Society for Science and Education
United Kingdom

TNC Transactions on Networks and Communications

Volume 7, No. 3
ISSN: 2054 -7420

Scalable Multicast Using MPLS in Software Defined Network

Lie Qian

Dept. of Chemistry, Computer & Physical Science, Southeastern Oklahoma State University, Durant, OK, USA
lqian@se.edu

ABSTRACT

Multicast helps to deliver data to multiple receivers efficiently. One scalability challenge faced by multicast is the per-channel forwarding states being maintained in the network layer, which increases linearly with the number of established multicast channels. MPLS helps to alleviate this problem by removing forwarding states from non-branch routers on the multicast tree and label switch packets in non-branch routers. To reduce the number of forwarding states in branch routers, many solutions were proposed to merge multicast trees/subtrees from different channels. Software Defined Network (SDN) decouples the control plane from the data plane, which enables low cost commodity design in routers and flexible network feature deployments through software implementation in centralized controllers. Equipped with SDN's flexible policy and packet processing action installation, multicast tree/subtree merging becomes more convenient in SDN. This paper proposes a new scalable multicast solution in SDN to further reduce the number of forwarding states in routers. In the new solution, first a 2 level MPLS label switching scheme is used to reduce the extra point to point LSPs needed when multicast trees are merged. Secondly, a new multicast tree construction algorithm is designed to pursue more aggressive subtree matching between channels by taking advantage of per channel packet dropping actions in SDN. Simulation results show that the new solution can achieve 10-20 percent reduction in the number of forwarding entries needed for multicast traffic's forwarding.

Keywords: Software Defined Network, OpenFlow, Multicast, MPLS, Scalability

1 Introduction

Multicast was proposed to deliver data from one or multiple sources to a set of destinations in a network efficiently. A set of destinations and the source(s) that receive/send the same data are identified by a multicast channel (group) [1]. A multicast channel is defined as a collection of packets identified by the same channel ID [2]. In case multiple sources exist, the multicast channel ID is a multicast IP address. For a single source multicast channel, a (S, G) address pair [3,4] is used as channel ID, where S is the source's address and G is a multicast group address. A logical multicast tree is created by the multicast protocol for each channel to connect the source(s) to destinations. Data packets are forwarded along the tree and duplicated in the routers at the branch of the tree. For single source channel, the source is the root and all destinations are leaves on the tree, which is called a source specific multicast tree. For multiple source channels, a Rendezvous Point router is chosen as the root, which collects data from all sources [3,5].

DOI: 10.14738/tnc.73.6561
Publication Date: 21st June 2019
URL: http://dx.doi.org/10.14738/tnc.73.6561

To forward packets of a channel properly along the multicast tree, routers on the multicast tree need to maintain a forwarding states for this channel. Forwarding state is composed of the channel ID and a list of the child routers on the multicast tree. The number of forwarding states increases linearly with the number of passing multicast channels, which causes scalability problems in routers. Multi-protocol label switching (MPLS) [6] is a network technology that promises to offer high speed packet forwarding, QoS, traffic engineering and many other new features to the current best-effort, IP-based Internet. MPLS can coexist with many existing network layer and data link layer protocols to provide scalability in today's networks. Scalable multicast in MPLS network has been proposed to reduce forwarding states in routers. MMT [7] removes forwarding states from non-branch routers. In [8], we proposed to replace forwarding states in branch routers with point to multipoint MPLS label switching when a tree or subtree could be used by multiple channels to reach the same set of destinations and a tree matching algorithm, TMST, was proposed in [8] to detect such multicast tree sharing. In [2], we proposed a multicast tree construction algorithm to further reduce forwarding states in branch routers by performing Partial Matching between SubTrees (PMST), which achieves higher matching rate between channels.

Software Defined Network (SDN) is proposed to decouple the control plane from the data plane [9]. In SDN, data plane is still implemented by the device vendor in hardware while the control plane is realized by software in one or multiple centralized controllers. Software such as network management applications, network operating systems and OpenFlow protocols work together to control how packets are handled in each devices' data plane. Deploying new internet architecture, network management, services or protocols is simplified to software update in the controllers and no change is needed in the large number of traffic forwarding devices. Network services such as QoS, virtualization, security, traffic engineering, Information centered networking, forensic analysis, transferrable network applications, deep packet inspection, cloud data center, dynamic middle box deployment, virtual collaborative working environment, VLAN, etc. are becoming more realistic in SDN [10][11].

With SDN's support, per-channel policies could be easily installed into any router in the network, which allows better, finer control over router's per channel packet processing. First, SDN can help to reduce the number of point to point (P2P) LSPs between branch routers. In previous solutions [2][8], to share point to multipoint (P2MP) MPLS LSPs between channels, extra P2P LSPs need to be setup in non-branch routers so that the proper ingress MPLS label used by branch routers could be pushed into the packets by these new P2P LSPs. Such extra P2P LSPs hurt the scalability in non-branch routers. In this paper, a new 2 level MPLS label switching design is proposed to remove the need for extra P2P LSPs in non-branch routers. Secondly, in both TMST and PMST algorithms [2][8], exact destination set match is required in the tree or subtree to avoid any channel's data being forwarded beyond the destination set G. In SDN, per channel packet dropping policies could be easily installed in any router using existing OpenFlow protocol [12]. In this paper, a new algorithm is proposed to merge subtrees between channels when they share enough common destinations. Such more aggressive sharing criteria enables more subtree sharing between channels and further reduces the number of forwarding states in branch routers. Simulations show that the new solution needs less branch router forwarding entries than both MMT and TMST/PMST solutions. The new solution needs a little bit more non-branch router forwarding entries than MMT but 21.8% less than that of TMST/PMST. Counting all forwarding entries, in both branch router and non-branch routers, the new solution has 16.38% less entries than MMT and 10.51% less than TMST/PMST.

