Softwarized IP Multicast in the Cloud

Shouxi Luo, Huanlai Xing, Pingzhi Fan

***Abstract*—In this article, we revisit the problem of *how to provide "native" IP multicast support to enterprise distributed applications in today's clouds, without touching neither the application implementation nor underlying network hardware*. We propose SDM, Software-Defined IP Multicast, to explore the idea of performing *Multicast-to-Unicast (M2U)* and *Unicast-to-Multicast (U2M)* translations at the virtualized network edge. At the data plane, with novel architecture designs, SDM achieves efficient M2U and U2M translations by only using Open vSwitch (OVS), an OpenFlow-compatible open-source software switch widely deployed in modern clouds. At the control plane, SDM dynamically adopts multicast trees respecting the join and leave of receivers with flexible cost-based algorithms. SDM is very easy to deploy and use, as its required features are already supported by off-the-shelf OVS software (version $\geq$ 2.3) and OpenFlow protocol (version $\geq$ 1.1). Performance evaluations of its control-plane algorithms also imply that SDM is flexible to construct throughput and latency/height optimized multicast trees respecting task requirements.**

***Index Terms*—IP multicast, software-defined networking, cloud**

I. Introduction

Nowadays, an increasing number of enterprises are migrating their applications deployed in private self-maintained data centers to public clouds for cost savings. Off-the-shelf advanced virtualization techniques like *Virtual Machine* (VM) and *Linux container* have made the migration of the computation-related parts easy. However, the migration of the involved network-related parts is quite challenging. For instance, many distributed applications have employed native IP multicast to implement the *point-to-multipoint* data deliveries they widely involve; the lack of the support of native multicast among today's cloud networks makes such migrations quite hard [1]. Despite a lot of effort has been focused on designing IP- or overlay- based solution to support multicast in large-scale data center networks, they cannot help, as they i) require new switch hardware, ii) introduce incompatible network programming APIs thus non-trivial application modification is required, or iii) both [1–5]. Accordingly, we ponder a fundamental question: *Is it possible to provide "native" IP multicast support to distributed cloud applications without modifying neither the application implementation nor the underlying cloud network?*

In this article, we provide a cautiously optimistic answer via the case design of SDM (Software-Defined IP Multicast). Motivated by the observation that applications in public clouds are generally packaged in VMs or containers which interconnect to the external cloud network via software switches like Open vSwitch (OVS) [6], SDM provides "native" IP multicast supports to these virtualized distributed applications

The authors are with Southwest Jiaotong University, China (Corresponding author: Shouxi Luo).

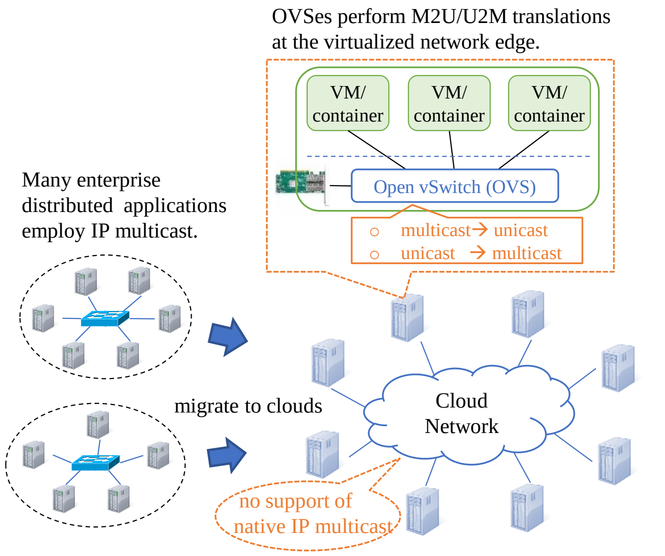

OVSes perform M2U/U2M translations at the virtualized network edge.
VM/ container
VM/ container
VM/ container
Open vSwitch (OVS)
multicast→ unicast
unicast → multicast
Many enterprise distributed applications employ IP multicast.
migrate to clouds
Cloud Network
no support of native IP multicast

Fig. 1: The motivation and insight behind softwarized IP multicast in the cloud.

by employing their OVSes to perform *Multicast-to-Unicast (M2U)* then *Unicast-to-Multicast (U2M)* translations at the virtualized network edge, as Figure 1 shows. Essentially, the core idea of SDM is to establish a tree-like overlay for each multicast task upon the involved OVSes through M2U and U2M translations, such that multicast traffic could be correctly delivered using the already existing unicast forwarding rules.

