Journal of Network and Computer Applications 78 (2017) 125–133

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

Scalable multicasting with multiple shared trees in software defined networking

Ying-Dar Lin[a], Yuan-Cheng Lai[b], Hung-Yi Teng[a,*], Chun-Chieh Liao[a], Yi-Chih Kao[c]

[a] Dept. of Computer Science, National Chiao Tung University, Hsinchu, Taiwan
[b] Dept. of Information Management, National Taiwan University of Science and Technology, Taipei, Taiwan
[c] Information Technology Service Center, National Chiao Tung University, Hsinchu, Taiwan

ARTICLE INFO

Keywords:
Software defined networking
Multiple shared trees
Locality-aware
Multicast clustering
Rendezvous point

ABSTRACT

With Software Defined Networking (SDN), IP multicast becomes promising again. For IPTV applications over SDN, existing works would not scale well since they are based on per-source trees. As control-plane in SDN is logically centralized, constructing multiple shared trees is more feasible than that in traditional IP networks. Thus, in this work, we present a locality-aware multicast approach (LAMA) to construct multi-group shared trees in SDN, where each shared tree covers multiple multicast groups. In LAMA, the controller first clusters the multicast sources located in the vicinity into the same multicast cluster. For each multicast cluster, the controller selects the center switch which has the minimum distance to all multicast sources as its rendezvous point (RP) and then constructs a shortest-path multicast tree from the RP to its hosts. Finally, based on the multi-group shared trees, the controller can establish coarse-grained flow entries into on-tree switches to reduce the number of installed flow entries. Emulations on the Ryu controller and the Mininet emulator show that only 2–5 shared trees would suffice. The computation time in the controller using LAMA is around 70 ms, much less than hundreds ms required for per-source trees. Moreover, LAMA only establishes 2300 flow entries, 4% of that with per-source trees in a large topology.

1. Introduction

Video streaming services, such as Internet Protocol TV (IPTV) and video conferencing, have become popular applications among users and have contributed a significant amount of Internet traffic. This type of applications usually requires high-bandwidth and low-delay video transmissions from a source to a large population of users. For these applications, IP multicast is the most efficient vehicle, avoiding the waste of bandwidth.

In traditional networks, due to the distributed nature, each router needs to perform the multicast routing algorithm specified in multicast routing protocols, such as Distance Vector Multicast Routing Protocol (DVMRP) (Waitzman et al., 1988), Protocol Independent Multicast-Dense Mode (PIM-DM) (Adams et al., 2005), Protocol Independent Multicast-Sparse Mode (PIM-SM) (Farinacci et al., 1998), or Multicast Open Shortest Path First (MOSPF) (Moy, 1994). These multicast routing protocols are responsible for determining how multicast packets are disseminated to avoid redundancy of information and prevent loop. Therefore, each router requires keeping a lot of duplicated states and exchanging information with neighboring routers to update its routing table. Furthermore, each router may be responsible for managing group events, e.g. host join or leave, through the Internet Group Management Protocol (IGMP) (Fenner, 1997). Consequently, deploying IP multicast in traditional networks would severely degrade the performance of networks and remains unpractical.

1.1. Motivation

Software defined networking (SDN) has emerged as a clean-slate approach, which enable us to reconsider the design of video streaming services based on IP multicast. In SDN, the functionalities of the control plane (decide how to handle traffic) are separated from switches and is managed by a logical centralized controller. Thus, the switches only have the functionalities of the data plane (forward traffic according to the decisions made by the control plane). The controller can support various applications to manage underlying switches via south-bound application programming interfaces (APIs) such as OpenFlow (McKeown et al., 2008). Under SDN context, (1) the switches no longer bear the burden induced by multicast routing protocols and group management protocols; (2) based on the full knowledge of network conditions, the controller is capable to build an optimal multicast tree and handle group events efficiently. Therefore,

* Corresponding author.
E-mail address: marstwng@gmail.com (H.-Y. Teng).

http://dx.doi.org/10.1016/j.jnca.2016.11.014
Received 29 October 2015; Received in revised form 14 November 2016; Accepted 15 November 2016
Available online 19 November 2016
1084-8045/ © 2016 Elsevier Ltd. All rights reserved.

using SDN makes IP multicast become a possible alternative for video streaming services.

Designing multicast routing algorithms in SDN has some differences with that in traditional networks. First, in traditional networks, multicast trees are built by the distributed routers, but in SDN, the controller centrally builds multicast trees and handles group events, significantly adding SDN controller's load. Second, the router keeps stateless unicast routing and stateful multicast routing only, so it does not need to store much information. However, in SDN, the switch maintains the states for unicast flows, causing that the flow entries are precious resources. Thus the flow entries for multicast trees should be kept as few as possible while keeping the desired QoS. Therefore, the scalability issues including computation time in the controller and the number of flow entries consumed by the multicast trees in the switches are main concerns in designing multicast routing algorithms in SDN.

Up to now, there are few investigations on IP multicast using SDN (Bondan et al., 2013; Marcondes et al., 2012; Zhao et al., 2014). Bondan et al. (2013) proposed a multicast approach in which the calculation of a route between a source and a host is performed when the controller receives an IGMP join message. This approach yields a longer initial latency when a large number of hosts joins simultaneously. In Marcondes et al. (2012), the authors presented a multicast approach in which the calculation of all possible routes from sources to hosts is done in advance. Thus, the initial latency can be reduced greatly. In Zhao et al. (2014), the authors proposed a congestion-free multicast approach for multi-party video conferencing. In the approach, delay-bounded multicast trees are built by either rerouting congested links or adjusting the video streaming rate. However, aforementioned approaches cannot handle a large number of multicast groups because they are based on per-source trees, i.e. each source owns a multicast tree. More specifically, when IPTV applications adopted aforementioned approaches, the computation time of the controller and the number of flow entries installed in the switches would be increased greatly and become performance bottlenecks. In order to address the scalability issues, the number of multicast trees should be minimized while maintaining a desirable video quality.