The remainder of this paper is organized as follows. Section 2 presents the background review including network model, MPLS, SDN and existing scalable multicast solutions. In section 3 the new multicast solution, including a new 2 level MPLS label switching scheme and a new tree construction algorithm, is presented. Simulation results are shown and discussed in Section 4 and conclusion is drawn in Section 5.

2 Related Works

In this section, a brief review is given for the network model, MPLS networks, Software Defined Network and related works in scalable multicast.

This paper focuses on the single source multicast model. A multicast tree is constructed for a channel with source as root and destinations as leaves. There are 3 types of routers on the tree 1) Branch Routers - root router and routers having more than one child on the tree, 2) Non-Branch Routers - routers having only one child, and 3) Destination Routers - leaf routers that deliver data to end hosts. Data packets are duplicated in Branch Routers. Given a branch router BR1, its next hop Branch Routers are the branch routers reachable on the tree from BR1 via non-branch routers only.

2.4 MPLS Network

MPLS [6] is a versatile data transport solution that addresses network problems such as scalability, QoS management, and traffic engineering. Ingress routers in MPLS networks compute and insert a 32-bit long MPLS shim header that includes a 20-bit long MPLS label, to each incoming packet. A MPLS enabled router, called Label Switching Router (LSR), maintains a MPLS label switching table, with each entry specifying an ingress MPLS label L_{in}, one or multiple egress MPLS labels L_{out} and egress interface ID(s) I_{out}. When packet P carrying MPLS label L_{in} arrives at LSR R, R searches for an entry with ingress label L_{in} in the MPLS label switching table. For each pair (L_{out}, I_{out}) in the matched entry, R swaps L_{in} in P with L_{out} and forwards packet P to neighbor router through egress interface I_{out}. Such process is called "label switching" of the packet. The specific path through the MPLS network that a packet follows based on its MPLS labels is called Label Switched Path (LSP). Label distribution protocol, such as LDP (Label Distribution Protocol) [13], or RSVP (Resource ReServation Protocol) [14] are used to setup LSPs. In this paper, all the routers in a MPLS network are LSRs.

2.5 Scalable Multicast

Various solutions have been proposed to improve the scalability of multicast through reducing the number of forwarding states in the network layer. Network layer aggregation solutions [15-17] reduce the storage and routing complexity in routers through replacing multiple forwarding states in a router with one forwarding state. Due to the non-hierarchical allocation of multicast IP addresses, existing network layer aggregation solutions yield either insignificant reduction in the number of forwarding states or leaky bandwidth, in which multicast packets are transmitted to links not on the multicast tree.

Smart packet solutions are proposed in [18], in which the information of packet forwarding is stored in packet headers to eliminate multicast forwarding states in routers. Routers forward packets based on information extracted from the packet headers. Smart packet solutions require changes in the format of packet headers, limit the number of branch routers, and cause problems due to excessive packet fragmentation. IP encapsulation solutions [19][20] transport packets from one branch router to its next hop BR through unicast IP encapsulation, in which the original packet is encapsulated in a new packet that has the next hop BR's address as destination address. Forwarding states are removed from non-branch routers. IP encapsulation solutions involve resource intensive extra IP packet en/decapsulation at branch routers.

A framework that explains IP multicast deployment in MPLS environment was proposed by Ooms et al [21], which is a pure MPLS layer multicast routing solution. In this framework, a point to multipoint (P2MP) LSP is used for a multicast tree. One P2MP LSP could be shared by multiple trees only if they have the same tree structure. Therefore, large number of multicast channels with various tree structures consumes too many MPLS labels to establish large number of P2MP LSPs and introduce scalability problem in MPLS layer.

MPLS Multicast Tree (MMT) solution was proposed in [7] for scalable multicast in MPLS networks, in which packets are transported from one branch router to its next hop BRs by point to point (P2P) MPLS label switching. Forwarding states are only stored in branch routers. However, MMT made no effort to reduce the forwarding states in branch routers. We proposed tunnel sharing scheme in [8] to reduce the number of forwarding states and MPLS label consumption by using the same P2MP LSP for multiple channels' forward. Such sharing is possible when the tree structures of different channels are similar enough. TMST solution was also proposed in [8] to merge Total Matched SubTree (TMST)'s LSPs to reduce the forwarding states and MPLS label consumption. In TMST solution, a centralized Network Information Management System (CNIMS) keeps records of all established multicast trees, branch routers and LSPs. When a new channel's request comes in, CNIMS tries to find an existing multicast subtree, whose downstream destination list exactly matches to the new channel's destination list. If such subtree exists, no new tree will be created. Instead, new channel's data will be sent to the matched subtree's root router through a newly established P2P LSP, and further delivered to destinations through the existing subtree's existing LSPs. If no such subtree exists, a new tree will be built. The performance gain in tunnel sharing and TMST is based on the match rate between multicast trees, which is not always realistic in real network. Partially Matched SubTree (PMST) solution was proposed in [2] to improve the performance of TMST. PMST tries to match subset of new channel's destinations to any existing subtree in the network. By doing this, the number of destinations remained in the new channel that need a new tree to deliver data to them could be further reduced.

