PAPER

Multicast Routing Model to Minimize Number of Flow Entries in Software-Defined Network

Seiki KOTACHI[†a)], ***Student Member***, **Takehiro SATO**[†], ***Member***, **Ryoichi SHINKUMA**[†], ***and*** **Eiji OKI**[†], ***Fellows***

SUMMARY The Software-defined network (SDN) uses a centralized SDN controller to store flow entries in the flow table of each SDN switch; the entries in the switch control packet flows. When a multicast service is provided in an SDN, the SDN controller stores a multicast entry dedicated for a multicast group in each SDN switch. Due to the limited capacity of each flow table, the number of flow entries required to set up a multicast tree must be suppressed. A conventional multicast routing scheme suppresses the number of multicast entries in one multicast tree by replacing some of them with unicast entries. However, since the conventional scheme individually determines a multicast tree for each request, unicast entries dedicated to the same receiver are distributed to various SDN switches if there are multiple multicast service requests. Therefore, further reduction in the number of flow entries is still possible. In this paper, we propose a multicast routing model for multiple multicast requests that minimizes the number of flow entries. This model determines multiple multicast trees simultaneously so that a unicast entry dedicated to the same receiver and stored in the same SDN switch is shared by multicast trees. We formulate the proposed model as an integer linear programming (ILP) problem. In addition, we develop a heuristic algorithm which can be used when the ILP problem cannot be solved in practical time. Numerical results show that the proposed model reduces the required number of flow entries compared to two benchmark models; the maximum reduction ratio is 49.3% when the number of multicast requests is 40.

key words: *SDN, multicast, flow entry, integer linear programming*

1. Introduction

Multicast communication supports applications that send the same information to multiple receivers, such as video conferencing and collaborative work applications, and reduces the traffic load in the network. In Internet protocol (IP) multicast [1]–[3], multicast groups are managed and identified by IP addresses. Receivers join and become members of multicast groups by using the Internet group management protocol (IGMP) [4], [5]. Since each router autonomously determines the route on which multicast packets are transmitted, the sender of a multicast service cannot control the transmission route of the packets. Furthermore, the sender cannot manage the members of the multicast group.

Software-defined network (SDN) has attracted attention due to its superior packet flow control functionality [6], [7]. The sender of a multicast service can take control of the multicast group by using the SDN technology. In an SDN, the SDN controller creates flow entries that determine the operation of SDN switches. Flow entries are stored in a flow table of each SDN switch. The procedure for providing multicast communication in an SDN is described below. The sender first specifies the receivers as a multicast group, and sends the receivers' IP addresses to the SDN controller. The SDN controller then assigns a new IP address to the multicast group, and determines the packet transfer route from the sender to the receivers. The route connecting the sender and each receiver forms the multicast tree of the multicast service. After determining the multicast tree, the SDN controller stores flow entries for the multicast tree in SDN switches located on the tree. The SDN controller then notifies the sender of the IP address assigned to the multicast group. The sender sends the multicast packets to the notified IP address. The SDN switches forward the packets according to the flow entries stored in the flow tables. Finally, the packets are forwarded to the receivers in the multicast group.

SDN switches have limited flow table capacity, so their efficiently use is essential [8], [9]. When a multicast service request arises, flow entries for the IP address allocated to the multicast group of the multicast service request are stored in the appropriate SDN switches. Hereafter, a flow entry for unicast is called a unicast entry, and that for multicast is called a multicast entry. Unicast entries and multicast entries are collectively called flow entries. A multicast entry is specific to a multicast group and cannot be shared with other multicast and unicast services. Although using multicast entries leads to more effective use of flow table capacity than forwarding packets to all receivers via unicast, flow table capacity can become scarce by using only multicast entries when multiple multicast services are established. If a receiver participates in multiple multicast groups, there is a chance of reducing the number of required flow entries by using unicast entries instead of multicast entries since unicast entries can be shared by multiple services. SDN switches use ternary content addressable memory (TCAM). While TCAM excels in packet high-speed processing, it is more expensive per megabit and consumes more power than random access memory (RAM) [10], [11]. This makes it difficult to increase the capacity of TCAM used in the SDN switch; the number of flow entries that can be stored in the flow table is restricted. For the above reasons, there is a need to efficiently use the limited flow table capacity when accommodating multicast services.

Humernbrum et al. [12] presented a scheme for determining the multicast tree that suppresses the number of required multicast entries and the type of flow entries stored

Manuscript received April 17, 2020.
Manuscript revised September 1, 2020.
Manuscript publicized November 13, 2020.

[†]The authors are with the Graduate School of Informatics, Kyoto University, Kyoto-shi, 606-8501 Japan.
a) E-mail: skotachi@icn.cce.i.kyoto-u.ac.jp

DOI: 10.1587/transcom.2020EBP3064

in each node for one multicast request in the SDN. In this scheme, unicast entries are stored in SDN switches, instead of multicast entries, in the section between each receiver and the nearest branch point to that receiver. This suppresses the total number of multicast entries stored in the SDN switches in the network. However, in this scheme, when multiple multicast services are requested, unicast entries dedicated to the same receiver can be distributed to various SDN switches. As a result, the scheme cannot effectively reduce the total number of flow entries.

Lin et al. [13] presented a locality-aware multicast approach (LAMA) that determines multicast trees and the type of flow entries stored with the aim of suppressing their total number. In LAMA, an SDN controller first clusters the multicast senders located in the vicinity into the same multicast cluster. For each multicast cluster, the SDN controller selects the SDN switch that has the fewest hops to all multicast senders as its rendezvous point (RP), and then constructs a shortest-path multicast tree from the RP to its receivers. The controller stores flow entries in the SDN switches on the multicast tree of each cluster. In the switches located between each sender and the corresponding RP, the controller stores flow entries dedicated to each sender of the cluster. In the switches located between the RP and receivers, the controller stores flow entries dedicated to each cluster. LAMA suppresses the number of stored flow entries since flow entries dedicated to each cluster, which includes multiple senders, are used instead of flow entries dedicated to each sender. However, if the number of senders that can be clustered is small due to a large number of hops between senders, LAMA cannot efficiently suppress the total number of flow entries since the number of clusters is large.

This paper proposes a routing model that can determine multicast trees simultaneously while minimizing the total number of flow entries for multiple multicast requests. In this model, a unicast entry for the same receiver can be shared among multiple multicast trees in an SDN switch. The proposed model minimizes the total number of flow entries regardless of the number of hops between senders. We formulate the problem of determining multicast trees while minimizing the total number of flow entries as an integer linear programming (ILP) problem. We solve the ILP problem to compare the proposed model with two benchmark models in terms of the total number of flow entries required to accommodate multiple multicast trees. The results show that the proposed model reduces the total number of flow entries more effectively than the benchmark models.

This paper is an extended version of [14]. The extensions to the work in [14] are as follows. We prove the NP-completeness of the decision version of the multicast tree routing problem in the proposed model. We develop a heuristic algorithm for the proposed model that runs in practical time. We evaluate the number of flow entries and computational time.

The rest of the paper is organized as follows. Section 2 describes the conventional scheme. Section 3 describes the proposed model. Section 4 evaluates the total number of flow entries by applying the proposed model. Finally, we summarize this paper in Sect. 5.