Fundamentally, multicast trees can be classified into two types: per-source tree and shared tree. The per-source tree means that each multicast source has its own multicast tree while a shared tree is a multicast distribution tree that are shared by multiple group(s). Note that a multicast group which contains a set of multicast sources and a set of hosts is usually used to multicast a video channel. Because control-plane are logical centralized in SDN, we can have three alternatives of shared tree: per-group shared tree, multi-group shared tree, and single shared tree. Firstly, the per-group shared tree represents that the multicast sources in the same multicast group share the same multicast tree. Secondly, the multi-group shared tree means that several multicast groups share the same multicast tree. Lastly, the single shared tree is an extreme case in which all the multicast sources share the same multicast tree.

Given a system with N multicast groups and each multicast group has M multicast sources, per-source tree and per-group shared tree would yield $N \times M$ and N multicast trees, respectively. Thus, the major drawback of the per-source tree and the per-group shared tree is that the controller and the switches need to handle a large number of multicast trees. On the other hand, the single shared tree requires only one multicast tree, but it would suffer from the longest end-to-end latency and on-tree switches would have very heavy workloads. Therefore, we believe that the multi-group shared tree would be a better approach for SDN because only N/α multicast trees are needed where α denotes the clustering factor. Table 1 shows a comparison of four types of multicast trees.

Table 1
A Comparison of Multicast Trees.

	per-source tree	per-group shared tree	multi-group shared tree	single shared tree
Number of trees	$N \times M$	N	N/α	1
Computation time	Long	Medium	Medium	Short
Number of states	Many	Few	Few	Very Few
Load on the on-tree switches	Light	Medium	Medium	Heavy
End-to-end distance	Short	Medium	Medium	Long

1.2. Contributions

In this paper, we propose a locality-aware multicast approach (LAMA) to achieve better scalability while meeting stringent requirements of video streaming services. In the LAMA, several multicast groups are adaptively clustered into a multicast cluster and a shortest-path multi-group shared tree is built for each multicast cluster. Since generating optimal shared multicast trees is an NP complete problem (Zappala and Fabbri, 2001), we divide the problem into three independent sub-problems (multicast group clustering, rendezvous point (RP) selection, and multicast tree construction) and solve these sub-problems separately. In the multicast group clustering, we propose a distance-based clustering algorithm in which the multicast sources located in the vicinity can be clustered into the same multicast cluster. In the RP selection, we present a locality-aware selection algorithm to determine a proper RP which has the minimum distance to all multicast sources within a multicast cluster. Finally, a shortest-path multi-group multicast tree is constructed from the selected RP to the hosts for each multicast cluster. Based on the multi-group shared trees, coarse-grained flow entries are inserted into on-tree switches, greatly reducing the number of required flow entries.

Specifically, this paper has the following contributions:

(1) LAMA is proposed to construct multi-group shared trees for IP multicasting in SDN.
(2) The number of flow entries in switches and the computation time in the controller for constructing multicast trees is small to keep excellent scalability of SDN.
(3) Numerous emulations are performed to compare LAMA with other approaches, including the per-source tree approach and the single shared tree approach, to verify its outperformance on scalability.

The remainder of this paper is organized as follows. Section 2 discusses the related works. Section 3 gives an overview of our system model and problem formulation. We present our approach, LAMA, in Section 4. Section 5 demonstrates our experimental results. Finally, the conclusions are given in Section 6.

2. Related work

In this section, we describe the related studies that have focused on IP multicasting using SDN. In Bondan et al. (2013), the authors proposed an approach for multimedia multicasting, in which the shortest-path route between the source and the client was computed on demand to decrease the end-to-end latency. But, when a large number of clients join the system simultaneously, this approach produces long initial latency. Marcondes et al. (2012) also presented a multicast approach, in which the controller calculates all possible routes from sources to group members in advance, so that initial latency could be reduced. In Zhao et al. (2014), the authors proposed a novel multicast construction and packing method for multiparty video conferencing, in which the SDN controller constructs multicast trees for video flows. In that method, multiple source-based multicast trees are constructed, which aims to maximize system-wide utility and guarantee an end-to-end distance bound. However, because the aforementioned works used source-based trees, when the numbers of sources became large, the system would face two scalability issues: (1) the controllers were required to maintain large numbers of multicast trees ; (2) these approaches consumed substantial amounts of space on the flow tables of the related switches.

Cui and Qian (2014) proposed a dual-structure multicast approach in which the controller classifies the multicast groups into two groups, mice and elephants, based a predefined threshold of bitrate. For mice, the traffic is forwarded using unicast so that the switches do not retain the flow entries for mice. For elephants, the scheme constructs multiple shared trees for each group depending on whether all group members are in the same pod or not. Compared to using source-based trees, using shared multicast trees is a better approach to reduce controller overhead as well as switch overhead. However, in their use case, using per-group shared trees is only suitable for rack area networks (i.e. datacenter). In a more generic network, using per-group shared trees may still cause noticeable controller overhead and consume considerable switches' flow entries when the number of groups is large.

Therefore, in this paper, we propose to use multi-group shared trees for arbitrary networks. By constructing multi-group shared trees with end-to-end distance bound, our approach can enhance the scalability of the controller and the switches in terms of both the computation time and the number of flow entries while maintaining a desirable video quality. Moreover, our approach can reduce initial latency because it can preset the flow entries in related switches. The aforementioned works are summarized in Table 2.