2.6 Software Defined Network and OpenFlow Protocol

Traditional network devices like switches or routers bound data plane (packet forwarding, dropping and modification) and control plane (QoS, routing, network monitoring, per-flow control) in the same equipment. When any new network service, protocol or architecture need to be deployed, the control plane usually requires significant modifications. Even when the change could be accommodated by reconfiguration, still all devices need to be reconfigured by the network administrator. More often, the change needed is too significant and go beyond what is allowed in reconfiguration. In such cases new devices that support the new control plane need to be purchased and deployed if you are lucky to find them on the market while most vendors are reluctant to implement new service into their products before the service become widely accepted on the market, which is not always the case for newly proposed services, protocols or architectures.

Software Defined Network (SDN) was proposed to decouple the control plane and data plane [9]. In SDN, the data plane remains in network devices like switches/routers and the control plane is moved to one or multiple centralized controllers. The controllers install policies in the network devices to control how network traffic are processed in each switch. Network device vendors can focus on improving the data plane performance and lowering the cost of the devices and don't need worry about the constantly changing control plane [22]. Whenever the control plane needs to be changed, controllers could be reprogrammed (install a new software) to be capable of installing different policies in switches to realize the new services or protocols. With SDN, the network service that can be provided in a network is not limited by what the vendor implemented inside the devices anymore, which encourages and enables new network technology, services, protocols and architecture's design.[23]

In a SDN, one or multiple controllers are deployed to make network management decisions. Network management applications, network operating systems and protocol interfacing the controller and switches are installed in the controllers. Network management applications are responsible for making network control decision. Network operating systems provide API to allow quick and easy network management applications programming and shield the network management application from the heterogeneous network technology. Openflow [12] protocol is installed in both switches and controllers. Openflow defines how the rules from the controller could be installed, updated, and removed in the switches and how these rules can be used in the switches for packet processing.

In an OpenFlow switch [12], there are one or multiple flow tables and one group table. Using OpenFlow protocol, the controller can add, modify and remove entries in the tables inside each switch. Each entry in any table consists of match fields, counters, and a set of instructions to apply to matching packets. Packets may need to be processed by multiple flow tables one by one when they come to a switch. When a packet is processed in a table, if a match is found, the associated instruction of that matched entry will be executed. If no match is found, a special miss entry in that table will determine the fate of the packet. The packet could be forwarded to the controller using Packet-in message, dropped or forwarded to the next flow table in the same switch depending on the policy installed in the miss entry. Controller can use Modify-State message to add flow entries in any table.

For a matched packet, instructions associated with the matched flow entry are executed. Some of the instructions edit the action set associated with the packet. All actions in the action set will be executed after the packet finishes its processing in the last flow table in the same switch. In addition to the actions in the action set, "Apply-Action" instruction could execute action of choice during the table's processing. Instruction "Goto" specify which table is the next to process the packet. Instructions also can send metadata to the next table to assist the packet processing there. Actions are defined in OpenFlow to describe the forwarding, modification of the packet, applying meters to the packet, sending the packet to specific queue for QoS purpose, etc.

3 New Scalable Multicast Solution in Software Defined Network

Software Defined Network (SDN) uses controllers to centralize network management. Controllers have complete picture of the whole network and can install forwarding policy in all routers in real time. In this section, a new scalable multicast solution in SDN is presented. The new solution uses a 2 level MPLS label scheme to share point to point LSPs on non-branch routers between channels and share point to multiple point LSPs on branch routers between channels without extra point to point LSP. The new solution also uses a newly proposed multicast tree construction algorithm to increase LSP sharing between channels to improve scalability.

2.7 Multicast in SDN with 2 Level MPLS Label Switching

SDN has one or multiple controllers responsible for control plane decisions, including multicast channel establishment, multicast membership management, MPLS policy setup, multicast tree construction, and forward states installation. After collecting membership information, the controller computes the multicast tree for that channel in the network. Using the TMST/PMST based multicast tree construction algorithm together with the new algorithm proposed in the next section to discover all branch and non-branch routers.

After identifying all routers on the tree, controller sends messages to all on tree routers to install policies to forward data along the multicast tree. In SDN using OpenFlow protocol [12], each incoming packet will be matched to one or multiple flow tables in a router. Table entries stored in these tables are used to specify traffic processing policy such as where to forward the packet, if the packet should be modified, if the packet should be dropped, etc. MPLS labels, IP addresses, TCP ports, etc. could be used to match flow entry in these tables.