2. Conventional Scheme

In the research [12], a scheme for determining the multicast tree that suppresses the number of required multicast entries and the type of flow entries stored in each node for a single multicast request was presented. The scheme assumes that the multicast tree is a shortest path tree (SPT) in terms of hop number. A multicast entry is stored in a node if multicast packets destined to two or more receivers pass through the node. Figure 1 shows an example of its operation in multicast communication. In Fig. 1, the multicast group includes receivers 0 and 1. "multi" represents a multicast entry and "uni (Receiver x)" represents a unicast entry dedicated to receiver x. The red arrow indicates the multicast tree. At first, the tree is constructed by using only multicast entries. Then, multicast entries in nodes 1 and 7 are replaced by a unicast entry dedicated to receiver 0 and multicast entries in nodes 3, 4 and 8 are replaced by unicast entries dedicated to receiver 1. Rewriting the destination IP address of a multicast packet addressed to the multicast group to the IP address of each receiver in node 0 yields multicast communication to receivers 0 and 1. The multicast packet is transmitted in the same way as a unicast packet in the section between node 0 and each receiver.

In the scheme, the tree that minimizes the number of hops from the sender to each receiver is used as the multicast tree. The authors developed the branch-aware modification (BAM) algorithm to search the SDN switches at which unicast entries should be stored instead of multicast entries in a multicast tree. By applying the BAM algorithm to an SPT, the flow entries stored in each SDN switch are determined. There can be multiple SPTs for one combination of sender and receivers. The authors compared the performance in terms of the number of multicast entries by using the following algorithms. The first algorithm finds an SPT so as to decrease the number of hops between receivers on the tree.

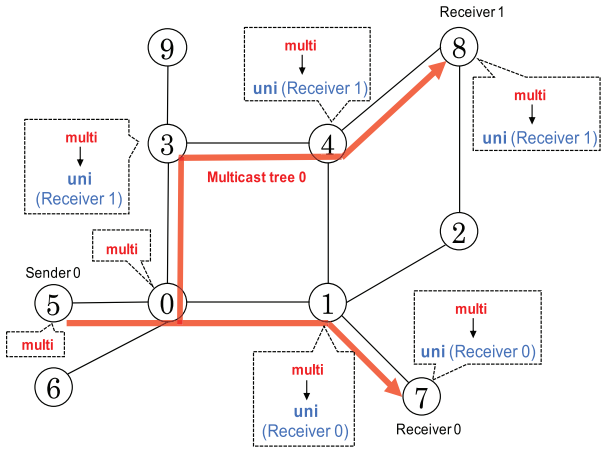

Fig. 1 Operation of multicast communication in SDN (rewriting destination IP address of packet in node 0.)

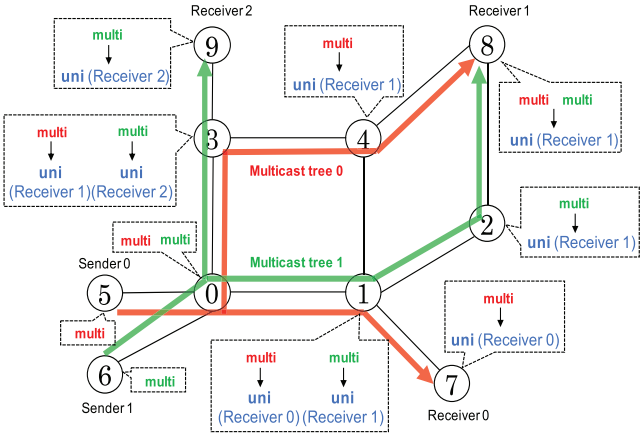

Fig. 2 Example of multicast trees yielded by EBSPT algorithm.

This algorithm is called the destination driven SPT (DDSPT) algorithm. The second algorithm finds an SPT so as to increase the number of hops between receivers on the tree. This algorithm is called the early branching SPT (EBSPT) algorithm. The DDSPT algorithm was developed in [15] and the EBSPT algorithm was developed by modifying the DDSPT algorithm in [12]. An SPT obtained by the EBSPT algorithm tends to branch at nodes closer to the sender node than that found by the DDSPT algorithm. Therefore, the SPT obtained by the EBSPT algorithm tends to have many links where a flow to a single receiver is transferred. When the BAM algorithm is applied to the above SPTs, the EBSPT algorithm can achieve fewer multicast entries than the DDSPT algorithm. In the scheme, the type of flow entries stored in each node is determined by applying the EBSPT algorithm and the BAM algorithm to a multicast request.

In general networking operation, there may be multiple multicast requests. When we apply the scheme to multiple multicast requests, we need to individually compute each multicast tree. Figure 2 shows an example of providing two multicast trees when there are two multicast requests and the EBSPT algorithm is applied to each multicast request. In Fig. 2, multicast group 0 includes sender 0 and receivers 0 and 1, and multicast group 1 includes sender 1 and receivers 1 and 2. "multi" represents a multicast entry and "uni (Receiver x)" represents a unicast entry dedicated to receiver x. The red and green arrows show multicast trees 0 and 1, respectively. At first, each of the trees is constructed by using only multicast entries. Then, in multicast tree 0, multicast entries in nodes 1 and 7 are replaced by unicast entries dedicated to receiver 0, and multicast entries in nodes 3, 4 and 8 are replaced by unicast entries dedicated to receiver 1. On the other hand, in multicast tree 1, multicast entries in nodes 1, 2 and 8 are replaced by unicast entries dedicated to receiver 1, and multicast entries in nodes 3 and 9 are replaced by unicast entries dedicated to receiver 2. In multicast tree 0, by rewriting the destination IP address of a multicast packet addressed to multicast group 0 to the IP address of each receiver in node 0, multicast communication from sender 0 to receivers 0 and 1 is realized. In multicast tree 1, by rewriting the destination IP address of a multicast packet addressed to multicast group 1 to the IP address of each receiver in node 0, multicast communication from sender 1 to receivers 1 and 2 is realized. If unicast entries dedicated to the same receiver are set in the same node by different multicast trees, only one unicast entry dedicated to the receiver is stored in the node and shared by the different multicast trees. However, in Fig. 2, unicast entries dedicated to receiver 1 are distributed to nodes 1, 2, 3, and 4. Only the unicast entry stored in node 8 is shared by multicast trees 0 and 1. If the flow from sender 1 to receiver 1 in multicast tree 1 is changed to the route that traverses nodes 5, 0, 1, 4, and 8, the unicast entry dedicated to receiver 1 stored in node 4 can be shared by multicast trees 0 and 1. This indicates that, in the example of Fig. 2, unicast entries cannot be shared effectively; there still remains the possibility of reducing the number of flow entries.

3. Proposed Model

3.1 Model Description

We describe the proposed model in this section. The proposed model determines multicast trees that minimizes the total number of flow entries when multiple multicast service requests occur in an SDN, under the constraint that different multicast trees are allowed to share a unicast entry dedicated to the same receiver stored in the same node.

We assume that a sender host and a receiver host are connected to a node with zero hop. A node connected by a sender host is called a sender node and that connected by a receiver host is called a receiver node. For simplicity, we assume that the maximum number of senders or receivers directly connected to one SDN switch in a single multicast tree is one.

Directed graph $G(V, E)$ represents the SDN as a set of nodes V and a set of links E. A set of multicast trees is denoted by R. A set of receivers in multicast tree $r \in R$ is denoted by K_r. The set of receivers in all multicast trees is denoted by K, where $K = \bigcup_{r \in R} K_r$. The sender node of multicast tree $r \in R$ is denoted by $s_r \in V$, and the node of receiver $k \in K$ is denoted by $d_k \in V$.

