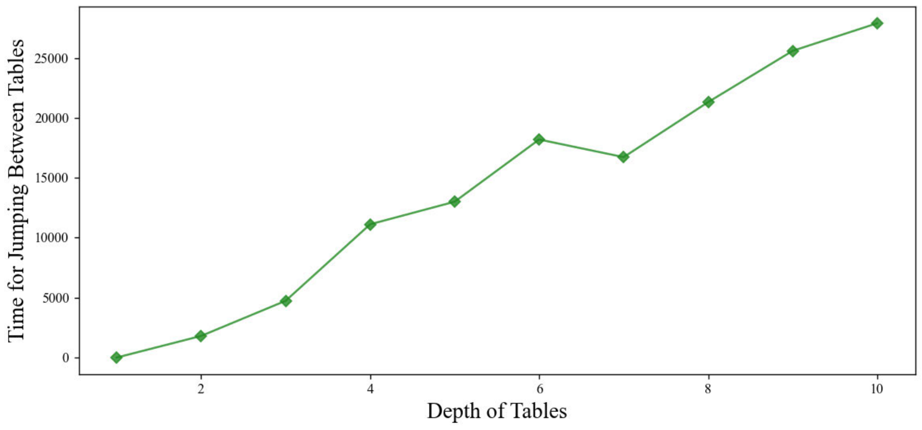

FIGURE 13. Time (ms) for jumping to the tables at different depth.

the value of K in Algorithm 3 to 100. Next, we initialized a series of random data for the test.

We conducted several experiments to evaluate the performance of this study's algorithm in terms of the encoding time, the matching counts, and the number of flow entries. The parameters used in the experiments are summarized in Table 3, and the experiments conducted are recorded in Table 4.

As shown in Fig. 15, the parameters of the topologies and topics in the test such as network connectivity, distribution of bandwidth resources, cost of links, and the publishing and subscription for topics were randomly generated by the

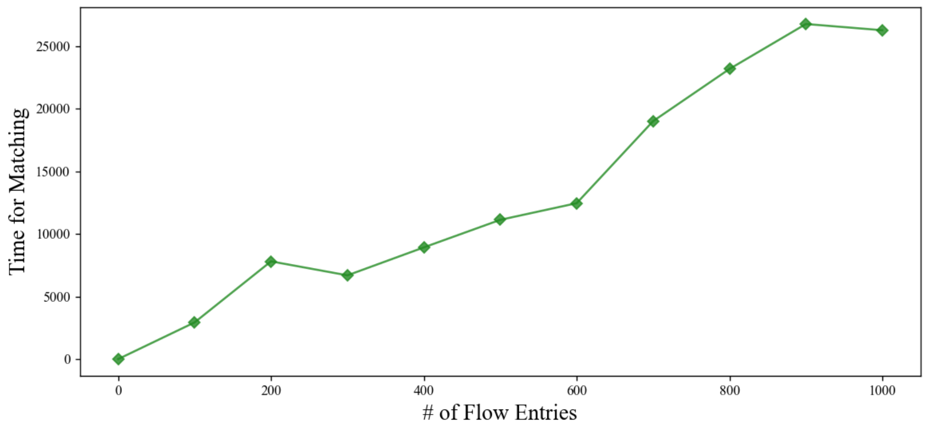

FIGURE 14. The matching time (ms) for tables with different numbers of entries.

TABLE 3. Parameter definitions.

Symbol	Definition
t	The number of pub/sub topics transmitted in the network.
n	The network size.
h	The height of the topic tree.
s	The number of sub-topics per non-leaf topic in the topic tree.

TABLE 4. Parameters used in experiments.

Exp.		T	N	H	S
	A1	40	100	4	3,4,5,6,7,8
A	A2	30	100	4	3,4,5,6,7,8
	A3	20	100	4	3,4,5,6,7,8
	B1	50,100,150,200, 250,300	10	5	4
B	B2	50,100,150,200, 250,300	10	6	4
	B3	50,100,150,200, 250,300	10	7	4
	C1	100	10,20,30,40,50,60	6	4
C	C2	150	10,20,30,40,50,60	6	4
	C3	200	10,20,30,40,50,60	6	4

“random” function in Java. For the topic tree, we generated it based on the values of h and s. For the topology, we firstly constructed a directed complete graph with n nodes; we then assign random values between 0 and 100 to the bandwidth capacity (unit: G) and cost of the links; finally, for anyone link, we randomly chose whether to delete it or not. Lastly, we randomly selected t topics to serve as the published and subscribed topics, and designated two publishers and two subscribers for each topic.