In this new solution, there are 3 kinds of forwarding table entries for multicast traffic, 1) Point to Point MPLS switch entry (P2P Entry), 2) network layer IP entry (IP Entry), and 3) Point to Multiple Point MPLS switch entry (P2MP Entry). P2P entry is used in non-branch routers to deliver traffic from one branch router to another. Such P2P LSPs could be statically or dynamically established between branch routers and can be shared by all channels trying to send between the same pair of branch routers. In SDN, each P2P entry applies following actions: pop ingress MPLS label, push egress MPLS label and forward the packets to the port toward the next hop router.

IP entries are used in branch routers to forward single channel's traffic. Routers match the packet's channel ID against the IP entry. The table entry first duplicates the packet for each next hop branch router. For each next hop branch router BR_{next}, the table entry pushes MPLS label so that the packet could be label switched in a P2P LSP toward BR_{next} (pushes the chosen P2P LSP's first ingress MPLS label). Then the packet is forwarded to the interface toward BR_{next}.

In a branch router where multiple channels try to reach the same set of downstream destinations, IP entries for these channels are merged into one P2MP entry. A special MPLS label is used to identify packets from these channels in this branch router and all downstream branch routers. Let's call such special MPLS label Aggregation Label (AG Label). In a branch router where a channel C_1 starts to share another channel C_2's sub multicast tree, the Aggregation Label used on the C_2's shared subtree will be pushed into the C_1's packets. C_1's packets then will use P2MP entries for branch router routing in the rest of the shared subtree. In a branch router on the subtree, the table entry that matches Aggregation MPLS label specifies following actions: duplicate the packet for each next hop branch, push P2P LPS ingress label in each duplicate, forward all duplicates to corresponding egress interface.

This 2 level MPLS label switching design, Aggregation Label and P2P Label, helps to reduce the number of P2P entries in non-branch routers. [2][8] use 1 level P2MP label switch in branch routers where the P2MP ingress labels need to be pushed into packets by the router at the end of each P2P LSP so that the label could be identified in the branch routers. Because each P2P LSP can only specify one last hop egress MPLS label therefore it can only be used to forward traffic for one tree/subtree. Such extra P2P LSPs cause scalability problem in non-branch routers. In the new 2 level MPLS label switching design, the last router on any P2P LSP doesn't need to push any MPLS label into a packet, because the packets will be routed based on 2nd level aggregation label or IP channel ID in branch routers. Such design totally relieves routers in P2P LSPs from being aware of their downstream subtree structure. Only one P2P LSP is needed between any pair of branch routers and it can be used by all channels that need deliver traffic between this pair branch routers.

To establish a P2P LSP in a router, say the packet from ingress interface I_{in} with label L_{in} will be switched to egress interface I_{out} with label L_{out}. One entry is installed in the first flow table of the router using OpenFlow. The entry's matching field uses I_{in} and L_{in} to identify packets with L_{in} coming from interface I_{in} while set value ANY to any other parts of the matching field. A "Write Action" instruction is in the Instructions part of the flow entry. The "Write Action" instruction writes following actions into this packet's action set 1) pop MPLS label, 2) push MPLS label L_{out}, 3) output on interface I_{out}. After the flow table processing, the packet will leave the table processing and has all actions in the action set executed, where the MPLS label L_{in} will be replaced by L_{out} and then the packet will be forwarded to interface I_{out}.

To establish a P2MP entry (IP or aggregation label based) in a router, say the packet from ingress interface I_{in} with aggregation label L_{ag} will be switched to egress interface I_{in_1} with label L_{out_1}, to I_{in_2} with L_{out_2}... to I_{in_n} with L_{out_n}. One entry is installed in the first flow table of the router. The entry's matching field use I_{in} and L_{ag} to identify packets with L_{ag} label coming from interface I_{in} while set value ANY to any other parts of the matching field (or use channel ID instead of L_{ag} if it is an IP entry). An "Apply-Actions" instruction is in the Instructions part of the flow entry. The "Apply-Actions" instruction specifies a list of 3n actions, 3 actions for each branch. For I_{out_i} and L_{out_i}, the 3 actions are 1) pop MPLS label, 2) push MPLS label L_{out_i}, 3) output on interface I_{out_i}. Each output action in the apply-action instruction makes a clone of the packet (with new MPLS label just pushed in) and forwarded to the corresponding output port. After all actions being executed, the packet will leave the flow table processing. Because there is no output action in its action set, the packet will be dropped.

2.8 OLST (Overlap SubTree) Algorithm

Base on the discussion in subsection 3.1, it is obvious that the multicast scalability performance in SDN depends on the extent of MPLS LSP sharing between multicast channels. Such sharing could be achieved when trees or subtrees from different channels are identical. Both TMST and PMST algorithms [2][8] try to construct new multicast trees by reusing existing subtrees.

However, TMST and PMST only reuse the existing subtrees whose destination set is a subset of the new channel's destination set. Existing subtrees that reach any destination beyond the new channel's designation list cannot be used on the new channel's tree construction. This design makes sense on traditional network. Because when an existing subtree is used to forward data of a new channel, all destinations reachable from that subtree will receive data from this new channel. If any destination is not in the destination list of the new channel, the resource such as bandwidth and forwarding state used to forward the data to these undesired destinations are wasted.