We explain how to set flow entries by using Fig. 3, which shows two multicast trees in a ten-node network; a red arrow for multicast tree 0 and a green arrow for multicast tree 1. Receivers 0, 1, and 2 are located at nodes 7, 8, and 9, respectively ($d_0 = 7$, $d_1 = 8$, $d_2 = 9$). In multicast tree 0, the sender node is node 5 ($s_0 = 5$). The set of receivers in multicast tree 0, K_0, consists of receivers 0 and 1. In multicast tree 1, the sender node is node 6 ($s_1 = 6$). The set of receivers in multicast tree 1, K_1, consists of receivers 1 and 2. In multicast tree 0, multicast entries dedicated to this tree are set in nodes 5, 0, 1, 4, 7, and 8. Here, multicast packets from sender s_0 are correctly transferred to receiver d_0 by the following operations.

- Rewrite the destination IP address of the packets to that of d_0 at node 1

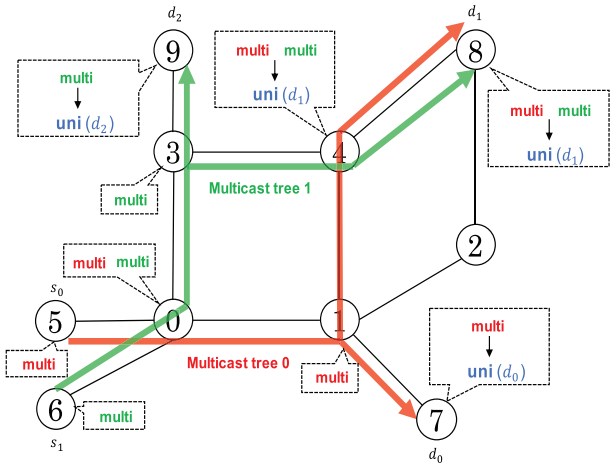

Fig. 3 Example of multicast trees yielded by proposed model.

- Replace the multicast entry stored in node 7 with a unicast entry for d_0

In the same way, the multicast entries set in nodes 4 and 8 can be replaced with unicast entries for d_1. In multicast tree 1, multicast entries dedicated to this tree are set in nodes 6, 0, 3, 4, 8, and 9. As with multicast tree 0, the multicast entries in nodes 4 and 8 can be replaced with unicast entries for d_1, and the multicast entry in node 9 can be replaced with a unicast entry for d_2. When we simultaneously accommodate these multicast trees 0 and 1 in the SDN, considering the case where all flow entries stored in the SDN switches are multicast entries, the total number of flow entries is 12. However, by replacing the flow entries stored in nodes 7, 4, 8, and 9 with unicast entries for each receiver, multicast trees 0 and 1 can share the unicast entries stored in nodes 4 and 8, and the total number of flow entries is 10. When multiple multicast trees are accommodated in the SDN as this example, the total number of flow entries can be reduced by replacing multicast entries with unicast entries and sharing the unicast entries among multiple multicast trees.

Note that we mainly focus on the number of flow entries in the proposed model since the flow table capacity can become a bottleneck to operate SDN networks, as described in Sect. 1. The proposed model can output multicast trees that branch near the source node since the proposed model stores unicast entries in SDN switches if it promotes the sharing of unicast entries among multiple multicast trees and suppresses the total number of flow entries. Therefore, the proposed model can yield packet transmission with lower efficiency than when multicast trees branch as close the destination nodes as possible. The proposed model can be expanded to consider packet transmission efficiency by adding additional parameters and constraints. Similarly, the proposed model can be expanded to include other network limitations, such as latency and throughput.

3.2 ILP Formulation

We formulate the proposed model to determine multicast trees presented in Sect. 3.1 as an ILP problem.

We define decision variables and parameters used in this formulation as follows. x_{rv} is a binary decision variable that is set to 1 if a multicast entry of multicast tree $r \in R$ is stored in node $v \in V$, and 0 otherwise. y_{kv} is a binary decision variable that is set to 1 if a unicast entry for receiver $k \in K$ is stored in node $v \in V$, and 0 otherwise. q_{rkuv} is a binary decision variable that is set to 1, in multicast tree $r \in R$, if the packet flow for receiver $k \in K_r$ passes through link $(u, v) \in E$, and 0 otherwise. w_{ruv} is a binary decision variable that is set to 1 if link $(u, v) \in E$ is included in multicast tree $r \in R$, and 0 otherwise. z_{rkv} is a binary decision variable that is set to 1 if one or more flow entries for receiver $k \in K_r$ in multicast tree $r \in R$ are stored in node $v \in V$, and 0 otherwise. e is a sufficiently small positive number.

The objective function to minimize is given as:

$$\text{Min} \sum_{r \in R} \sum_{v \in V} x_{rv} + (1 - e) \times \sum_{k \in K} \sum_{v \in V} y_{kv}. \tag{1}$$

The first term represents the total number of multicast entries stored in all nodes in all multicast trees. The second term excluding $(1 - e)$ represents the total number of unicast entries for each receiver stored in all nodes. By multiplying the second term by a number slightly smaller than one, unicast entries are stored with high priority when the total number of flow entries does not change, no matter whether the flow entry stored in the node is a multicast entry or a unicast entry.

The constraints are given by (2)–(12), as shown below.

$$\sum_{u \in V:(v,u) \in E} q_{rkvu} - \sum_{u \in V:(u,v) \in E} q_{rkuv} = \begin{cases} 1 & \text{if } v = s_r, v \neq d_k \\ -1 & \text{if } v = d_k, v \neq s_r \\ 0 & \text{otherwise,} \end{cases} \tag{2}$$

$$\forall v \in V, \forall r \in R, \forall k \in K_r$$

$$q_{rkuv} + q_{rkvu} \leq 1, \forall (u, v), (v, u) \in E, \forall r \in R, \forall k \in K_r \tag{3}$$

Equation (2) preserves the packet flow from sender to receiver. Equation (3) is a constraint that does not allow a packet flow to return to the previous hop node.

$$q_{rkuv} \leq w_{ruv}, \forall (u, v) \in E, \forall r \in R, \forall k \in K_r \tag{4}$$

$$w_{ruv} \leq \sum_{k \in K_r} q_{rkuv}, \forall (u, v) \in E, \forall r \in R \tag{5}$$

$$\sum_{u \in V:(u,v) \in E} w_{ruv} \leq 1, \forall r \in R, \forall v \in V \tag{6}$$

$$\sum_{u \in V:(u,v) \in E} q_{rkuv} = z_{rkv}, \forall v \in V \backslash \{s_r\}, \forall r \in R, \forall k \in K_r \tag{7}$$

$$\sum_{u \in V:(s_r,u) \in E} q_{rks_ru} = z_{rks_r}, \forall r \in R, \forall k \in K_r \tag{8}$$

Equations (4) and (5) represent the link utilization in each

multicast tree. Equation (6) indicates that the number of nodes from which packets are transferred to node $v \in V$ is at most one in multicast tree $r \in R$. This constraint prohibits the existence of loops in multicast tree $r \in R$. Equations (7) and (8) indicate whether there is a flow entry for receiver $k \in K_r$ of multicast tree $r \in R$ in each node.

$$2 \times x_{rs_r} \leq \sum_{k \in K_r} \sum_{u \in V:(s_r,u) \in E} q_{rks_r u}, \forall r \in R \tag{9}$$

$$2 \times x_{rv} \leq \sum_{k \in K_r} \sum_{u \in V:(u,v) \in E} q_{rkuv}, \forall v \in V \backslash \{s_r\}, \forall r \in R \tag{10}$$

Equations (9) and (10) indicate whether a multicast entry is stored in each node.

$$z_{rkv} - x_{rv} \leq y_{kv}, \forall v \in V, \forall r \in R, \forall k \in K_r \tag{11}$$

Equation (11) indicates whether a unicast entry is stored in each node.

$$z_{rku} - x_{ru} + q_{rkuv} - 1 \leq y_{kv}, \forall (u, v) \in E, \forall r \in R, \forall k \in K_r \tag{12}$$

Equation (12) indicates, when a unicast entry for receiver $k \in K_r$ is stored in a node once in multicast tree $r \in R$, unicast entries for receiver $k \in K_r$ are stored in all nodes from the node to receiver k's node. This constraint prohibits a packet once transferred by using unicast entries from being transferred again through the use of multicast entries.