The encoding algorithm in this paper focuses on optimizing the forwarding delay and table space occupation of the P4 switches in the data plane by redesigning the flow table architecture and performing semantic aggregations on the table entries in the control plane. However, the forwarding latency of packets in a multi-level flow table structure is related to the eventual position where the packets hit a table entry. Therefore, we assume that the relationships between the amounts of packets in various topics meet the assumptions in Algorithm 3. We use the average number of matches per packet in the pipeline (where one flow table jump will be equated to K table entry matches) to reflect this metric. So, in the experiments of this study, we mainly measured the average number of matching counts and the total number of table entries before and after aggregation. We evaluated the three parameters, t, n, and s, which may affect the results of the experiments, through three groups of experiments, A, B, and C, respectively. For each experimental group, we conducted three control experiments to ensure the reliability of the observed trends in the metrics. Besides, based on the content of the algorithms, we know that s is a factor that affects the encoding time of a certain number of table entries. Thus, to confirm the efficiency of the encoding algorithm, we measured the execution time of the algorithm in Experiment A.

The algorithm was implemented in Java with Open JDK version 1.8.0_282. We used IntelliJ IDEA 2021.1.3 (Ultimate Edition) to develop, run and test the algorithm. The computer used in the experiments is an Intel(R) Core (TM) i7-1165G7 (2.80 GHz) machine with 16 GB of RAM. Additionally, we used the P4$_{16}$ language to define the data plane of the switches. The results of the experiments are graphically visualized in Fig. 16–Fig. 21.

In Fig. 16 to Fig. 18, we can observe the variation in the matching counts and the total number of table entries before

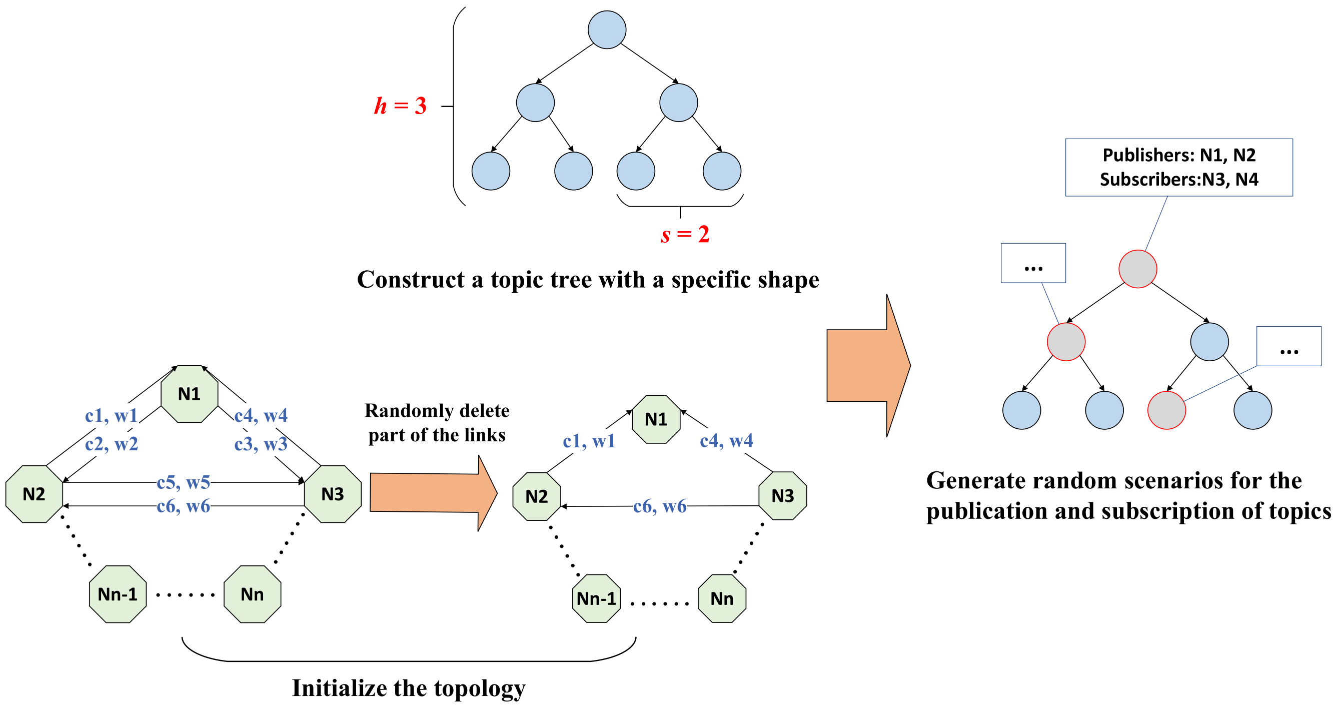
h = 3
s = 2
Construct a topic tree with a specific shape
Publishers: N1, N2
Subscribers:N3, N4
...
...
N1
N2
N3
Nn-1
Nn
c1, w1
c2, w2
c4, w4
c3, w3
c5, w5
c6, w6
Randomly delete part of the links
Initialize the topology
Generate random scenarios for the publication and subscription of topics