In SDN network, per channel packet processing (such as dropping) policy could be easily installed. Such flexibility allows the tree construction algorithm to reuse the existing subtree even if the tree reaches some non-destination routers. The resource waste could be limited to minimum by installing dropping policies in routers to drop the packets of this new channel to prevent them from being forwarded toward undesired destinations. To implement such drop policy in a router, a new table entry is created in the flow table to match the multicast channel's id (source address and port number). The entry has its priority field set to be greater than the P2MP entry or P2P entry. The priority field setting is necessary to guarantee that the packets from this channel be matched to this new entry but not the MPLS label switch entry. Action "Drop" is specified in this new entry to drop packets from this channel. Now the packets carrying the same ingress MPLS label could be matched to one of the two flow entries. If the packets belong to the new channel, they will match the channel ID entry with high priority and be dropped. If the packets belong to any other channel, they will match the MPLS label entry and be label switched to next hop router(s).

Equipped with the capability of dropping specific channel's packets on a shared subtree, I am proposing a new **OverLap SubTree** (**OLST**) algorithm in Software Defined Network to improve the multicast tree sharing between channels. OLST reuses existing subtrees whose downstream destination set overlaps significantly with new channels destination set. First let's define **overlap factor** (**OF**) for an existing subtree and a new channel as the ratio between the number of destinations on the subtree that are not in the new channel's destination set and the number of new channel's destinations that reachable from the subtree. In PTMST, only the existing subtrees with OF = 0 can be used to build the tree for a new channel. Now, in SDN, the acceptable OF value could be set by the SDN controller based on historical multicast tree statistics. In next section, simulation results are used to demonstrate how the optimal OF value could be estimated.

SDN controller maintains a list of existing subtrees in the network for existing multicast channels. Only subtrees rooted at branch routers are included in the list. A subtree rooted at branch router $BR1$ for channel D is denoted as $T\_sub(D, BR1)$. It records the reachable destinations on this subtree, denoted as $Dest(D, BR1)$. The list of subtrees is sorted based on the number of reachable destinations in descending order.

Assume C is a new arriving channel with source S. OLST algorithm builds a multicast tree using following steps:

Step 1. Compute a temporary multicast tree $T_{temp}(C, S)$ rooted at source S using any existing multicast tree algorithm to connect all destinations of channel C (e.g. shortest path tree, Steiner tree)

Step 2. For each branch router *BR* in $T_{temp}(C, S)$:

a. Compare set $Dest(C, BR)$ using every existing subtree record maintained by the controller. An existing subtree $T\_sub(a, B)$ with $Dest(a, B)$ is deemed as a match when 1) $Dest(C, BR) \subseteq Dest(a, B)$ and 2) $\frac{|Dest(a,B) - Dest(c,BR)|}{|Dest(a,B) \cap Dest(c,BR)|} \leq OF$.

b. For a matched $T\_sub(a, B)$, remove $Dest(C, BR)$ from L_C, add B into L_C if B was not there. Install drop policies for channel C to prevent packets reach any destination in set $Dest(a, B) - Dest(C, BR)$.

Step 3. If L_C is not empty, for every existing subtree $T\_sub(a, B)$ from controller's subtree list

a. Compare $Dest(a, B)$ against L_C, $T\_sub(a, B)$ is deemed as a match when $\frac{|Dest(a,B) - L_C|}{|Dest(a,B) \cap L_C|} \leq OF$

b. If $T\_sub(a, B)$ is a match, remove $Dest(a, B) \cap L_c$ from L_C and add B into L_C if B was not there. Install drop policies for channel C to prevent packets reach any destination in set $Dest(a, B) - Dest(c, BR)$.

Step 4. Build a final tree T_{final} to connect all members in L_C with source S. Install a new IP entry in every branch router on T_{final}. On any leaf node on T_{final}, if it is not the destination of channel C, add push action in the upstream branch router's IP entry to push Aggregation Label into channel C's packets as discussed in section 3.1.

Packets of channel C are first label switched on the new tree T_{final} using the IP entries in branch routers. When the packets reach the leaf routers on T_{final}, either they reach the destinations, or they will be switched on existing subtrees with an Aggregation MPLS Label, which is pushed into the packets in the last branch router before the leaf router. Drop policies for channel C are installed in existing subtrees to prevent them from reaching destinations not in channel c. The cost added to the network from new channel C includes the new IP entries on T_{final} and all the IP entries with drop actions. If the OF is set too large, the cost from the drop policy entries will surpass the save from sharing P2MP entries. In simulation section, such tradeoff will be further studied.

Figure 1 depicts an example of this new solution. There are 2 channels. $C1$ has source $S1$ and $C2$ has source $S2$. Channel $C1$'s destination set is $\{d1, d2, d3, d4, d5, d6\}$. Before $C2$'s existence, $C1$'s traffic is identified in branch routers $BR1$, $BR2$, $BR3$ and $BR4$ using IP entry as Figure 2 shows. 10 P2P LSPs are used to deliver packets between branch routers. Branch routers need duplicate the packets, push P2P LSP's labels and forward the packets toward each branch.