3. System model

Let $G = \{V, E\}$ denote the network topology where V is a set of nodes and E is a set of links representing the connections between nodes. There are three type of nodes: switch S, multicast source M, and host H. Let $|S|$, $|M|$, $|H|$, and $|E|$ be the number of switches, multicast sources, hosts, and links, respectively. Each switch s_i has its traffic load $L(s_i)$ and the bandwidth capacity of each switch is equal to B. Each multicast source m_k has its streaming rate r_k. For each network link $e_{u,v}$ where $u \in V$ and $v \in V$, $D(e_{u,v})$ denotes the distance of the link e_{uv}. Let $P(u, v) = \{e_{u,a}, e_{a,b}, \ldots, e_{c,v}\}$ denotes the path from the node u to node v. The distance of the path $P(u, v)$ is the sum of distance of all links in $P(u, v)$ as

$$D(P(u, v)) = \sum_{e \in P(u,v)} D(e). \tag{1}$$

A tree $T(s, H)$ is a sub-graph of the network topology G spanning from a node s to the set of nodes H. The distance of the tree $T(s, H)$ is defined as follows:

$$D(T(s, H)) = \max D(P(s, h)),\ \forall\ h \in H. \tag{2}$$

Let $g_k = \{m_k, H_k\}$ denote a multicast group k which contains a multicast source m_k and the hosts H_k, which subscribe the streaming from the multicast source m_k. We assume that multicast sources and hosts are registered to the controller in advance. Here, we define a multicast cluster c_n, $1 \le n \le |M|$, which is a set of multicast groups. Let $I_{n,k}$ be an indicator where

$$I_{n,k} = \begin{cases} 1 & \text{if multicast cluster } c_n \text{ includes multicast group } g_k \\ 0 & \text{otherwise} \end{cases}. \tag{3}$$

Therefore, $c_n = \bigcup_{k=1}^{|M|} I_{n,k} \cdot g_k$. For a multicast cluster c_n, the aggregated streaming rate is defined as

$$R_{agg}(c_n) = \sum_{g_k \in c_n} r_k. \tag{4}$$

The maximum distance among multicast sources in the cluster c_n is defined as

$$D_{\max}(c_n) = \max D(P(m_u, m_v)), \quad \forall\ m_u \in c_n \text{ and } \forall\ m_v \in c_n. \tag{5}$$

In additional, the distance of two clusters c_A and c_B is defined as

$$D(c_A, c_B) = \max D(P(m_u, m_v)), \quad \forall\ m_u \in c_A \text{ and } \forall\ m_v \in c_B. \tag{6}$$

Let $T(c_n)$ be the shared tree of the multicast cluster c_n. According to Eqs. (1) and (2), the distance of the shared tree $T(c_n)$ is defined as

$$D(T(c_n)) = \sum_{g_k \in c_n} D(P(m_k, rp_n)) + D(T(rp_n, H_k)). \tag{7}$$

Finally, the number of flow entries of the shared tree is defined as $F(T(c_n))$. Table 3 shows the notations used in this study.

Given a network topology $G = \{V, E\}$, the bandwidth capacity of switch B, and the streaming rate r_k of multicast source m_k, we aim to (1) minimize the number of shared trees by clustering multicast groups, (2) minimize the distance of each shared tree $D(T(c_n))$ by selecting a proper rendezvous point rp_n, and (3) minimize the number of flow entries of the shared tree $F(T(c_n))$. The constraints of our problem are (1) the maximum distance among multicast sources in each cluster c_n must be smaller than a predefined threshold, D_{th}, that is $D_{\max}(c_n) < D_{th}$, and (2) each switch cannot be overloaded, that is $L(s_i) \le B$.

4. Locality-aware multicast approach

To solve the problem, we propose a locality-aware multicast approach (LAMA), which includes three-stages: multicast source clustering, RP selection, and multicast tree construction. In the multicast source clustering, we propose a distance-based clustering algorithm to cluster the multicast sources located in the vicinity into a multicast cluster. For each multicast cluster, we present a locality-aware selection algorithm to select the center switch which has the minimum distance to all multicast sources in a multicast cluster as a proper RP. Then, a shortest-path multicast tree is constructed from the selected RP to the hosts. Furthermore, the video data transmitted by the same shared multicast tree is forwarded by matching the same flow entry so that the number of flow entries installed on the related switches can be greatly reduced.

Table 2
A Comparison of related works.

	Multiflow (Bondan et al., 2013)	CastFlow (Marcondes et al., 2012)	MTCP (Zhao et al., 2014)	DuSM (Cui and Qian, 2014)	Ours work
Topology	Generic	Generic	Generic	Fat Tree	Generic
Type of multicast tree	Per-source tree	Per-source tree	Per-source tree	Per-group shared tree	Multi-group shared tree
End-to-end distance	Shortest	Shortest	Bounded	Bounded	Bounded
Flow entries in RP	0	0	0	Depends on topology	O(|M|)
Flow entries in on-tree switch	O(|M|)	O(|M|)	O(|M|)	Depends on topology	O(|RP|)
Initial latency	High	Medium	Medium	Low	Low

Table 3
Description of notations.

Categories	Notation	Type	Descriptions
Topology	$G = \{V, E\}$	Input	The network topology
Nodes	$V = \{S, M, H\}$	Input	The set of network nodes
	$S = \{s_i \mid i \geq 1\}$	Input	The set of switches
	$M = \{m_k \mid k \geq 1\}$	Input	The set of multicast sources
	$H = \{h_j \mid j \geq 1\}$	Input	The set of hosts
	$L(s_i)$	Variable	The traffic load of switch s_i
	B	Input	The bandwidth capacity of each switch
	$R = \{r_k \mid k \geq 1\}$	Input	The set of streaming rate of multicast source m_k
	$g_k = \{m_k, H\}$	Input	The multicast group from the multicast source m_k
	$C = \{c_n \mid 1 \leq n \leq \lvert M \rvert\}$	Variable	The set of multicast clusters
	$RP = \{rp_n \mid 1 \leq n \leq \lvert M \rvert\}$	Variable	The set of rendezvous points
Links	$E = \{e_{u,v} \mid u \in V, v \in V\}$	Input	The set of network links
	$D(e_{u,v})$	Input	The link distance of link $e_{u,v}$
	D_{th}	Input	The distance threshold
Path	$P(u, v) = \{e_{u,a}, e_{a,b}, \ldots, e_{c,v}\}$	Output	The path from node u to node v, where $a, b, c \in V$
Tree	$T(c_n)$	Output	The shared tree of multicast cluster c_n
	$F(T(c_n))$	Output	The number of flow entries of shared tree $T(c_n)$
Decision variable	$I_{n,k}$	Variable	A multicast cluster c_n includes multicast group g_k or not