FIGURE 15. An example of generating an experimental topology.

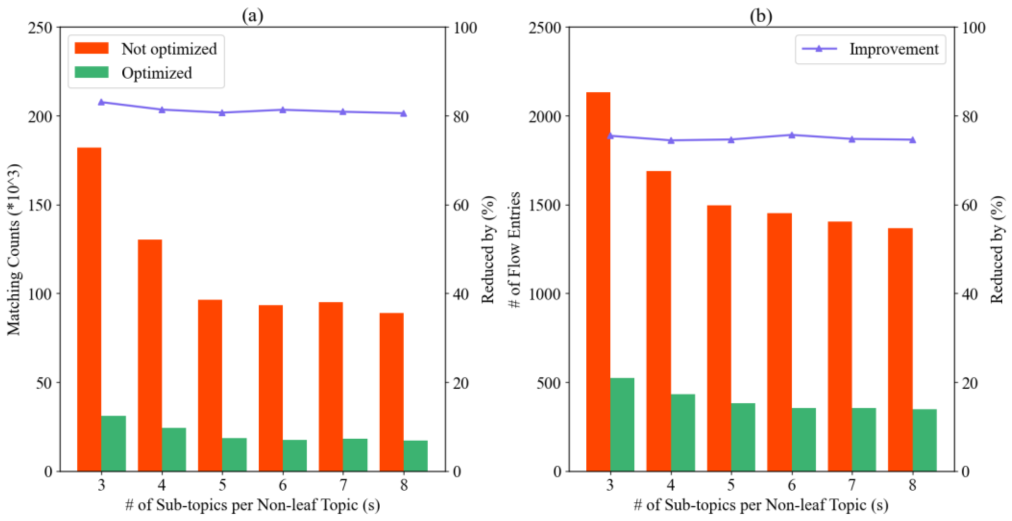

FIGURE 16. Exp. A1 results: (a) Matching Counts. (b) The Number of Flow Entries.

and after adopting our encoding algorithm for experiments A1, B1, and C1.

In Experiment A1, the two metrics show nearly similar variation before and after optimization, so we guess that the parameter s does not have much influence on the optimization ability of the encoding algorithm.

In Experiment B1, the total number of table entries grows proportionally with the parameter t, regardless of whether the encoding algorithm is used. And the number of matches shows a parabola-like relationship with t. It is clear the value of t has a significant influence on the forwarding time and flow table space occupation before and after optimization.

In Experiment C1, we find that in the small-scale network of Experiment C1, the average number of matches after optimization is unaffected by changes in network size and remains stable regardless of the variations before optimization. Similarly, the total number of table entries before optimization changes significantly as the network size