Indeed, we are not the first to employ M2U and U2M translations for IP multicast. For instance, many data center networks only support a limited number of active IP multicast groups; in case the available IP multicast addresses run out, MCMD employs the design of mapping subsequent multicast operations to unicasts [7]. To further balance the load, DuSM suggests only conduct translations on mice flows [5]. Compared with them, SDM involves two types of novelties, making it excellent to provide native IP multicast services in large-scale multi-tenant public clouds. To sum up, the main contributions, as well as the novelties of SDM, are two-fold:

- **Architecture** (§III): Different from [5, 7], which conduct both M2U and U2M translations through either a customized network stack [7] or a hypervisor [5], SDM directly implements the translations upon existing OVSes with novel data plane architecture designs. As OVS is open-source, production-quality, and widely-employed, SDM is much easier to deploy and use.
- **Algorithm** (§IV): Given a task, there are multiple choices for the construction of its multicast overlay/tree, yielding different throughputs and latencies. To provide customizable multicast services to applications, when receivers

join and leave, SDM employs pTree, a suite of novel algorithms, to optimize the multicast overlay/tree with respect to application-specified requirements. In contrast, trees conducted by previous algorithms like [5, 7, 8] are inflexible thus far from optimal, as the heterogeneity among the application requirements is overlooked.

We prototype SDM upon Mininet [9], confirming that it is readily-deployable. Also, we develop a simulator based on Python 3 to analyze the performance of SDM's control-plane algorithms. We find that i) all its required features are already supported by OVS (version $\geq$ 2.3), and ii) SDM can implement the multicast overlay/tree by installing a few standard OpenFlow rules (version $\geq$ 1.1) on involved OVS switches [6, 10]; moreover, iii) numerical results indicate that SDM is flexible to achieve optimized multicast trees for dynamic *point-to-multipoint* communication requests.

The rest of the article proceeds as follows. Section II first sketches the design of SDM. Then, Section III and IV present the design detail of the data plane and control plane of SDM, respectively. A preliminary performance evaluation follows in Section V; and finally, Section VI summarizes the article.

II. SDM Overview

Basically, SDM achieves flexible "native" IP multicast without the support of hardware switch by performing M2U and U2M translations at the network edge. For each multicast transfer, SDM duplicates its packets and rewrites the packet headers to translate the multicast transfer into multiple unicast flows at the network ingress (Besides header rewrite, SDM can alternatively employ OpenFlow-supported encapsulations like IP-in-IP to perform the translation in practice.); then, following the existing unicast routes, these "unicast" packets are delivered to their destinations efficiently; finally, at the network egress, SDM rewrites the packet headers back and forwards them to the final receivers. For performance optimization, some network egresses are selected as relaying nodes as well. Thus, besides sending multicast packets to their directly connected receivers, like a network ingress, a selected relay node also duplicates and redirects packets to other receivers via unicast following the multicast tree.

Such a design has many advantages, making SDM readily-deployable, flexible, and very efficient. In summary, SDM is:

- **Easy and ready to deploy**. First of all, in production, multicast endpoints are generally virtual nodes like VMs or containers, rather than physical servers. These virtual nodes connect to the host and cloud network using high-performance software switches like OVS. As §III will show, with novel designs, the data plane needed by SDM can be implemented upon OVS, without touching hardware switches. Hence, SDM is readily-deployable.
- **Efficient at establishing multicast trees**. Second, according to the design (§III), to establish a multicast tree, SDM only needs to install a few standard OpenFlow rules on OVSes located at the involved servers, greatly reducing the number of switch operations, as no internal routing modification is needed anymore. Thus, SDM is much simpler than other solutions [1, 2]. Moreover, as a performance-optimized software [6], configuring the forwarding rules of an OVS is much faster than that of a hardware switch, since no TCAM movement will be triggered [11]. Hence, SDM can establish trees for dynamically-arriving multicast requests very efficiently.
- **Highly-scalable**. Third, as is known, today's hardware switch has a very limited size of forwarding table [11]; with OVS-based M2U translation, SDM consumes no additional rules in the underlying network. Regarding OVS, it can support a huge amount of rules respecting the server's capacity; meanwhile, each OVS only needs to maintain rules for its directly-connected virtual nodes. All these facts make SDM easy to scale up, becoming increasingly attractive for large-scale networks.
- **Expressive**. Fourth, the SDM data plane is built upon OVS, which supports OpenFlow protocols by design. With a logically centralized controller, SDM can easily implement high-level policies (e.g., access controls, QoS, fine-grained measurements, etc.) for multicast transfers.
- **Collaborative**. Last but not least, the M2U-translation-based design also enables SDM future-proof to jointly work with the abundant existing and newly-proposed advanced flow scheduling proposals designed for unicast (e.g., flowlet-based load balance, fast-failover, etc. [12]).