3.3 NP-Completeness

We prove that the decision version of the multicast tree routing problem in the proposed model is NP-complete. We define the multicast tree routing problem to minimize the number of flow entries (MTR-M) as follows:

Definition: When network topology $G(V, E)$, a set of multicast service requests R, a source node $s_r \in V$ of each service $r \in R$, a set of receivers K, a set of receivers K_r of each service $r \in R$, and the nodes $d_k \in V$ of each receiver $k \in K$ are given, is there any routing of multicast trees for $|R|$ service requests such that the total number of flow entries is at most f?

Theorem: The MTR-M problem is NP-complete.

Proof: First, we show that the MTR-M problem belongs to NP. When an instance of MTR-M problem is given, we can calculate the total number of multicast entries in $O(|R||V|)$ and the total number of unicast entries in $O(|K||V|)$. Therefore, we can verify whether the total number of flow entries is at most f in polynomial time $O((|R| + |K|)|V|)$. Therefore, the MTR-M problem belongs to NP.

Next, we show that the minimum Steiner tree problem under the condition that all edge weights are equal, which is a known NP-complete problem [16], can be reduced to the MTR-M problem in polynomial time. The minimum Steiner tree problem as a decision problem is defined as: given undirected graph $G'(V', E')$, a set of vertices $T \subseteq V'$, and weights of edges in E', is there any tree in G' that includes all the vertices of $T \subseteq V'$ and whose sum of edge weights is at most g?

We construct an instance of the MTR-M problem from any instance of the minimum Steiner tree problem. An instance of the MTR-M problem is constructed with the following algorithm, which runs in polynomial time $O(|V'| + |E'|)$.

1. Let undirected graph $G'(V', E')$ correspond to network topology $G(V, E)$. Each undirected link in E' is converted to two directed links that connect the same pair of nodes in E at different directions.
2. Set $|R| = 1$. Let the number of receivers in $r \in R$ be $|K| = |K_r| = |T| - 1$.
3. Let one of the nodes in $T \subseteq V'$ be the sender node of multicast request r, $s_r \in V$, and the other $|T| - 1$ nodes in $T \subseteq V'$ be the receiver nodes of multicast request r, $d_k \in V$, where $k = 1, 2, \cdots, |K_r|$.
4. Let the weight of all edges be 1.
5. Let f be $g + 1$.

If the instance of the minimum Steiner tree problem is a Yes instance, there is a tree that includes all the vertices of $T \subseteq V'$ in G' and whose sum of edge weights is at most g. At this time, in the corresponding MTR-M instance, there is a route that connects the sender node of the multicast request, $s_r \in V$, and the receiver nodes of r, $d_k \in V$, where $k = 1, 2, \cdots, |K_r|$. Since we consider one multicast request, a unicast entry dedicated to receiver node $d_k \in V$ is stored in each node on the multicast tree if a flow to one receiver passes through the node, and a multicast entry dedicated to multicast request $r \in R$ is stored in each node on the multicast tree if a flow to two or more receivers passes through the node. The total number of flow entries placed on the network is at most $g + 1 = f$. This shows that there is a multicast tree assignment in which the total number of flow entries is at most f. Therefore, the corresponding MTR-M problem instance is a Yes instance.

Conversely, if the MTR-M instance is a Yes instance, there is a multicast tree routing for $|R|$ service request ($|R| = 1$) in which the total number of flow entries is at most f. Moreover there are at most f nodes that have a unicast or multicast entry in the network. Let $W \subseteq V$ denote a set of these nodes. Let $F \subseteq E$ denote a set of links in the multicast tree. Graph $G''(W, F)$ is a subgraph of G. G'' includes at most f nodes. This means that, in the corresponding instance of the minimum Steiner tree problem, there is a tree containing all the vertices of $T \subseteq V'$. Since all edge weights are 1, the sum of edge weights of the tree is at most $f - 1 = g$. Therefore, the corresponding instance of the minimum Steiner tree problem is a Yes instance.

In summary, the MTR-M problem belongs to NP and the minimum Steiner tree problem, which is a known NP-complete problem, can be reduced to the MTR-M problem in polynomial time. Therefore, the MTR-M problem is NP-complete. □

4. Heuristic Algorithm

As the scale of the problem grows, the ILP problem introduced in Sect. 3.2 can become intractable. In this section, we present a heuristic algorithm, named the "two-step algorithm," to obtain multicast trees by using the proposed model in a practical time even when the problem scale is large.

4.1 Overview

Algorithm 1 shows the two-step algorithm. This algorithm determines routes of all pairs of sender and receiver in all multicast requests before the type of flow entries stored in each node is determined. This algorithm is composed of two steps.

The first step consists of lines 1 to 13 in Algorithm 1. This step determines all muticast trees so that as many links where flows to the same receiver pass through as possible are overlapped for different multicast requests. Using these trees promotes the sharing of unicast entries dedicated to the same receiver among different multicast requests.

The second step consists of lines 14 to 34 in Algorithm 1. This step determines the type of flow entries stored in each node for realizing all multicast requests. We determine the type of flow entries stored in each node so that the total number of flow entries is as small as possible by sharing unicast entries dedicated to the same receiver between different multicast requests.

4.2 Step 1 in Two-Step Algorithm

Step 1 starts by computing a multicast tree for every receiver node by assuming that sender nodes and receiver nodes are interchanged. We call this multicast tree "an interchanged tree" hereafter. Next, we determine multicast trees for original multicast requests by computing paths on which a sender node and receiver nodes of each original multicast request are connected via "interchanged trees." The details of step 1 in the two-step algorithm are explained as follows.

s_{reverse} is a receiver node in original multicast requests, which is set to a sender node of an "interchanged tree." For receiver $k \in K$, s_{reverse} is set to d_k. R' is a set of multicast requests where k is contained in the set of receivers K_r. P_{reverse} is a set of sender nodes s_r for all multicast requests $r \in R'$. We apply the DDSPT algorithm by setting the sender node to s_{reverse} and receiver nodes to P_{reverse} to obtain an "interchanged tree" for receiver k. The DDSPT algorithm finds an SPT that tends to branch at nodes close to receivers so as to decrease the number of hops between receivers on the tree. Therefore, by applying the DDSPT algorithm, the number of links, through which multiple multicast flows destined to the same receiver pass, tends to be large, which leads to enhanced sharing of unicast entries dedicated to receiver $k \in K$. The paths connecting s_{reverse} and each node in P_{reverse} in the "interchanged tree" are used as the paths connecting receiver node d_k of the original multicast requests r and s_r. Finally, in line 8 to 13 in Algorithm 1, the multicast trees of the original multicast requests are formed by the "interchanged trees" obtained in lines 1 to 7.

4.3 Step 2 in Two-Step Algorithm

Step 2 determines the type of flow entries stored in each node. Its details are as follows.