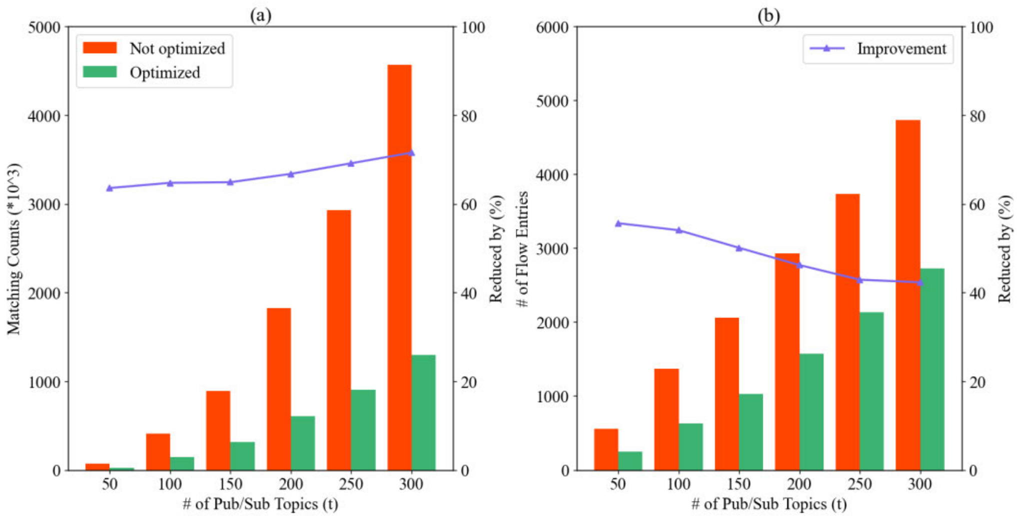

FIGURE 17. Exp. B1 results: (a) Matching Counts. (b) The Number of Flow Entries.

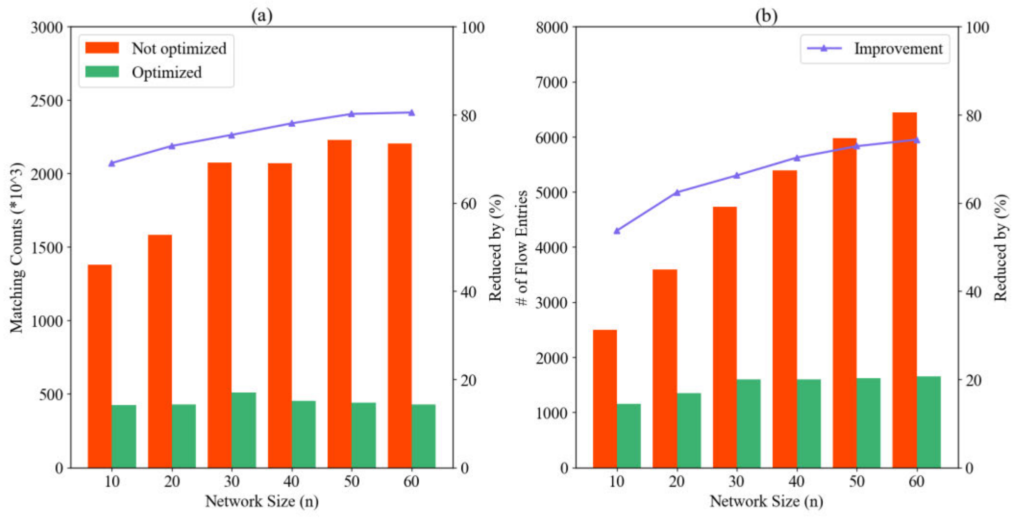

FIGURE 18. Exp. C1 results: (a) Matching Counts. (b) The Number of Flow Entries.

increases, but remains relatively stable after optimization. Therefore, we speculate that our algorithm is suitable for some networks with limited storage resources when the network size is not large.

We have processed the test data of the three experimental groups and visualized them in Fig. 19 to Fig. 21. In Fig. 19 (a), the encoding time for a certain number of table entries is proportional to the number of sub-topics per non-leaf topic, but the encoding of tens of thousands of table entries can be done within seconds. So, our algorithm has good time efficiency.

Next, we analyze and summarize the optimization rates of the two metrics in Experiments A, B and C. The optimization rates of the number of matches and table entries remain relatively stable no matter how the parameter s varies. Here, for a network of 100 nodes size, the matching counts can be reduced by about 80%, and the number of table entries can be reduced by about 70%.