4.1. Detailed algorithms

4.1.1. Multicast source clustering

Fig. 1 shows pseudo code of the distance-based clustering algorithm. This algorithm is based on complete linkage clustering (Johnson, 1967), which iteratively decides whether two clusters should be merged or not according to the distance between clusters. In complete linkage clustering, the distance between clusters equals the distance between those two multicast sources (one in each cluster) which are farthest away from each other. Therefore, the multicast sources located in the vicinity can be clustered in the same multicast cluster. In this algorithm, each multicast source forms an individual cluster in the beginning. Next, we select two multicast clusters c_A and c_B which have the minimum distance according to Eq. (6). Then we check (1) whether the distance between the multicast clusters c_A and c_B is smaller than the distance threshold D_{th} and (2) whether the aggregated streaming rate is smaller than the bandwidth capacity. If the conditions (1) and (2) are met, the multicast clusters c_A and c_B are merged together and the procedure is executed repeatedly. Otherwise, either the procedure is aborted (when the condition (1) is not met) or the multicast clusters c_A and c_B are removed from the set of multicast clusters (when the condition (2) is not met).

```
Algorithm 1 Near Source Clustering
Input:
    $G = (V, E)$: network topology
    $R$: set of streaming rate
    $B$: switch bandwidth

Output:
    $C = \{c_n \mid 1 \leq n \leq |M|\}$: set of multicast clusters

1: $C = \{c_n \mid 1 \ldots |M|\}$
2: while True do
3:     select min $D(c_A, c_B)$, $c_A \neq c_B$
4:     if $D(c_A, c_B) < D_{th}$ then
5:         if $R_{agg}(c_A) + R_{agg}(c_B) \leq B$ then
6:             $c_A = c_A \bigcup c_B$
7:         else
8:             remove $(c_A, c_B)$
9:         end if
10:    else
11:        break
12:    end if
13: end while
```

Fig. 1. Pseudo code of the distance-based clustering algorithm.

4.1.2. RP selection

Inspired by Zappala and Fabbri (2001), the main idea of the locality-aware RP selection algorithm is to select the switch that has the minimum sum of distance to multicast sources as a proper RP from a set of candidate RPs. Fig. 2 shows the pseudo code of the locality-aware RP selection algorithm. In this algorithm, given the network topology G and the set of multicast clusters C as the inputs, we first select all source-attached switches as candidate RPs for each multicast cluster c_n. Then, for each candidate RP, we calculate the distances from each candidate RP to all multicast sources in this cluster. Finally, we select the switch which has the minimum distance to all multicast sources as the RP, rp_n, for each multicast cluster c_n.

4.1.3. Multicast tree construction

In order to reduce the end-to-end latency, a shortest-path multicast tree is constructed for each multicast cluster. Fig. 3 shows pseudo code of the shortest-path tree construction algorithm. In this algorithm, we

```
Algorithm 2 Source Center RP Selection
Input:
    $G = (V, E)$: network topology
    $C = \{c_n \mid 1 \leq n \leq |M|\}$: set of multicast cluster

Output:
    $RP$: set of rendezvous points

1: $RP = \phi$
2: for $n = 1$ to $|M|$ do
3:     $CRP = \{$all source-attached switches in $c_n\}$
4:     select switch $s_i$ which has min $\sum_{m_k \in c_n} D(P(m_k, s_i))$ as $rp_n$, where $s_i \in CRP$
5:     $RP = RP \bigcup \{rp_n\}$
6: end for
```

Fig. 2. Pseudo code of the locality-aware RP selection algorithm.

Algorithm 3 Shortest-path Tree Construction

Construct shortest path tree $\mathrm{T}(c_n)$ spanning $\{m_k \in c_n\} \bigcup H$ using Dijkstra's algorithm

```
Input:
    $G = (V, E)$: network topology
    $C = \{c_n \mid 1 \le n \le |M|\}$: set of multicast clusters
    $RP = \{rp_n \mid 1 \le n \le |M|\}$: set of rendezvous points

Output:
    $T(c_n)$: shared tree for multicast cluster $c_n$

1: $T(c_n) = \phi$
2: for $n = 1$ to $|M|$ do
3:     for $\forall m_k \in c_n$ do
4:         $P(m_k, rp_n) = shortest\_path(m_k, rp_n)$
5:         $\mathrm{T}(c_n) = \mathrm{T}(c_n) \bigcup P(m_k, rp_n)$
6:     end for
7:     $T(rp_n, H) = shortest\_path\_tree(rp_n, H)$
8:     $\mathrm{T}(c_n) = \mathrm{T}(c_n) \bigcup T(rp_n, H)$
9: end for
```

Fig. 3. Pseudo code of the shortest-path tree construction algorithm.

first select the multicast cluster with the highest aggregated streaming rate to construct the multicast tree because the multicast cluster with the higher aggregated streaming rate would occupy more bandwidth. For each multicast cluster c_n, we use the Dijkstra's algorithm to construct the shortest paths from each multicast source $\forall\, m_k \in c_n$ to the RP rp_n, and then construct the shortest path tree from the RP rp_n to the hosts H_k. Based on multi-group shared-trees, we finally insert a coarse-grained flow entry into on-tree switches instead of several fine-grained flow entries. Note that our approach, LAMA, tries to construct multi-group shared trees, rather than per-source trees, to reduce the number of required flow entries while keeping the acceptable path length. This stage will construct the shortest paths from each multicast source in the same cluster to RP and also construct the shortest paths from RP to the hosts. Therefore, the path from a source to a host may not be the shortest path.

4.2. An example

An example shown in Fig. 4 is presented to explain the overall procedure of LAMA. Fig. 4(a) shows the network topology G. The streaming rates of multicast sources, m_1, m_2, m_3, and m_4, are 200 KB/s, 50 KB/s, 100 KB/s, and 200 KB/s, respectively. The switch bandwidth B and the distance threshold D_{th} are 300 KB/s and 5 hops, respectively.