R_v is the set of multicast requests that pass through node v. We determine the type of flow entries stored in each node, $v \in V$, to form multicast requests whose multicast trees pass through v. Nodes in V are processed in ascending order of the number of receivers located downstream of node v in multicast request $r \in R_v$. If multicast request $r \in R_v$ cannot be formed by unicast entries already stored in node v, we compare the following two numbers to determine the type of additional flow entries stored in node v. Assume that we need to store one or more additional unicast entries in v to realize multicast request $r \in R_v$. One number is the number of additional unicast entries stored in v, which is represented by $a - b$ in Algorithm 1. The other number is the number of multicast requests which can be realized by using all unicast entries in v, including the additional ones, which is represented by c in Algorithm 1. a is the number of receivers located downstream of node v in multicast request $r \in R_v$. b is the number of receivers located downstream of node v in multicast request $r \in R_v$ and whose dedicated unicast entries are already stored in node v. If $a - b < c$, the increase in the number of flow entries can be suppressed by storing additional unicast entries instead of new multicast entries. Therefore, unicast entries dedicated to receiver k, which is located downstream of node v in multicast request $r \in R_v$, are stored in all nodes located between node v and receiver node d_k for every receiver located downstream of node v in multicast request $r \in R_v$. If $a - b = c$, the increase in the number of flow entries created by the additional unicast entries is equal to the increase in the number of flow entries yielded by storing new multicast entries. In this case, unicast entries are stored in the same way as the case of $a - b < c$ to promote the sharing of the unicast entries among multiple multicast trees. If $a - b > c$, the increase in the number of flow entries can be suppressed by storing new multicast entries instead of additional unicast entries. Therefore, multicast entries dedicated to multicast request $r \in R_v$ are stored in all nodes located between sender node s_r and node v. Lines 15 to 32, in which we determine the type of flow entries stored in each node, are repeated for all nodes in descending order of the number of flows passing through the node.

5. Numerical Results

5.1 Environment

We call the proposed model "multicast routing considering all requests (MR-A)" in this section.

Algorithm 1 two-step algorithm

Input: Network $G(V, E)$, sender node $s_r \in V$, set of receivers K_r of multicast request $r \in R$, and set of receivers K of all multicast requests $r \in R$
Output: Flow entries in each node $v \in V$

for receiver $k \in K$ **do**
  $s_{\text{reverse}} = d_k$
  R' = a set of multicast requests where k is contained in a set of receivers K_r
  P_{reverse} = a set of sender nodes s_r of all multicast request $r \in R'$
  Apply the DDSPT algorithm by setting the sender node to s_{reverse} and receiver nodes to P_{reverse}
  Store paths that connect s_{reverse} and nodes in P_{reverse}
end for
for multicast request $r \in R$ **do**
  for receiver $k \in K_r$ **do**
    Determine path that connects $s_r \in V$ and d_k by using the stored paths
  end for
  Determine the multicast tree in multicast request r
end for
for $v \in V$ in descending order of the number of flows passing through v **do**
  if there are one or more flows passing through node v **then**
    R_v = a set of multicast requests which pass through node v
    for $r \in R_v$ in ascending order of the number of receivers located downstream of node v **do**
      if r cannot be constructed by unicast entries already stored in node v **then**
        a = the number of receivers located downstream of node v in multicast request r
        b = the number of receivers located downstream of node v in multicast request r and whose dedicated unicast entries are already stored in node v
        c = the number of multicast requests that can be realized by using all unicast entries stored in node v, including those need to be additionally stored in v to realize multicast request $r \in R_v$ by using only unicast entries
        if $a - b \leq c$ **then**
          for receivers k reached from node v in multicast request r **do**
            Store unicast entry dedicated to receiver k in every nodes from v to d_k
          end for
        end if
        if $a - b > c$ **then**
          Store multicast entry dedicated to multicast request r in every nodes from s_r to v
        end if
      end if
    end for
  end if
end for

We evaluate the total number of flow entries with respect to the number of multicast requests for benchmark models and MR-A. Furthermore, under the condition that the number of multicast requests is fixed, we evaluate the ratio of the number of multicast entries to that of unicast entries in each case of applying the benchmark models and MR-A.

In this evaluation, we use the National Science Foundation (NSF) network topology with 14 nodes and 21 bidirectional links shown in Fig. 4; it is often used for network simulations [17]–[19]. We use the number of flow entries, computation time, ratio of the type of flow entries, and the number of flows as the evaluation metrics. The number of receivers in each multicast request is set to a random integer from 1 to 10. We assume that there is no upper limit on the number of storable flow entries of each node nor on the bandwidth of each link. For each multicast request, we first determine the sender node randomly and then choose the receiver nodes randomly from the remaining nodes. The number of senders or receivers in one node is at most one.

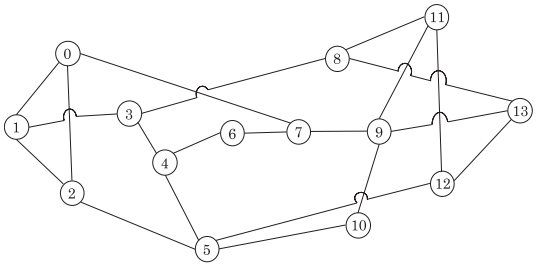

Fig. 4 NSF network.

The hardware platform uses an Intel Xeon E3-1270 v6 3.80 GHz 4-core CPU and 64 GB RAM to solve the ILP problems and the heuristic algorithms. The ILP problems are solved by using CPLEX® Interactive Optimizer 12.8.0.0 [20].

5.2 Benchmark Models

In this numerical evaluation, in addition to MR-A with ILP and MR-A with the two-step algorithm, we introduce three benchmark models that determine multicast trees one by one.

Benchmark model 1 is based on the conventional scheme in [12]. Benchmark model 1 determines each multicast tree by using the EBSPT algorithm. Flow entries stored in each node are determined based on the multicast trees under the condition that different multicast trees can share unicast entries dedicated to the same receiver. We call benchmark model 1 "multicast routing considering individual requests with the EBSPT algorithm (MR-IE)."

One may consider that the total number of flow entries can be decreased efficiently by minimizing the number of flow entries in each multicast tree. Therefore, we introduce benchmark model 2. Benchmark model 2 determines each multicast tree so as to minimize the total number of flow entries for the multicast request. In this model, we determine each multicast tree by solving the ILP problem in (1)–(12) under the condition that the number of multicast requests $|R|$ is one. We call benchmark model 2 "multicast routing considering individual requests with ILP (MR-II)."

Benchmark model 3 uses the DDSPT algorithm to determine a multicast tree. We call benchmark model 3 "multicast routing considering individual requests with the DDSPT algorithm (MR-ID)." The reason we introduce MR-ID is that the number of flows in a link can be suppressed by using the DDSPT algorithm since the DDSPT algorithm finds a multicast tree that tends to branch at nodes close to receivers. In the evaluation of the number of flows, we use MR-ID instead

of MR-IE.

5.3 Number of Flow Entries

Figure 5 shows the relationship between the number of multicast requests and the total number of flow entries for multicast trees yielded by each model in NSF network. The horizontal axis is the number of multicast requests, and the vertical axis is the number of flow entries. We take the average value of 200 simulation runs for each multicast request number. When the number of multicast requests is small, the total number of flow entries of MR-II, that of MR-IE, and that of MR-A with the two-step algorithm are almost the same, and are larger than that of MR-A with ILP. However, as the number of multicast requests increases, the number of flow entries of MR-A with the two-step algorithm and MR-A with ILP become smaller than that of MR-II and MR-IE. This is because in MR-II and MR-IE, each multicast tree is determined individually. By determining each multicast tree individually, unicast entries are distributed to multiple nodes and multiple multicast entries are stored in a node instead of a single unicast entry that can be shared between multiple multicast requests. In MR-A with the two-step algorithm and MR-A with ILP, all multicast trees are determined simultaneously. Compared with MR-II, MR-IE and MR-A with the two-step algorithm, MR-A with ILP offers a larger reduction in the total number of flow entries as the number of multicast requests increases. In the case of 40 multicast requests, the total number of flow entries in MR-A with ILP is decreased by 49.3% compared to MR-IE. Although reduction in flow entry number is not as large as MR-A with ILP, the total number of flow entries in MR-A with the two-step algorithm is decreased by 35.2% compared to MR-IE. This is because MR-A promotes the sharing of unicast entries among multicast trees. This result indicates that MR-A reduces the total number of flow entries more effectively relative to the benchmark models.