As shown in Fig. 20, the two optimization rates show a slow increase and decrease with the value of t, respectively. Although it is hard to maintain a high optimization rate for both parameters when t increases, the optimization rate for the number of table entries will not keep decreasing since the total number of topics in the topic tree is the max value of t. Meanwhile, we can observe from the results of Experiment C that when the network size is small, the optimization of both matching efficiency and flow table space occupation becomes difficult. However, even in a small-scale network with only ten nodes, the decrease in the number of matches and table entries is substantial.

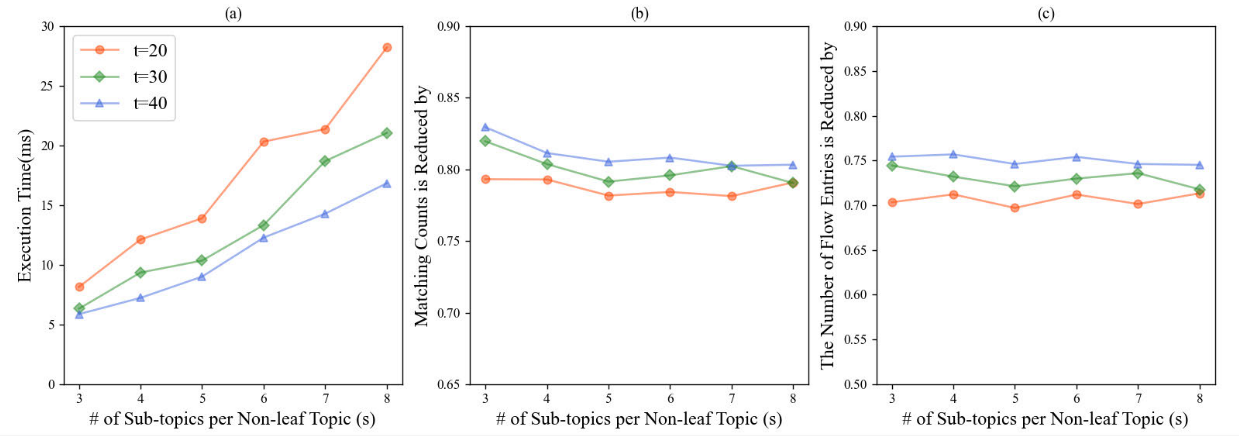

FIGURE 19. Exp. A results: (a) Encoding time (ms/100 entries). (b) Matching Counts Reduced. (c) The Flow Entries Reduced.

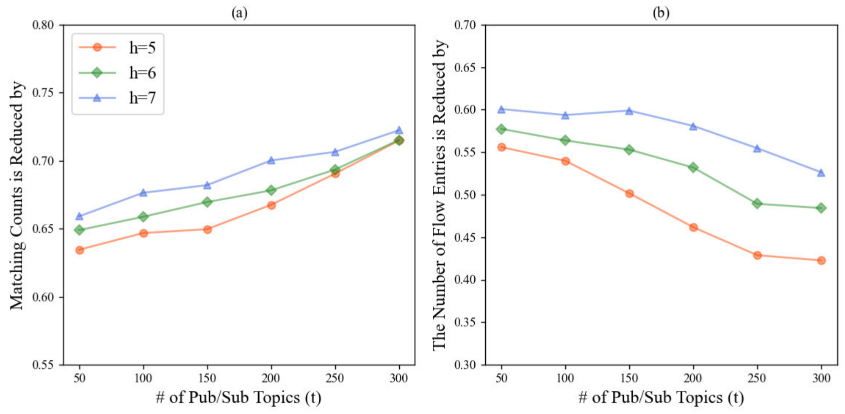

FIGURE 20. Exp. B results: (a) Matching Counts Reduced. (b) The Flow Entries Reduced.

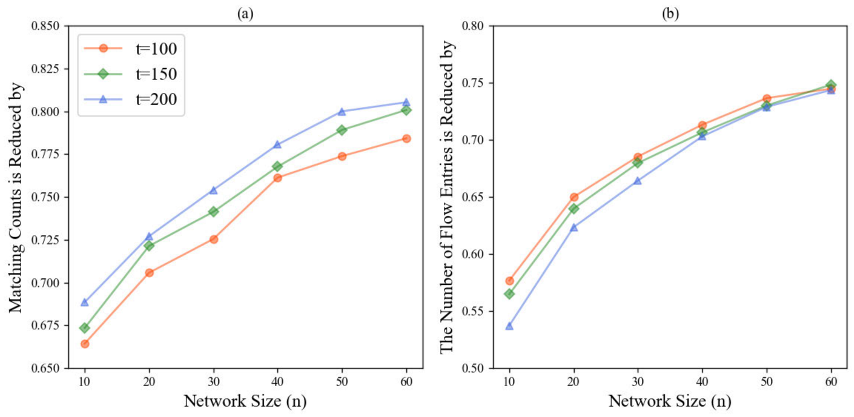

FIGURE 21. Exp. C results: (a) Matching Counts Reduced. (b) The Flow Entries Reduced.