As there is no free lunch, the softwarization and overlay based design of SDM also introduces processing overheads to the involved servers, bandwidth overheads to the data center network, and latency overheads to the multicast transfer. More specifically, for a *point-to-multipoint* transfer whose receivers distributed among n servers, compared with the legacy design of configuring the underlying network to achieve IP multicast, SDM would install about $2n + 1$ to $3n$ forwarding rules at involved OVSes in total to conduct the M2U and U2M translations, respecting the multicast tree's structure. According to the current design of OVS, increased numbers of rules and active flows would impact the forwarding throughput [6, 8, 13]. Meanwhile, about n times of more access uplink bandwidth would be consumed at the edge because of the translation. To limit both the processing and bandwidth overheads, proper optimization is to distribute the operations of M2U translation among involved servers to balance their loads. For example, constructing each multicast tree as a chain would yield the minimum overheads. However, such a design also enlarges the multicast delay, as more hops are involved.

To address these issues, SDM employs a suite of policy-based algorithms named **pTree** for the construction of multicast trees. In practice, data center traffic is fragile and both the multicast requests themselves and their involved receivers are likely to arrive and depart online; thus we argue that letting the network stick to the optimal multicast configuration for a snapshot network state is less attractive; instead, greedy-yet-flexible algorithms are preferred. To be customizable, SDM allows applications to specify the desired latency bounds along with their multicast requests and allows operators to configure the number of M2U translations that each server would conduct through *filters*, such that both the processing and bandwidth overheads could be controlled and balanced.

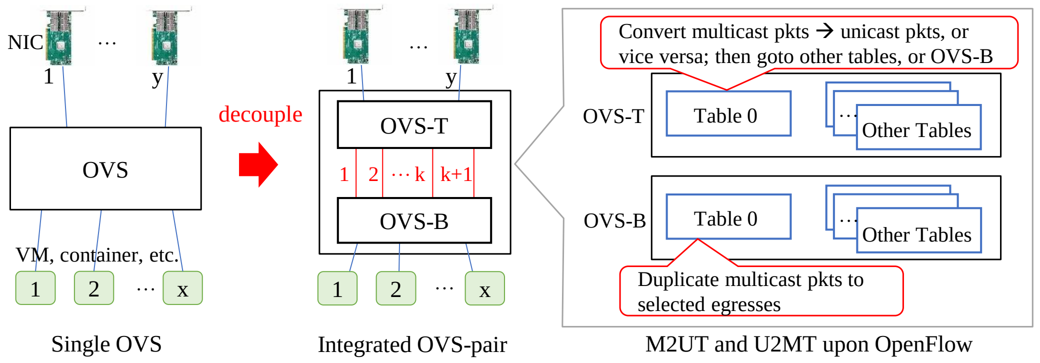

NIC
1
y
OVS
VM, container, etc.
1
2
x
Single OVS
decouple
OVS-T
1 2 ⋯ k k+1
OVS-B
Integrated OVS-pair
Convert multicast pkts → unicast pkts, or vice versa; then goto other tables, or OVS-B
OVS-T
Table 0
Other Tables
OVS-B
Duplicate multicast pkts to selected egresses
M2UT and U2MT upon OpenFlow

Fig. 2: The architecture of SDM's OVS-based data plane.

III. SDM Data Plane

As Figure 2 shows, consider that there are x virtual nodes (e.g., VMs, containers) hosted on a server interconnecting to the data center networks with y network interface cards (NICs). By default, these virtual nodes might be attached to a single OVS; by decoupling the function of this single OVS into an integrated OVS-pair, SDM can implement the needed data plane functions using standard OpenFlow features supported by OVS (version ≥ 2.3). Meanwhile, such a data-plane design is very easy to implement and deploy, since it only requires each server to launch two OVS instances and configure their virtual links along with forwarding rules properly.

In this section, we first show how such an integrated OVS-pair achieves M2U and U2M translations at the network edge (§III-A), then discuss how to repopulate rules residing in the original single OVS to fit the decoupling (§III-B).

A. Integrated OVS-Pair

Inside an integrated OVS-pair, these two OVSes at the top and bottom, interconnected with $k+1$ virtual links as Figure 2 sketches, are named OVS-T and OVS-B, respectively. Here, k is a reconfigurable parameter indicating the maximum fanout of the M2U-translated tree allowed by the SDM data plane. For these $k+1$ internal virtual links labeled $1, 2, \cdots, k+1$, SDM employs the first k links to transmit the duplication of multicast packet, and the last one to transmit other traffic. For each multicast, this integrated OVS-pair performs M2U translations when its packets enter, and then performs U2M translations just before packets leave.