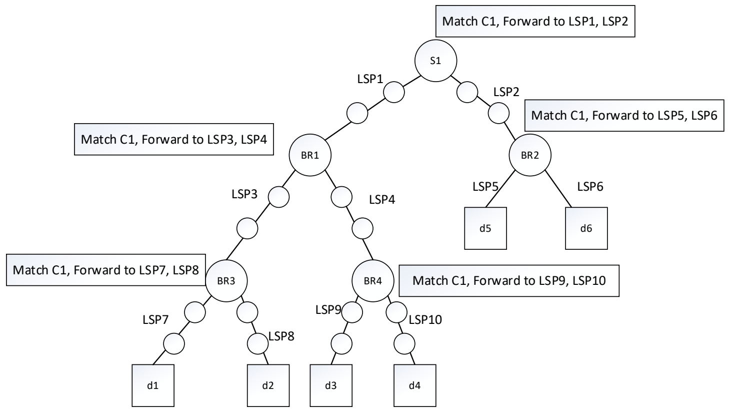
Match C1, Forward to LSP1, LSP2
S1
LSP1
LSP2
Match C1, Forward to LSP5, LSP6
Match C1, Forward to LSP3, LSP4
BR1
BR2
LSP3
LSP4
LSP5
LSP6
d5
d6
Match C1, Forward to LSP7, LSP8
BR3
BR4
Match C1, Forward to LSP9, LSP10
LSP7
LSP8
LSP9
LSP10
d1
d2
d3
d4

Figure 1: Single Channel without Tree Sharing

When channel $C2$ with destination set $\{d2, d3, d4\}$ needs to be established, through running OLST algorithm, SDN controller decides to reuse the subtree including $BR1$, $BR3$ and $BR4$. As shown in Figure 2, the IP entries in $BR1$, $BR2$ and $BR4$ are changed to P2MP entries with Aggregation Label L_{ag}. In $BR1$'s previous hop branch routers ($S2$ for $C2$, $S1$ for $C1$), L_{ag} label is pushed into the packets of $C1$ and $C2$ toward $BR1$ so that they can be matched with L_{ag} on the subtree. Therefore, no new entry is added into branch routers for $C2$ in the subtree rooted at $BR1$. Because $d1$ is not a destination of $C2$, an extra drop entry that match $C2$'s channel ID needs to be installed in the first router after $BR3$ toward $d1$ to drop the traffic of $C2$ but let $C1$'s traffic pass.

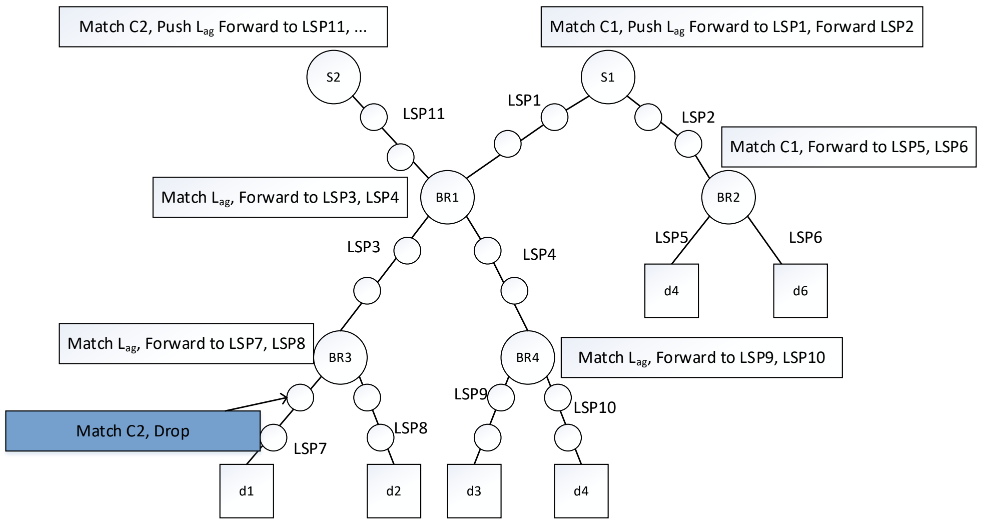
Match C2, Push L_{ag} Forward to LSP11, ...
Match C1, Push L_{ag} Forward to LSP1, Forward LSP2
S2
S1
LSP11
LSP1
LSP2
Match C1, Forward to LSP5, LSP6
Match L_{ag}, Forward to LSP3, LSP4
BR1
BR2
LSP3
LSP4
LSP5
LSP6
d4
d6
Match L_{ag}, Forward to LSP7, LSP8
BR3
BR4
Match L_{ag}, Forward to LSP9, LSP10
LSP9
LSP10
Match C2, Drop
LSP8
LSP7
d1
d2
d3
d4

Figure 2: Two Channels with Overlapped Subtree Sharing and 2 Level Label Switching

4 Simulation Results

Simulations are used to evaluate the new multicast solution with OLST algorithm. In the simulations, topology is randomly generated with 4000 routers and 2000 multicast channels, each of which has one source and a maximum of 1200 destinations (the actual number of destinations for each channel is randomly selected between 500 and 1000). Shortest path multicast tree algorithm is used for tree construction. For the sake of controller's processing burden, SDN controller only keeps records of any subtree whose destination size is greater than 10. The number of flow entries in router's tables is measured to evaluate the scalability.