First, every multicast source forms a multicast cluster and calculates the distance from one multicast cluster to the others. The original distance is shown in Fig. 4(b). Next, we select two multicast clusters which have the minimum distance, i.e., c_1 and c_3. The distance $D(c_1, c_3) = 3$ is smaller than $D_{th} = 5$, and the aggregated streaming rate is equal to the bandwidth capacity, $R_{agg}(c_1) + R_{agg}(c_3) = 200 + 100 \le B$. Therefore, we merge c_1 and c_3 together, i.e., $c_1 = c_1 \cup c_3$, $I_{3,3} = 0$, and $I_{1,3} = 1$. The updated distances between each cluster are calculated. The next two near clusters are c_1 and c_2. The distance $D(c_1, c_2) = 4 < D_{th} = 5$. However, the aggregated streaming rate is greater than the bandwidth capacity, $R_{agg}(c_1) + R_{agg}(c_2) = 300 + 50 > B$. Therefore, the pair (c_1, c_2) is removed. The next two near clusters are c_1 and c_4. The distance $D(c_1, c_4) = D_{th} = 5$, means that no clusters are near. Finally, the set of multicast clusters C is $\{c_1, c_2, c_4\}$, where $c_1 = \{g_1, g_3\}$, $c_2 = \{g_2\}$, and $c_4 = \{g_4\}$.

Then LAMA selects a proper RP for each cluster. For cluster c_1, there are two multicast groups in this cluster $c_1 = \{g_1, g_3\}$. The set of candidate RPs is $\{s_1, s_3\}$. We calculate the distances of s_1 and s_3 as $D(P(m_1, s_1)) + D(P(m_3, s_1)) = 1 + 2 = 3$ and $D(P(m_1, s_3)) + D(P(m_3, s_3)) = 2 + 1 = 3$, respectively. In this case, we randomly select s_1 as an RP. For cluster c_2 and c_4, we select s_6 and s_5, respectively. Therefore, the set of rendezvous points $RP = \{s_1, s_6, s_5\}$.

Finally, for cluster c_1, we construct two shortest paths $m_1 \to s_1$ and $m_3 \to s_3 \to s_1$. In additional, we construct a shortest-path tree from s_1 to all hosts, as shown in Fig. 4(c). Then the shortest paths for cluster c_2 and c_4 are also constructed correspondingly.

4.3. Implementation

Fig. 5 shows the architecture of our implementation. Based on OpenFlow specification 1.3.2 (OpenFlow Switch Specification Version 1.3.2), our approach is implemented on Ryu controller (RYU SDN Framework) as a controller module. In the LAMA module, there are four functional blocks: *topology discoverer*, *group membership manager*, *multicast tree constructor*, and *flow inserter*. The *topology discover* and the *group membership manager* are responsible for periodically acquiring global topology information and multicast group membership, respectively. The *multicast tree constructor* is to cluster all multicast sources into multiple clusters and to build a multi-group shared tree for each multicast cluster according to the proposed algorithms. The *flow inserter* is to insert the required flow entries into the related switches.

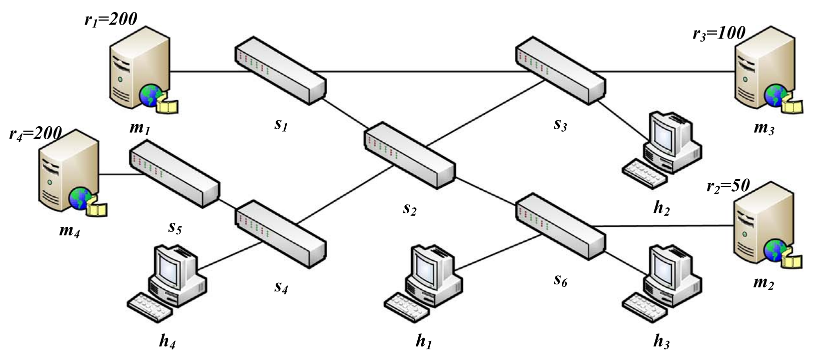
The flow entries are inserted into the related switches through the *modify_state* messages of *OFPMatch*() and *OFPInstruction*(). In our approach, these are two kinds of flow entries. One is from each multicast source to the RP, where the match field is "*ip_src=multicast_source_ip*, *ip_dst=multicast_group_ip*" and the action field is the output port transmitted to the next switch along with the shortest path. The other is from the RP to hosts, where the match field is "*vlan_id=x*" instead of "*ip_dst=multicast_group_ip*" to reduce the required flow entries, and the action field is the output port sent to on-tree switches. The reason is when several multicast trees have the same partial path, the switches on the path will record many flow entries when the match field is set as "*ip_dst=multicast_group_ip*". Therefore, the same path for different multicast trees should be aggregated into one flow entry in the switches on the path.

The technique of VLAN is adopted as our implementation for aggregating flow entries. We use *OFPActionPushVLan*() and *OFPActionSetField*() functions to insert the flow entries with action "*tag vlan_id=n*" at the RP and use *OFPActionOutput*() function to pass the packets to the next on-tree switch, where *n* is the index of the shared tree. By matching the VLAN tag, the on-tree switches can forward the packets to the hosts. Since the host can't recognize the packets with the VLAN tag, we use *OFPActionPopVLan*() to insert the flow entries with action "*untag vlan_id*" at edge switches. After popping the VLAN tag, the edge switch forwards the packets to the hosts by matching "*ip_dst=multicast_group_ip*".

5. Performance evaluation

5.1. Emulation environment

We employ two servers to build our emulation environment. The first server (i.e., Intel core i5-4590 CPU 3.30 GHz, Debian 7.8 on VMware workstation) acts as the OpenFlow controller, running the Ryu version 3.13. In the second server (i.e., Intel core i5-4440 CPU 3.10 GHz, Debian 7.8), we used the Internet topology generator BRITE to simulate the real topology because most multicast experiments in other researches also used BRITE to simulate the real topology. Given the number of hosts, switches, and sources, BRITE can generate the topology by creating proper links and their distances.