In Fig. 21, we can see that the optimization rate increases with the number of nodes but gradually steadies. The possible reason is that when the network size enlarges, it allows different streams to share forwarding paths as well as to maintain similar forwarding behavior on most nodes. Thus, our algorithm may work better for large-scale networks and situations where the forwarding paths between topics are more centralized.

Based on the results of the above tests, we draw the following conclusions:

A. EFFECTIVENESS

Using the P4 language and P4 switches to implement a flexible multi-level flow tables architecture, the average number of table entry matches for the packets on a switch can be effectively reduced, thereby effectively reducing the packet processing time on the switches. Based on the semantic aggregation of table entries, we reduce the number of matching fields for some of the flow rules and the total number of table entries, achieving less space occupation.

B. INFLUENCE FACTOR

The performances of the proposed mechanisms and algorithms are impacted by the number of topics involved in publishing and subscribing, and the network scale. As more and more topics in the topic tree are involved in publishing and subscribing, the construction of multi-level flow tables and the optimization for the table matches become easier. However, the increase in the number of topics will complicate the publish-subscribe situation, and fewer virtual table entries can be added flexibly, which is not conducive to the semantic aggregation of the table entries and makes the optimization difficult. When the network size gets larger, the encoding algorithm proposed in this paper is more effective. Therefore, it is suitable for large-scale networks.

C. AREAS FOR IMPROVEMENT

First, we have only verified the feasibility of the adopted multi-level flow table structure on P4 languages and P4 devices without considering other protocols such as OpenFlow. Second, the parameters required for the optimization of the multi-level flow table structure in Algorithm 3 are related to specific P4 devices, while the performance of different devices in the network may be different from each other, which has not been considered in our algorithm. In the next step of our research, we will address these issues.

VII. CONCLUSION

This paper integrates the multi-level flow table architecture with topic-based pub/sub systems via P4. It presents the idea of semantic aggregation to reduce the number of table entries and achieve a high space optimization rate. First, this paper forwards the concept of semantic aggregation and then elaborates on its ideas and procedures. Next, it proposes the multi-level flow table architecture which is suitable for this research scenario, and introduces its components. Then, it elaborates on the implementation process of the algorithm. Finally, this paper implements several experiments to evaluate the proposed algorithm and analyze the performance of it. The results have proved that this algorithm has good time efficiency and achieves a high optimization rate on the number of table entries and matching time.

In the future, we will consider how the controller program can enable the automatic acquisition of flow table matching and jumping performance in the pipeline of P4 switch devices, followed by generating and optimizing the flow table structure. This way, users will not need to manually measure these parameters for each type of switch product, thus also addressing the issue of performance discrepancies among switches in the network. Additionally, we will explore implementation approaches based on other SDN technologies and more application scenarios of this research.

REFERENCES

[1] S. Xuan, H. Yu, Y. Liu, and W. Yang, "An algorithm of managing the TCP stream based on two-level hash tables," in *Proc. Int. Conf. Netw. Inf. Syst. Comput.*, Wuhan, China, Jan. 2015, pp. 90–96, doi: 10.1109/ICNISC.2015.115.

[2] L. Chen, Y. Zhang, and Y. Tang, "On the optimization of flow tables of SDN-enabled switches," in *Proc. 5th Int. Conf. Commun. Inf. Syst. (ICCIS)*, Chongqing, China, Oct. 2021, pp. 18–22, doi: 10.1109/ICCIS53528.2021.9646059.

[3] Z. Ming, W. Ling, L. Zhongqiu, L. Yuehua, Q. Ning, and L. Ran, "The design of SDN technology application in power communication access network," in *Proc. IEEE Int. Conf. Commun. Problem-Solving (ICCP)*, Guilin, China, Oct. 2015, pp. 578–581, doi: 10.1109/ICCPS.2015.7454235.