Three solutions' are simulated and measured, 1) MMT solution [7] (label switch between branch routers, channel id forward state in branch routers without subtree reuse), 2) 1 level MPLS label solution with TMST/PMST tree construction algorithms [2] (shared P2MP LSPs with TMST/PMST algorithm), and 3) 2 level MPLS label solution with OLST tree construction algorithm using different overlap factors.

The results are plotted in Figure 3. MMT solution needs 588782 flow table entries in all routers to support all 2000 multicast channels, including 535226 point to multipole point IP entries in branch routers and 53556 P2P MPLS label entries in non-branch routers. TMST/PMST solution installs 550187 table entries in all routers. Among these entries, there are 458245 point to multipoint entries in branch routers, which is fewer than MMT because of multicast tree sharing. However, as described in section 3.1, extra P2P LSPs are needed in 1 level label switching solutions to enable multicast tree sharing. In the TMST/PMST simulation, 91942 P2P entries are installed in non-branch routers, which is about 71% more than that of MMT. When the overlap factor is set to about 0.1-0.125, the newly proposed solution with 2 level MPLS label switching and OLST algorithm only installs 400798-403361 point to multiple points entries in branch routers (including both IP and Aggregation Label matched entries), 71591-71899 P2P entries in non-branch routers and 17576-19669 dropping entries. The new solution reaches the best performance compared to MMT and TMST/PMST solutions. The overall table entry number is 16.38% less than that of MMT solution and is 10.51% less than that of the solution using 1 level label switching and TMST/PMST algorithm.

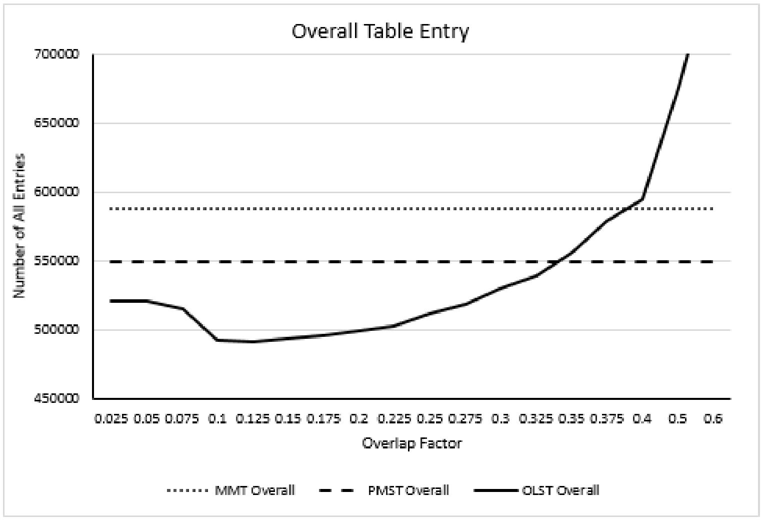

Figure 3: Overall Table Entry Number Comparison

If the P2P LSPs between branch routers are statically setup and therefore have no effect on the scalability, the scalability is only affected by the number of dynamically created table entries (P2MP entries and IP entries for both forwarding and dropping). Figure 4 shows that the new solution's dynamic table entry number is 21.44% less than that of MMT and 8.24% less than that of TMST/PMST algorithm.

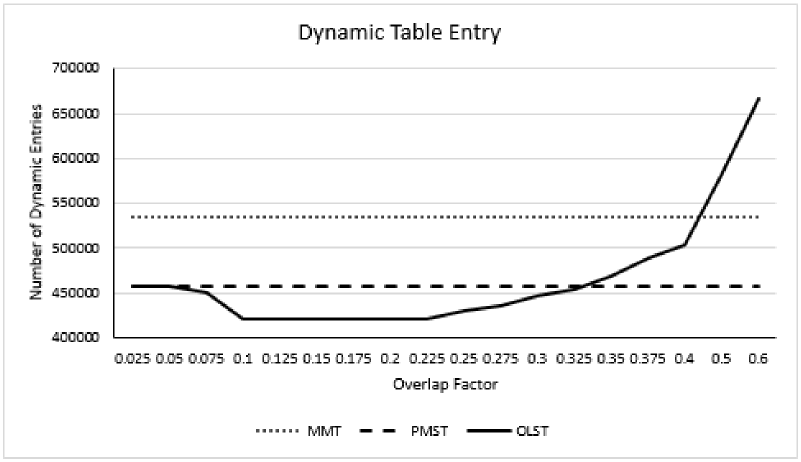

Figure 3: Dynamic Table Entry Number Comparison

The overlap factor plays an important role in the performance of the new solution. When the overlap factor becomes larger, new subtrees will be matched to existing subtrees with more unnecessary destinations, which requires more per channel drop policies to be installed on routers. The results in Figure 3 and 4 show that after overlap factor reaches 0.1, the new solution's performance starts getting worse. Overall number of entries needed become worse than PMST solution after overlap value >0.35, and worse than MMT after overlap value >0.4.