1) Multicast-to-Unicast Translation (M2UT): Basically, for a received multicast data packet, M2UT involves two basic operations: i) duplicate the packet at OVS-B, then ii) rewrite the multicast IP address in each duplication to the right unicast IP address at OVS-T, respectively, both of which are already supported by OpenFlow-compatible switches like OVS (version ≥ 2.3) [6, 10]. In modern OpenFlow-supported switches (OpenFlowVersion ≥ 1.1), there are multiple pipelined flow tables in the format of match-action that packets could go through. To implement the packet duplication at an OVS-pair (say u for instance, whose k-parameter is with the value of K_u), SDM installs a go-to-group OpenFlow rule at the first table of u's OVS-B to match the multicast packet, then multicast the packet to a group of egresses selected from virtual link $1, 2, \cdots, K_u$, through standard OpenFlow features. Suppose that the packet is multicasted to κ ($\kappa \leq K_u$) egresses. On getting these κ duplicated packets, based on their ingress indexes, OVS-T rewrites their destination IP addresses with unicast IP addresses, to redirect them to κ unvisited receivers selected from the multicast task's receiver pool. Notably, these header-rewrite rules reside in the first table of OVS-T; each modified packet continues to go through other tables just like a normal unicast packet. Thus, there is no need for SDM to touch the delivery of these translated packets; they are free to enjoy the abounded network optimizations already deployed (e.g., flowlet-based load balancing, fast-failover, etc. [12]).

2) Unicast-to-Multicast Translation (U2MT): At the receiver side, by looking into the detailed header fields, it is easy to identify which packets are originally converted from multicast packets. Accordingly, SDM installs OpenFlow rules at the first table of the receiver's OVS-T to convert these packets back to their multicast formats and redirect them to OVS-B. On getting a multicast packet, OVS-B multicasts it to the directly-connected receiver nodes, and other virtual links for M2UT if there are still unvisited receivers.

The above-mentioned data plane functions are easy and ready to deploy, as only standard OpenFlow (version ≥ 1.1) rules are employed. We prototype the design upon Mininet [9], confirming the ready-deployability of SDM.

B. Rule Repopulation

Besides the OpenFlow rules installed by SDM, many other forwarding rules are residing in the original single OVS. As SDM has decoupled this OVS into an integrated OVS-pair, the corresponding rules and tables should be repopulated into the integrated OVS-pair as well. For this problem, a promising solution is to i) let the network controller be aware of this augmented topology and ii) restrict that all other traffic except the multicast duplications made by OVS-B only uses the last virtual link, such that services like link discovery and MAC-learning can work properly. Also, as SDM mainly installs OpenFlow rules into the first ingress table, all other rules can just occupy tables starting from the second. Regarding the group rules specified by SDM, they can share the group tables with others.

IV. SDM Control Plane

By configuring all OVS-B switches to redirect Internet Group Management Protocol (IGMP) messages to the controller which reacts following the corresponding RFCs [14], SDM can capture both the join and leave of multicast receivers. By redirecting unknown multicast traffic to the controller, SDM could also precisely know and control which senders could multicast. Triggered by these node dynamic events, SDM can reconfigure all involved OVS-pairs to construct the delivery trees for multicast tasks on demand.

In this section, we look into the detail of how SDM constructs optimized multicast trees respecting node dynamics. We first analyze the design choose adopted by SDM (§IV-A), then describe the network abstraction and problem formulation of SDM (§IV-B), and finally move to **pTree**, the algorithm suite that SDM employs to deal with node dynamic and perform continual tree optimizations (§IV-C).

A. Analysis

To achieve efficient one-to-many delivery, SDM needs to maintain an optimized multicast tree upon the overlay for every multicast task, respecting the join and leave of receivers. Recent studies have shown that data center traffic is fragile and many load balancing techniques such as Equal-Cost Multi-Path (ECMP), Valiant Load-Balancing (VLB), and Conga, have been widely deployed or developed for unicast traffic [15]. In consideration that SDM traffic is translated into unicast at the network edge, we argue that **oblivious routing**, i.e., picking routes for pairs without knowledge of the traffic/demand matrix, is good enough for the construction of SDM trees in practice. Nevertheless, there are still several other constraints that dominate the algorithm design.