Figure 6 shows the number of multicast and unicast entries when the multicast trees are obtained by applying each model. The number of multicast requests is set to 40. We take the average value of 200 simulation runs for each model. In Fig. 6, the ratio of the number of multicast entries to that of all flow entries with MR-IE is 71.8%, that with MR-II is 81.6%, that with MR-A with the two-step algorithm is 22.7%, and that with MR-A with ILP is 35.7%. It can be seen that the ratios of the number of multicast entries to that of all flow entries with MR-IE and MR-II, in which multicast trees are determined individually, are close. Also, it can be seen that the ratios of the number of multicast entries to that of all flow entries with MR-A with the two-step algorithm and MR-A with ILP, in which all multicast trees are determined simultaneously, are close. The ratios of the number of multicast entries to that of all flow entries with MR-IE and MR-II are larger than that with MR-A with the two-step algorithm and MR-A with ILP. The total number of flow entries with MR-IE and that with MR-II are larger than that with MR-A with the two-step algorithm and that

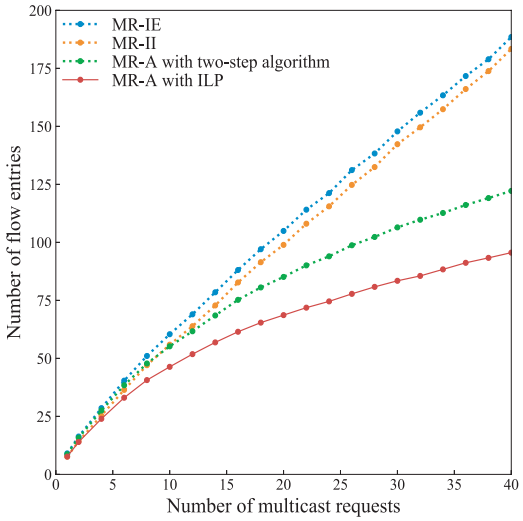

Fig. 5 Relationship between number of multicast requests and total number of flow entries in NSF network.

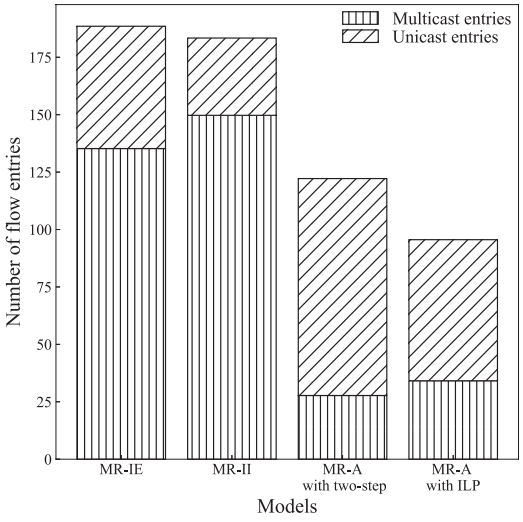

Fig. 6 Relationship between models and number of flow entries in NSF network (number of trees = 40).

with MR-A with ILP. These results mean that sharing unicast entries between different trees is effective in decreasing the total number of flow entries and is enhanced by considering multiple multicast requests simultaneously.

Figures 7 and 8 show the relationship between the number of multicast requests and the maximum number of flow entries stored in a node. Figure 7 shows the maximum number of flow entries per node averaged over 200 trials for each multicast request number. Figure 8 shows the maximum number of flow entries per node observed in 200 trials for each multicast request number. In Fig. 7, when the number of multicast requests is less than or equal to 14, all models yield almost the same average values of maximum number of flow entries stored in a node; the difference in the average values is at most 1.41. The same observation can be seen in Fig. 8; the difference in the results of all models is at most

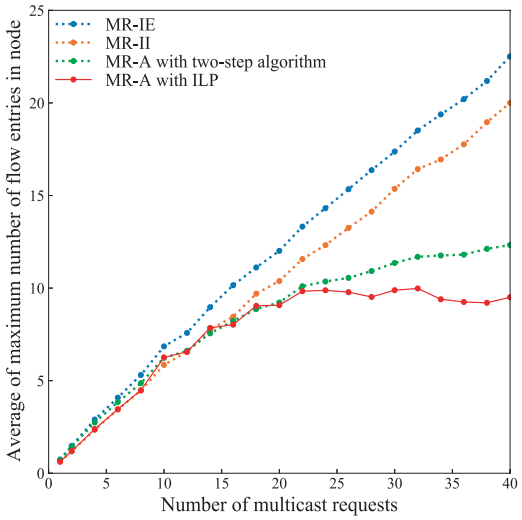

Fig. 7 Relationship between number of multicast requests and maximum number of flow entries in node when averaging results of all trials in NSF network.

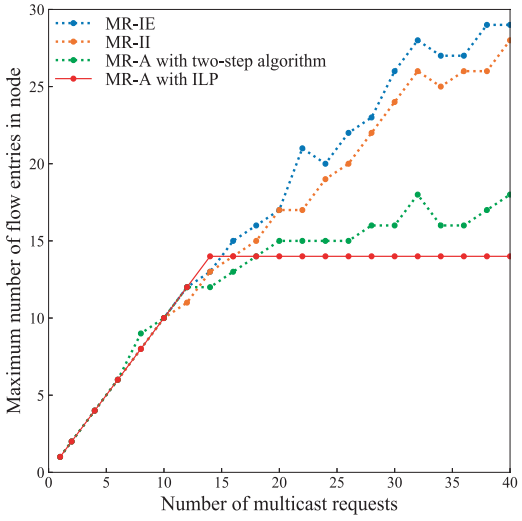

Fig. 8 Relationship between number of multicast requests and maximum number of flow entries in node in all trials in NSF network.

2. As shown in Figs. 7 and 8, as the number of multicast requests increases, the maximum number of flow entries in a node with MR-A with two-step algorithm and that with MR-A with ILP become smaller than those with MR-IE and MR-II. When the number of multicast requests is 40, the maximum number of flow entries stored in a node with MR-IE is the largest, followed by MR-II, MR-A with two step algorithm, and then MR-A with ILP. These trends of Figs. 7 and 8 are similar to the result of Fig. 5. Figures 7 and 8 show that the maximum number of flow entries stored in a node when each multicast tree is determined individually tends to be larger than that when all multicast trees are determined simultaneously. This is because multicast entries tend to be stored in order to decrease the number of flow entries in each multicast tree with MR-IE and MR-II, whereas unicast entries are preferentially stored if that promotes the sharing

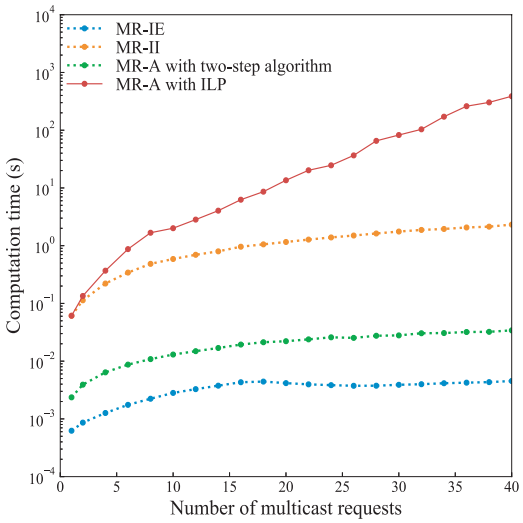

Fig. 9 Relationship between number of multicast requests and computation time in NSF network.

of unicast entries which suppresses the total number of flow entries with MR-A. In Fig. 8, the maximum number of flow entries per node with MR-A with ILP is less than or equal to 14 since multicast entries are not required to be stored when unicast entries dedicated to all receivers are stored in the node.