5 Conclusion

Reducing the number of forwarding entries in branch routers and non-branch routers help to improve the scalability of multicasting. MPLS network allows traffic from different channel share the same point to point or point to multipoint label switching entries in a network. Software Defined Network enables flexible per flow policy installation and MPLS label action in routers. In this paper, a 2 level MPLS label switching design and a new multicast tree construction algorithm are proposed to encourage more forwarding entry sharing in SDN MPLS network to improve multicast scalability. Simulations show that the new solution can reduce 10-20 percent table entries in the SDN network for multicast traffic forwarding.

REFERENCES

[1]. S. Deering, "Host extensions for IP multicasting," *RFC 1112,* August 1989.

[2]. Qian, L., Liu, X., & Wang, Y. (2010). A New Tree Construction Algorithm for Scalable Multicast in MPLS Networks. Accepted in Proceedings of International Symposium on Computer Network and Multimedia Technology, 2010, CNMT.

[3]. S. Bhattacharyya, "An overview of source-specific multicast (SSM)," RFC 3569, July 2003.

[4]. H. Holbrook, and B. Cain, "Source-specific multicast for IP," Internet draft, draft-ietf-ssm-arch-04.txt, Oct. 2003.

[5]. D. Estrin, D. Farinacci, A. Helmy, D. Thaler, S. Deering, M. Handley, V. Jacobson, C. Liu, P. Sharma, and L. Wei, "Protocol independent multicast-sparse mode (PIN-SM): protocol specification," RFC 2362, June 1998.

[6]. E. Rosen et al., "Multipotocol label switching architecture," RFC 3031, January 2001.

[7]. A. Boudani, B. Cousin, and J. –M. Bonnin, "An effective solution for multicast scalability: the MPLS multicast tree (MMT)," Internet draft, draft-boudani-mpls-multicast-tree-06.txt, October 2004.

[8]. Lie Qian, Yiyan Tang, Yuke Wang, Bashar Bou-Diab, and Wladek Olesinski, "A New Scalable Multicast Solution in MPLS Networks," IEEE GLOBECOM 2006, San Francisco, November 2006.

[9]. H. Yin et al., SDNi: A Message Exchange Protocol for Software Defined Networks (SDNS) across Multiple Domains, Jun. 2012, Internet draft. [Online]. Available: http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-481360.pdf

[10]. W. Xia, Y. Wen, C. Foh, D. Niyato and H. Xie, A Survey on Software-Defined Networking, IEEE Communication Surveys & Tutorials, Vol. 17, no. 1, pp. 27-51, 1st Quarter 2015

[11]. F. Hu, Q. Hao and K. Bao, A Survey on Software-Defined Network and OpenFlow: From Concept to Implementation, IEEE Communication Surveys & Tutorials, Vol. 16, no. 4, pp. 2181-2206, 4th Quarter 2014

[12]. OpenFlow Switch Specification, version 1.5.1, Open Networking Foundation, March 26, 2015, [Online]. Available: https://www.opennetworking.org/wp-content/uploads/2014/10/openflow-switch-v1.5.1.pdf

[13]. L. Andersson, P. Doolan, N. Feldman, A. Fredette, and B. Thomas, "Label Distribution Protocol Specification," RFC 3036, January 2001.

[14]. D. Awdeche, L. Berger, D. Gan, T. Li, V. Srinivasan, and G. Swallow, "RSVP-TE: Extensions to RSVP for LSP tunnels," RFC3209, December 2001.

[15]. Jun-Hong Cui, Li Lao, Dario Maggiorini, Mario Gerla, “BEAM: A distributed aggregated multicast protocol using bi-directional trees,” IEEE ICC 2003, vol. 1, pp. 689 – 695, 11-15 May 2003.

[16]. A. Fei, J. H. Cui, M. Gerla, and M. Faloutsos, “Aggregated multicast: an approach to reduce multicast state,” Proc. Of Sixth Global Internet Symposium (GI2001), November 2001.

[17]. Jun-Hong Cui, Jinkyu Kim, A. Fei, M. Faloutsos, and M. Gerla, “Scalable QoS multicast provisioning in Diff-Serv-supported MPLS networks,” IEEE GLOBECOM 2002, vol. 2, pp. 1450 – 1454, 17-21 November 2002.

[18]. A. Striegel, and G. Manimaran, “A scalable approach for DiffServ multicasting,” IEEE ICC 2001, vol. 8, pp. 2327-2331, 11-14 June 2001.

[19]. A. Boudani, and B. Cousin, “Simple explicit multicast (SEM),” Internet draft, draft-boudani-simple-xcast-04.txt, March 2004.

[20]. Jining Tian, and Gerald Neufeld, “Forwarding state reduction for sparse mode multicast Communication,” IEEE INFOCOM 1998, March 1998.

[21]. D. Ooms et al., “Framework for IP multicast in MPLS”, RFC 3353, August 2002.

[22]. S.H. Yeganeh, A. Tootoonchian, and Y. Ganjali. *On scalability of software-defined networking*. IEEE Commun. Mag., 51(2):136–141, February 2013.

[23]. E. Kissel, G. Fernandes, M. Jaffee, M. Swany, and M. Zhang, *Driving software defined networks with xsp*. In SDN12: Workshop on Software Defined Networks, 2012.