(a) Network topology

	c_1	c_2	c_3	c_4
c_1	0	4	3	5
c_2	4	0	4	5
c_3	3	4	0	5
c_4	5	5	5	0

(1) Original distance

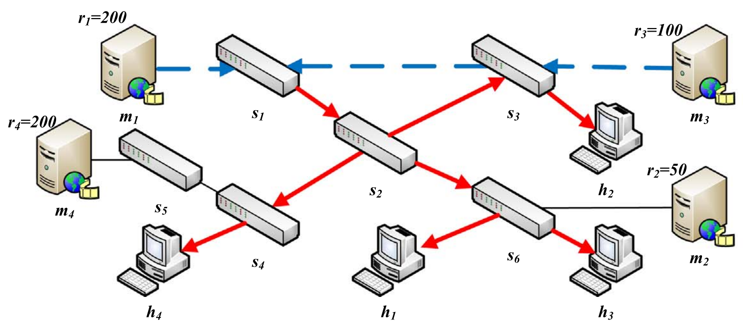
	c_1	c_2	c_4
c_1	0	4	5
c_2	4	0	5
c_4	5	5	0

(2) First update

	c_1	c_2	c_4
c_1	0	∞	5
c_2	∞	0	5
c_4	5	5	0

(3) Final

(b) Distances between each cluster



(c) Shared tree $T(c_1)$

Fig. 4. An example. (a) Network topology. (b) Distances between each cluster. (c) Shared tree $T(c_1)$.

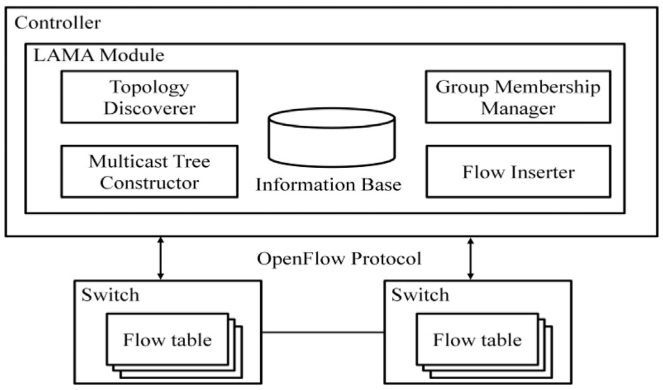


Fig. 5. Implementation architecture.

Mininet (Lantz et al., 2010) is used to emulate the topology composed of switches, multicast sources, and hosts. VLC tool is used to generate streaming data on multicast sources and receive streaming data on hosts. Table 4 shows the default setting of parameters used in the emulation environment.

In our experiments, we compare our approach, LAMA, with the per-source tree (PST) approach and the single shared tree (SST) approach. We do not compare LAMA with Multiflow (Bondan et al., 2013), CastFlow (Marcondes et al., 2012), and MTCP (Zhao et al., 2014) individually, which are listed in Table 2, because they all construct per-source trees and will have similar performance with PST. Although DuSM (Cui and Qian, 2014) constructs per-group shared trees, it will be reduced to construct per-source trees because a source corresponds to a group in our experiment. Therefore, DuSM is not considered in our experiment since its performance is also similar to PST and it is mainly applied to the fat-tree topology. The computation time and the total number of installed flow entries are adopted as performance metrics. The computation time shows the controller spends how much time to build the required multicast trees. The total number of installed flow entries indicates how many TCAM spaces consumed by the required multicast trees.

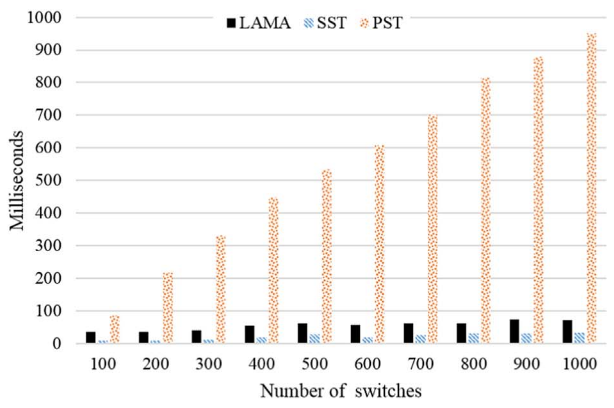
5.2. Results

5.2.1. Scalability of the control-plane

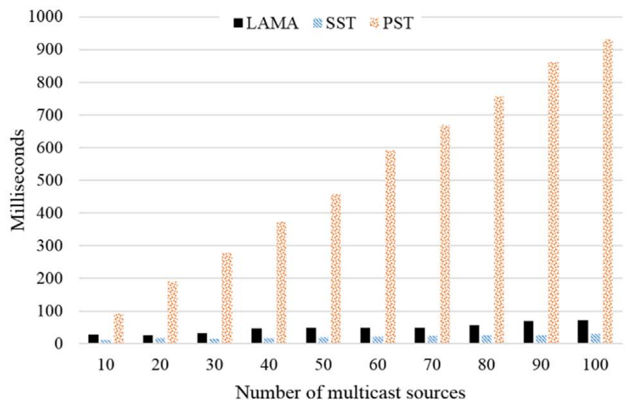
Firstly, we investigate the scalability of the proposed approach in the control-plane. Fig. 6 shows the average computation time of the proposed approach with various number of (a) switches, (b) multicast

Table 4
Defaulting setting of parameters.

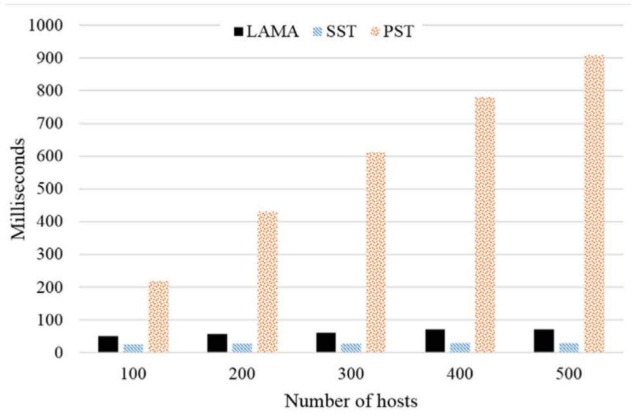
Notation	Meaning	Default setting
$\|S\|$	The number of switches	1000
..	The number of multicast sources	100
$\|H\|$	The number of hosts	500
B	The bandwidth capacity of switch	Generated by BRITE
$E = \{e_{u,v} \| u \in V,\ v \in V\}$	The set of network links	Generated by BRITE
$D(e_{u,v})$	The link distance of link $e_{u,v}$	1 (hop count)
$R = \{r_k \| k \geq 1\}$	The set of streaming rate of multicast source m_k	Default setting in VLC
$g_k = \{m_k, H_k\}$	The multicast group, H_k, from the multicast source m_k	All H_k=H
D_{th}	The distance threshold	The maximum distance between any two multicast sources