5.4 Computation Time

Figure 9 shows the computation times required to obtain the multicast trees and the type of flow entries in each model. MR-A with ILP takes the largest computation time, followed by MR-II, MR-A with the two-step algorithm and MR-IE. Compared with MR-II, MR-A with the two-step algorithm, and MR-IE, the computation time of MR-A with ILP increases rapidly with the number of multicast requests increasing.

5.5 Number of Flows Transferred in Link

Figures 10 and 11 show the relationship between the number of multicast requests and the maximum number of flows in a link. Figure 10 shows the maximum number of flows in a link averaged over 200 trials for each multicast request number. Figure 11 shows the maximum number of flows in a link observed in 200 trials for each multicast request number. Note that each flow transferred over a link is counted as one flow regardless of its type, i.e., multicast flow or unicast flow. As shown in Figs. 10 and 11, as the number of multicast requests increases, the number of flows in a link with MR-A rapidly increases compared with MR-ID and MR-II. This is because MR-A tends to determine multicast trees that branch near the source node, which leads to an increase in the number of flows in a network. As a result, the number of flows in a link becomes large. As described in Sect. 3.1, we focus on suppressing the total number of flow entries in MR-A. If it is required to suppress the number of flows in a

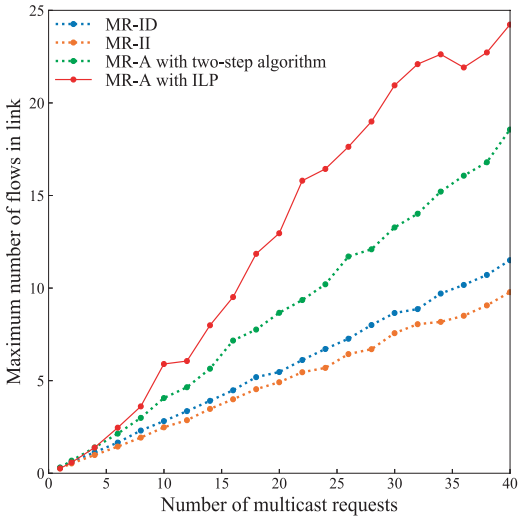

Fig. 10 Relationship between number of multicast requests and maximum number of flows in link when averaging results of all trials in NSF network.

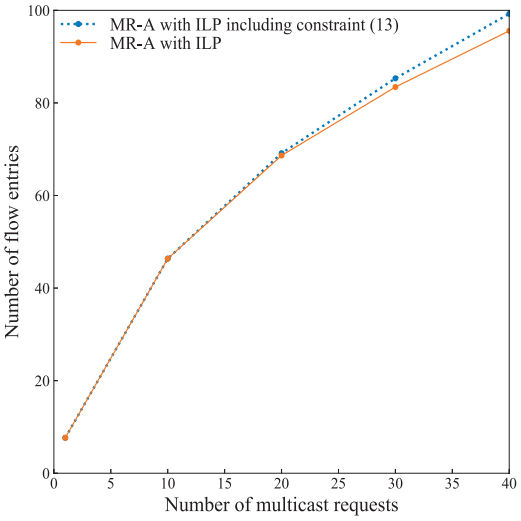

Fig. 12 Difference of total number of flow entries by adding constraint (13) to MR-A with ILP in NSF network.

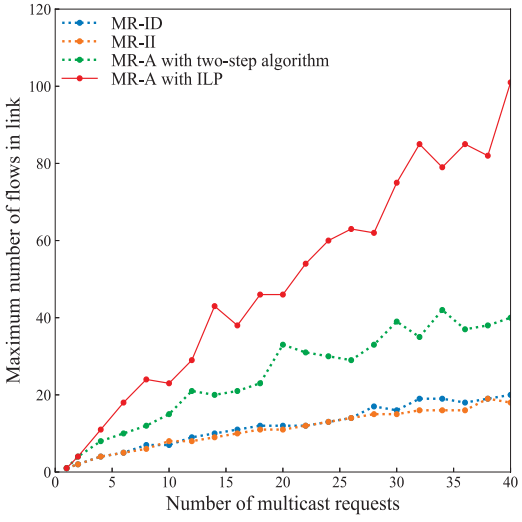

Fig. 11 Relationship between number of multicast requests and maximum number of flows in link in all trials in NSF network.

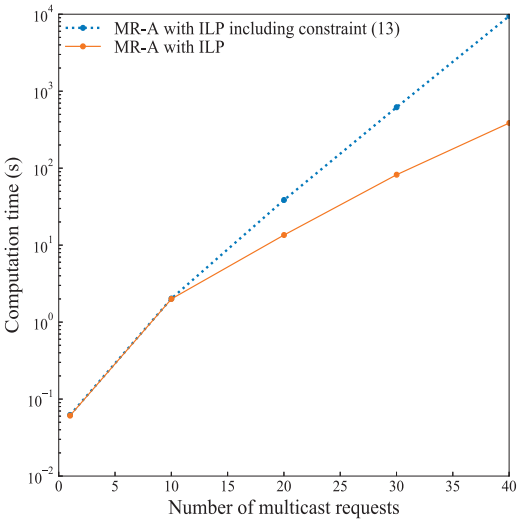

Fig. 13 Difference of computation time by adding constraint (13) to MR-A with ILP in NSF network.

link, we can add a new constraint to MR-A.

5.6 Constraint of Maximum Number of Flow Entries in Node

We evaluate the effect of maximum number of flow entries that can be stored in a node on the performance of MR-A in this section. A constraint of the maximum number of flow entries stored in a node is given by:

$$\sum_{r \in R} x_{rv} + \sum_{k \in K} y_{kv} \leq M, \forall v \in V. \tag{13}$$

Equation (13) indicates that the number of flow entries stored in node $v \in V$ is less than or equal to M. In this evaluation, we set M to $|V| - 1$ in consideration of the result of MR-A with ILP in Fig. 8; at most 14 flow entries, each of which is a unicast entry dedicated to each receiver node, are required to be stored in each node to provide all multicast trees.

Figure 12 shows the total number of flow entries when we obtain the multicast trees by applying MR-A with ILP and that by applying MR-A with ILP including constraint (13). The total number of flow entries increases by 3.69 when the number of multicast requests is 40 by adding constraint (13) to MR-A with ILP. This is because the number of hops between a sender and a receiver in the determined multicast trees can increase when constraint (13) is added since constraint (13) prohibits unicast entries dedicated to all nodes from being stored in a node. Figure 13 shows the computation times required to obtain the multicast trees and the type of flow entries in MR-A with ILP and MR-A with ILP including constraint (13). As the number of multicast requests increases, the computation time increases more rapidly in

MR-A with ILP including constraint (13) than MR-A with ILP.