[4] C. He and X. Feng, "POMP: Protocol oblivious SDN programming with automatic multi-table pipelining," in *Proc. IEEE Conf. Comput. Commun.*, Honolulu, HI, USA, Apr. 2018, pp. 998–1006, doi: 10.1109/INFOCOM.2018.8485848.

[5] E. Kutay and A. Yener, "Semantic communications: A paradigm whose time has come," in *Proc. IEEE 8th Int. Conf. Collaboration Internet Comput. (CIC)*, Atlanta, GA, USA, Dec. 2022, pp. 68–71, doi: 10.1109/CIC56439.2022.00020.

[6] W. Yang, H. Du, Z. Q. Liew, W. Y. B. Lim, Z. Xiong, D. Niyato, X. Chi, X. Shen, and C. Miao, "Semantic communications for future Internet: Fundamentals, applications, and challenges," *IEEE Commun. Surveys Tuts.*, vol. 25, no. 1, pp. 213–250, 1st Quart., 2023, doi: 10.1109/COMST.2022.3223224.

[7] X. Luo, H.-H. Chen, and Q. Guo, "Semantic communications: Overview, open issues, and future research directions," *IEEE Wireless Commun.*, vol. 29, no. 1, pp. 210–219, Feb. 2022, doi: 10.1109/MWC.101.2100269.

[8] L. Roffia, F. Morandi, J. Kiljander, A. D'Elia, F. Vergari, F. Viola, L. Bononi, and T. Salmon Cinotti, "A semantic publish-subscribe architecture for the Internet of Things," *IEEE Internet Things J.*, vol. 3, no. 6, pp. 1274–1296, Dec. 2016, doi: 10.1109/JIOT.2016.2587380.

[9] C. Cañas and E. Pacheco, "Graph-based publish/subscribe research with dynamic subscriptions," in *Proc. Doctoral Symp. 16th Int. Middleware Conf.*, New York, NY, USA, Dec. 2015, pp. 1–4, doi: 10.1145/2843966.2843973.

[10] N. Pavlopoulou and E. Curry, "PoSSUM: An entity-centric publish/subscribe system for diverse summarization in Internet of Things," *ACM Trans. Internet Technol.*, vol. 22, no. 3, pp. 1–30, Aug. 2022, doi: 10.1145/3507911.

[11] J. Lee, S. M. Hwang, T. Abdelzaher, K. Marcus, and K. Chan, "Pub/Sub-Sum: A content summarization Pub/Sub protocol for information-centric networks," in *Proc. IEEE Mil. Commun. Conf. (MILCOM)*, Norfolk, VA, USA, Nov. 2019, pp. 847–852, doi: 10.1109/MILCOM47813.2019.9020777.

[12] P. Zehnder, P. Wiener, and D. Riemer, "Using virtual events for edge-based data stream reduction in distributed publish/subscribe systems," in *Proc. IEEE 3rd Int. Conf. Fog Edge Comput. (ICFEC)*, Larnaca, Cyprus, May 2019, pp. 1–10, doi: 10.1109/CFEC.2019.8733146.

[13] M. Hungyo and M. Pandey, "SDN based implementation of publish/subscribe paradigm using OpenFlow multicast," in *Proc. IEEE Int. Conf. Adv. Netw. Telecommun. Syst. (ANTS)*, Bangalore, India, Nov. 2016, pp. 1–6, doi: 10.1109/ANTS.2016.7947820.

[14] Y. Wang, Y. Zhang, and J. Chen, "Pursuing differentiated services in a SDN-based IoT-oriented Pub/Sub system," in *Proc. IEEE Int. Conf. Web Services (ICWS)*, Honolulu, HI, USA, Jun. 2017, pp. 906–909, doi: 10.1109/ICWS.2017.118.

[15] L. Mendiboure, M. A. Chalouf, and F. Krief, "A SDN-based Pub/Sub middleware for geographic content dissemination in Internet of Vehicles," in *Proc. IEEE 90th Veh. Technol. Conf. (VTC-Fall)*, Honolulu, HI, USA, Sep. 2019, pp. 1–6, doi: 10.1109/VTCFALL.2019.8891151.