(a) Various number of switches (100 multicast sources and 500 hosts)

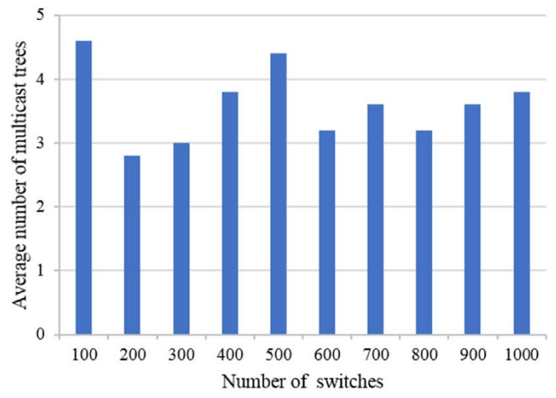


(b) Various number of multicast sources (1000 switches and 500 hosts)

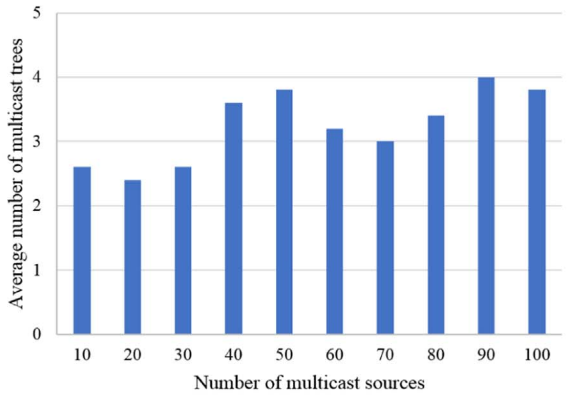


(c) Various number of hosts (100 multicast sources and 1000 switches)

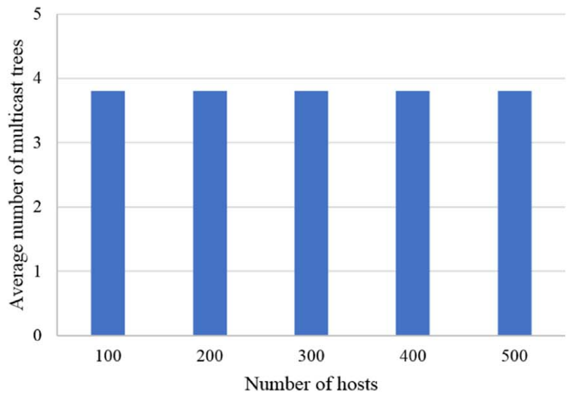
Fig. 6. Average computation time of the controller.



(a) Various number of switches (100 multicast sources and 500 hosts)

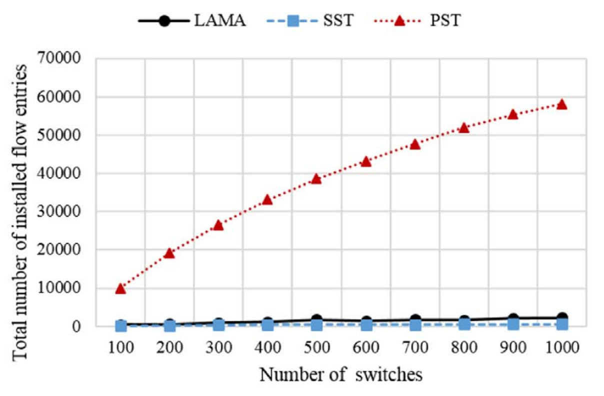


(b) Various number of multicast sources (1000 switches and 500 hosts)

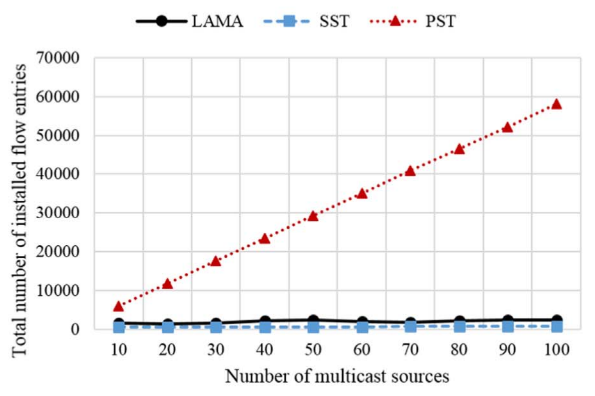


(c) Various number of hosts (100 multicast sources and 1000 switches)

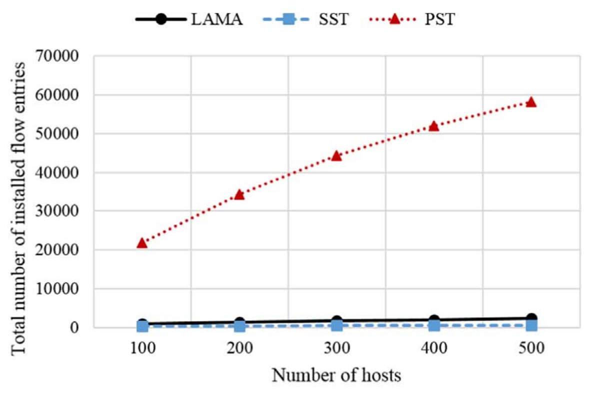
Fig. 7. Average number of multicast trees.