6. Conclusion

In this paper, we proposed a routing model for determining multicast trees that minimizes the total number of flow entries for multiple multicast requests in an SDN. When multicast trees of multiple multicast requests are determined based on the conventional scheme, further reductions in the number of flow entries are possible since multicast trees are determined individually. The proposed model determines all multicast trees simultaneously and promotes the sharing of unicast entries among different multicast trees. We formulated the proposed model as an ILP problem and developed a heuristic algorithm which can be used in the case that the ILP problem is intractable. We evaluated the proposed model by solving the ILP problem and using the heuristic algorithm. Simulation results showed that the proposed model reduces the total number of flow entries more effectively than the benchmark models. The proposed model is intended for the static scenario, in which the sender and the set of receivers in each multicast request are known in advance. In practical network operations, the proposed model can be applied to situations where routes of multiple multicast requests need to be optimized, such as the beginning of network operation or scheduled network maintenance. As a future work, we will consider the dynamic scenario, where receivers are added to or deleted from multicast trees.

Acknowledgments

This work was supported in part by JSPS KAKENHI, Japan, under Grant Numbers 18H03230 and 19K14980.

References

[1] T. Kusunoki, T. Kurakake, K. Otsuki, and K. Saito, "Improvement of 4K/8K multi-channel IP multicast using DOCSIS over in-building coaxial cable network," 2019 IEEE International Conference on Consumer Electronics (ICCE), pp.1–5, Jan. 2019.
[2] L. Yen, M. Wang, S. Wu, and C. Tseng, "PIM-compliant SDN-enabled IP multicast service," NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium, pp.1–4, April 2018.
[3] O. Komolafe, "IP multicast in virtualized data centers: Challenges and opportunities," 2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM), pp.407–413, May 2017.
[4] T. Hardjono and B. Cain, "Key establishment for IGMP authentication in IP multicast," 1st European Conference on Universal Multiservice Networks. ECUMN'2000 (Cat. no.00EX423), pp.247–252, Oct. 2000.
[5] B. Cain, S. Deering, I. Kouvelas, B. Fenner, and A. Thyagarajan, "Internet group management protocol, version 3," RFC 3376 (Proposed Standard), Oct. 2002.
[6] S. Islam, N. Muslim, and J.W. Atwood, "A survey on multicasting in software-defined networking," IEEE Commun. Surveys Tuts., vol.20, no.1, pp.355–387, Firstquarter 2018.
[7] L. Huang, H. Hsu, S. Shen, D. Yang, and W. Chen, "Multicast traffic engineering for software-defined networks," IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications, pp.1–9, April 2016.
[8] C. Zhang, H. Yang, and G.F. Riley, "Admission control in software-defined datacenter network in view of flow table capacity," IEEE INFOCOM 2018 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), pp.871–876, April 2018.
[9] X. Jia, Y. Jiang, Z. Guo, and Z. Wu, "Reducing and balancing flow table entries in software-defined networks," 2016 IEEE 41st Conference on Local Computer Networks (LCN), pp.575–578, Nov. 2016.
[10] X. Zhang, S. Yu, Z. Xu, Y. Li, Z. Cheng, and W. Zhou, "Flow entry sharing in protection design for software defined networks," GLOBECOM 2017 - 2017 IEEE Global Communications Conference, pp.1–7, Dec. 2017.
[11] P.M. Mohan, T. Truong-Huu, and M. Gurusamy, "TCAM-aware local rerouting for fast and efficient failure recovery in software defined networks," 2015 IEEE Global Communications Conference (GLOBECOM), pp.1–6, Dec. 2015.
[12] T. Humernbrum, B. Hagedorn, and S. Gorlatch, "Towards efficient multicast communication in software-defined networks," 2016 IEEE 36th International Conference on Distributed Computing Systems Workshops (ICDCSW), pp.106–113, June 2016.
[13] Y.D. Lin, Y.C. Lai, H.Y. Teng, C.C. Liao, and Y.C. Kao, "Scalable multicasting with multiple shared trees in software defined networking," J. Netw. Comput. Appl., vol.78, no.C, pp.125–133, Jan. 2017.
[14] S. Kotachi, T. Sato, R. Shinkuma, and E. Oki, "Multicast routing model to minimize number of flow entries in software-defined network," 2019 20th Asia-Pacific Network Operations and Management Symposium (APNOMS), pp.1–6, Sept. 2019.
[15] B. Zhang and H.T. Mouftah, "A destination-driven shortest path tree algorithm," 2002 IEEE International Conference on Communications. Conference Proceedings. ICC 2002 (Cat. no.02CH37333), pp.2258–2262, April 2002.
[16] M.R. Garey and D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W.H. Freeman & Co., New York, NY, USA, 1979.
[17] R. Hulsermann, A. Betker, M. Jager, S. Bodamer, M. Barry, J. Spath, C. Gauger, and M. Kohn, "A set of typical transport network scenarios for network modelling," ITG FACHBERICHT, pp.65–72, Jan. 2004.
[18] C.L. Triveni, P.C. Srikanth, and T. Srinivas, "An optimized routing algorithm for elastic optical network," 2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT), pp.3873–3878, March 2016.
[19] "NSFNET: A partnership for high-speed networking: Final report," https://www.merit.edu/wp-content/uploads/2019/06/NSFNET_final-1.pdf, Accessed: 2020-4-17.
[20] "CPLEX optimization studio," https://www.ibm.com/products/ilog-cplex-optimization-studio, Accessed: 2020-4-15.

Seiki Kotachi is currently pursuing the master degree at the Graduate School of Informatics, Kyoto University, Kyoto, Japan. He received the B.E. in Undergraduate School of Electrical and Electronic Engineering, Kyoto University, Kyoto, Japan, in 2019. His research interests include software-defined network, modeling, optimization, and algorithms.

Takehiro Sato received the B.E., M.E. and Ph.D. degrees in engineering from Keio University, Japan, in 2010, 2011 and 2016, respectively. He is currently an assistant professor in the Graduate School of Informatics, Kyoto University, Japan. His research interests include communication protocols and network architectures for the next generation optical network. From 2011 to 2012, he was a research assistant in the Keio University Global COE Program. From 2012 to 2015, he was a research fellow of Japan Society for the Promotion of Science. From 2016 to 2017, he was a research associate in Graduate School of Science and Technology, Keio University, Japan. He is a member of the IEEE.

Ryoichi Shinkuma received B.E., M.E., and Ph.D. degrees in communications engineering from Osaka University in 2000, 2001, and 2003, respectively. In 2003, he joined the Communications and Computer Engineering Faculty of the Graduate School of Informatics, Kyoto University, where he is currently an associate professor. He was a visiting scholar in the Wireless Information Network Laboratory, Rutgers University from Fall 2008 to Fall 2009. His research interest is mainly cooperation in heterogeneous networks. He received the Young Researchers' Award from IEICE in 2006 and the Young Scientist Award from Ericsson Japan in 2007. He also received the TELECOM System Technology Award from the Telecommunications Advancement Foundation in 2016 and the best tutorial paper award from the IEICE Communications Society in 2019. He was the Chairperson of the Mobile Network and Applications Technical Committee of the IEICE Communications Society from 2017 to 2019. He is a fellow of IEICE and a senior member of IEEE.

Eiji Oki is a Professor at Kyoto University, Japan. He received the B.E. and M.E. degrees in instrumentation engineering and a Ph.D. degree in electrical engineering from Keio University, Yokohama, Japan, in 1991, 1993, and 1999, respectively. He was with Nippon Telegraph and Telephone Corporation (NTT) Laboratories, Tokyo, from 1993 to 2008, and The University of Electro-Communications, Tokyo, from 2008 to 2017. From 2000 to 2001, he was a Visiting Scholar at Polytechnic University, Brooklyn, New York. In 2017, he joined Kyoto University, Japan. His research interests include routing, switching, protocols, optimization, and traffic engineering in communication and information networks. He is an IEEE Fellow.