[16] T. Sylla, R. Singh, L. Mendiboure, M. Stübert Berger, M. Berbineau, and L. Dittmann, "SoD-MQTT: A SDN-based real-time distributed MQTT broker," in *Proc. 19th Int. Conf. Wireless Mobile Comput., Netw. Commun. (WiMob)*, Montreal, QC, Canada, Jun. 2023, pp. 92–97, doi: 10.1109/WiMob58348.2023.10187779.

[17] L. Mingche and G. Lei, "Two-level tries: A general acceleration structure for parallel routing table accesses," *J. Commun. Netw.*, vol. 13, no. 4, pp. 408–417, Aug. 2011, doi: 10.1109/JCN.2011.6157461.

[18] L. Huang, Q. Shen, W. Shao, and C. Xiaoyu, "Optimizing segment routing with the maximum SLD constraint using openflow," *IEEE Access*, vol. 6, pp. 30874–30891, 2018, doi: 10.1109/ACCESS.2018.2826925.

[19] R. Biswas and J. Wu, "Traffic engineering to minimize the number of rules in SDN datacenters," *IEEE Trans. Netw. Sci. Eng.*, vol. 8, no. 2, pp. 1467–1477, Apr. 2021, doi: 10.1109/TNSE.2021.3060372.

[20] Z. Li and Y. Hu, "PASR: An efficient flow forwarding scheme based on segment routing in software-defined networking," *IEEE Access*, vol. 8, pp. 10907–10914, 2020, doi: 10.1109/ACCESS.2020.2964800.

[21] L. Hu, Y. Li, H. Zhang, L. Yuan, F. Zhou, and Q. Wu, "Robust semantic communication driven by knowledge graph," in *Proc. 9th Int. Conf. Internet Things, Syst., Manage. Secur. (IOTSMS)*, Milan, Italy, Nov. 2022, pp. 1–5, doi: 10.1109/IOTSMS58070.2022.10061867.

[22] F. Zhou, Y. Li, X. Zhang, Q. Wu, X. Lei, and R. Q. Hu, "Cognitive semantic communication systems driven by knowledge graph," in *Proc. IEEE Int. Conf. Commun.*, Seoul, South Korea, May 2022, pp. 4860–4865, doi: 10.1109/ICC45855.2022.9838470.

[23] X. Peng, Z. Qin, D. Huang, X. Tao, J. Lu, G. Liu, and C. Pan, "A robust deep learning enabled semantic communication system for text," in *Proc. IEEE Global Commun. Conf.*, Rio de Janeiro, Brazil, Dec. 2022, pp. 2704–2709, doi: 10.1109/GLOBECOM48099.2022.10000901.

[24] D. Huang, X. Tao, F. Gao, and J. Lu, "Deep learning-based image semantic coding for semantic communications," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Madrid, Spain, Dec. 2021, pp. 1–6, doi: 10.1109/GLOBECOM46510.2021.9685667.

[25] A. Li, X. Liu, G. Wang, and P. Zhang, "Domain knowledge driven semantic communication for image transmission over wireless channels," *IEEE Wireless Commun. Lett.*, vol. 12, no. 1, pp. 55–59, Jan. 2023, doi: 10.1109/LWC.2022.3216994.

[26] Z. Weng, Z. Qin, and G. Y. Li, "Semantic communications for speech signals," in *Proc. IEEE Int. Conf. Commun.*, Montreal, QC, Canada, Jun. 2021, pp. 1–6, doi: 10.1109/ICC42927.2021.9500590.

[27] P. Jiang, C.-K. Wen, S. Jin, and G. Y. Li, "Wireless semantic communications for video conferencing," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 1, pp. 230–244, Jan. 2023, doi: 10.1109/JSAC.2022.3221968.

YU ZHOU received the bachelor's degree in computer science and technology from Beijing University of Posts and Telecommunications, Beijing, China, in 2021, where she is currently pursuing the master's degree in computer science and technology. Her research interests include the Internet of Things and software-defined networking.

YANG ZHANG received the Ph.D. degree in computer applied technology from the Institute of Software, Chinese Academy of Sciences, in 2007. He is currently with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing, China. He leads a team making scientific research on the theoretic foundation of EDSOA for IoT services (National Natural Science Foundation of China). His research interests include service-oriented computing, the Internet of Things, and service security and privacy.