(a) Various number of switches (100 multicast sources and 500 hosts)



(b) Various number of multicast sources (1000 switches and 500 hosts)



(c) Various number of hosts (100 multicast sources and 1000 switches)

Fig. 8. Total number of installed flow entries.

sources, and (c) hosts. In Fig. 6, with the number of switches, multicast sources, and hosts increases, the average computation time of the PST approach increases significantly. In cases of 1000 switches, 100 multicast sources, and 500 hosts, the average computation time of the PST approach is larger than 900 ms. Therefore, when IPTV applications employ the PST approach, the controller would become a performance bottleneck. On the contrary, the average computation time of the SST approach is less than 30 ms in all cases since the controller only needs to maintain a single multicast shared tree for all the multicast groups. However, the SST approach is impractical because the QoS requirements cannot be satisfied. The average computation time of LAMA is around 70 ms regardless of the number of switches, multicast sources, and hosts. It is because LAMA can dynamically group a large number of multicast groups into few multicast clusters and effectively build a multi-group shared tree for each multicast cluster. As shown in Fig. 7, LAMA only yields 2–5 multicast shared trees. Thus, these results show our approach has good scalability in control-plane and is a practical approach.

5.2.2. Scalability of the data-plane

Next, we investigate the scalability of the proposed approach in the data-plane. Fig. 8 shows the total number of installed flow entries with various number of (a) switches, (b) multicast sources, and (c) hosts. In Fig. 8, we can observe the total number of installed flow entries of the PST approach increase significantly as the number of switches, multicast sources, and hosts increases. In cases of 1000 switches, 100 multicast sources, and 500 hosts, the total number of installed flow entries of flow entries of the PST approach is 57,647. Therefore, when the PST approach is employed, a large TCAM space in the switches would be occupied. Although the SST approach only needs to insert very few flow entries in the related switches, it also make the switches become overloaded and congested. However, the total number of the installed flow entries of LAMA is 2298, which is only 4% of that yielded by the PST approach. It is because LAMA can insert a coarse-grained flow entry into on-tree switches instead of several fine-grained flow entries for each multicast cluster. These results show that our approach also has good scalability in data-plane.

6. Conclusion

In this paper, we proposed a locality-aware multicast approach (LAMA) to enhance the scalability of the controller and the switches in SDN. In the proposed approach, there are three stages: multicast group clustering, RP selection, and multicast tree construction. In the multicast group clustering, a distance-based clustering algorithm is presented to cluster all the multicast sources into few multicast clusters. In the RP selection, a locality-aware selection algorithm is proposed to determine the center switch which has the minimum distance to all multicast sources within a multicast cluster as a proper RP. In the multicast tree construction, a shortest-path multicast tree is constructed from the selected RP to the hosts for each multicast cluster. Also, based on the multi-group shared trees, coarse-grained flow entries are inserted into on-tree switches, greatly reducing the number of required flow entries.

The emulation results show only 2–5 multicast shared trees would suffice. The computation time in the controller using LAMA is only around 70 ms, much less than hundreds ms required for the PST approach. The total number of installed flow entries of LAMA is about 2300, only 4% of that yielded by the PST approach. These results show LAMA has good scalability in both control-plane and data-plane, so it is very suitable for IPTV applications.

Acknowledgment

The authors would like to thank the Ministry of Science and Technology, Taiwan for financially supporting this research under Contract No. MOST 103-2622-E-009-012 and MOST 103-2221-E-009-100-MY.

References

Adams, A., Nicholas, J., Siadak, W., 2005. Protocol independent multicast-dense mode (PIM-DM): Protocol specification (revised), Internet RFC 3793.
Bondan, L., Müller, L.F., Kist, M., 2013. Multiflow: Multicast clean-slate with anticipated route calculation on OpenFlow programmable networks. Journal of Applied Computing Research 2, 68–74.
BRITE. 2001. Available online at: ⟨⟨http://www.cs.bu.edu/brite/⟩⟩.
Cui, W., Qian, C., 2014. Dual-structure Data Center Multicast Using Software Defined Networking. arXiv preprint arXiv:1403, 8065.
Farinacci, D., Liu, C., Deering, S., Estrin, D., Handley, M., Jacobson, V., Wei, L., Sharma, P., Thaler, D., Helmy, A., 1998. Protocol independent multicast-sparse mode (PIM-SM): Protocol specification, Internet RFC 4601.
Fenner, W. C., 1997. Internet group management protocol, version 2. Internet RFC 2236.
Johnson, S.C., 1967. Hierarchical clustering schemes. Psychometrika 32, 241–254.
Lantz, B., Heller, B., McKeown, N., 2010. A network in a laptop: rapid prototyping for software-defined networks. In: Proceedings of the 9th ACM SIGCOMM Workshop on

Hot Topics in Networks.pp. 1-6.

Marcondes, C. A., Santos, T. P., Godoy, A. P., Viel, C. C., Teixeira, C. A., 2012. CastFlow: Clean-slate multicast approach using in-advance path processing in programmable networks. In:Proceedings of the IEEE Symposium Computers and Communications (ISCC), pp. 94-101.

McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., Turner, J., 2008. OpenFlow: enabling innovation in campus networks. ACM SIGCOMM Comput. Commun. Rev. 38, 69–74.

Moy, J., 1994. Multicast extensions to OSPF.Internet RFC 1584.

OpenFlow Switch SpecificationVersion 1.3.2. Open Networking Foundation. 2013. Available online at: ⟨⟨https://www.opennetworking.org⟩⟩.

RYU SDN Framework. 2014. Available online at: ⟨⟨http://osrg.github.io/ryu/⟩⟩.

VLC. 2001. Available online at: ⟨⟨http://www.videolan.org/vlc/⟩⟩.

Waitzman, D., Deering, S., Partridge, C., 1988. Distance Vector Multicast Routing Protocol, Internet RFC 1075.

Zappala, D., Fabbri, A., 2001. An evaluation of shared multicast trees with multiple active cores. In: Proceedings of the International Conference on Networking. pp. 620–629.

Zhao, M., Jia, B., Wu, M., Yu, H., Xu, Y., 2014. Software defined network-enabled multicast for multi-party video conferencing systems. In: Proceedings of the 2014 IEEE International Conference on Communications (ICC). pp. 1729–1735.