First of all, based on the design of the SDM data plane, for a multicast transfer, saying T_i for instance, $\deg_i^+(u)$, the out-degree of node u, is bounded by the OVS-pair configuration K_u. Moreover, the M2UT operations at OVS-pair u would amplify T_i's traffic volume $\deg_i^+(u)$ times, significantly increasing the load on the server's uplinks, especially if each server only employs a single NIC. Suppose that the hosting server of OVS-pair u is with the uplink capacity of b_u and the set of all active multicast transfers is denoted by $\mathbb{T}$. In case that OVS-pair u is the single global bottleneck and flow-based max-min fairness is employed by multicast transfers, each translated unicast would obtain the bandwidth of $b_u/\sum_{T_i\in\mathbb{T}}\deg_i^+(u)$ at most. Hence, on constructing multiple trees for concurrent transfers, SDM should not only limit the out-degree of each multicast tree but also balance their needed M2UT operations among different OVS-pairs/servers for better throughputs. Second, in practice, both the multicast tasks themselves and their involved receivers are likely to arrive and depart online. In this case, letting the network stick to the optimal multicast configuration for a snapshot network state is unattractive; instead, SDM should $i)$ establish multicast routes for new tasks/receivers within a short time, then $ii)$ optimize their multicast trees continually. Third, it is oblivious that the most bandwidth-efficient way for multicast might be maintaining each tree as a chain. However, chains introduce non-trivial multicast latencies to receivers especially those close to the tail. Thus, SDM should try to let the height of the constructed (rooted) tree meet the task's requirement on latency bounds.

B. Formulation

Given a cloud network, SDM abstracts it out as a complete directed graph, G, in which each node denotes an OVS-pair, and $e_{u,v}$, or (u,v), the abstracted link from node u to v, involves a weight $c_{u,v}$ representing the cost of delivery one unit of data from u to v. In this paper, SDM directly employs the (average) hop count of the candidate forwarding paths as the weight. Indeed, the abstracted weight from u to v can be any customizable metrics like the value of $b_{u,v}/d_{u,v}$, where $b_{u,v}$ and $d_{u,v}$ are the available throughput and average hop count from u to v, respectively. Obviously, such an abstraction is powerful, making SDM generic to support various kinds of cloud network architectures including Fat-tree, Leaf-Spine, Xpander, DCell, etc [15]. Accordingly, the problem of constructing optimized out-trees for the i-th multicast task is to find a routed out-tree $T_i \subset G$ with low summarized weight, high throughput, and controlled height, over the abstracted overlay.

Formally, a multicast task (say t_i, the i-th task, for instance) which is to deliver data from root s_i to a group of receivers R_i, can be described by the tuple of $\langle s_i, R_i, \tau_i\rangle$, in which τ_i indicates the upper bound of its desired height (latency) of constructed multicast tree. Let $\mathbb{T}$ be the set of multicast trees already established in the network G, and $\chi[u]$ be the total number of unicast flows translated from multicast at u; the multicast cost of the i-th task's tree T_i in $\mathbb{T}$, which is made up of the level of height-violation, throughput, and summarized weight, can be defined by the triple tuple of

$$\hbar(\chi, T_i) := \left\langle \frac{\max(0, H(T_i) - \tau_i)}{\tau_i}, \max_{u\in V(T_i)} \frac{\chi[u]}{b_u}, \sum_{e\in E(T_i)} \frac{c_e}{\rho_i} \right\rangle \tag{1}$$

Here, $H(T_i)$ is the weighted height of T_i, $\chi[u]$ is the sum of the out-degree of node u among active trees defined by (2), b_u is the uplink capacity of node u's hosting server, c_e is the cost of delivery one unit of data over abstracted link e, and ρ_i is the minimum weight of the corresponding tree for multicast task i provided the constraint of out-degree is released.

$$\chi[u] = \sum_{T_i\in\mathbb{T}: u\in V(T_i)} \deg_i^+(u) \tag{2}$$

C. Algorithms

Now, we present how SDM adopts multicast trees to deal with the join and leave of receivers in detail. We name the scheme as **pTree** (**p**olicy-based overlay multicast **Tree**), which involves two sub-parts, as Table I summarises.

The intuition is that in case the latency/height requirement is satisfied, SDM prefers the multicast tree with larger throughput and less total weight; otherwise, it prefers the one with a lower level of latency (height) violation. To reduce the network load and simplify multicast trees, on constructing and

TABLE I: The schemes that pTree employs for the join and leave of receiver.

Operation		**Description** (consider m multicast tasks each involving no more than n receivers)	**Time complexity**
JOIN		Link the newly joined receiver to the existing node yielding the minimum cost respecting (1).	$\mathcal{O}(n)$
LEAVE	DELTA	Reconstruct **only** the involved tree by using JOIN to process its receivers in their arrival orders.	$\mathcal{O}(n^2)$
	FRESH	Reconstruct **all** multicast trees by using JOIN to process all receivers in their arrival orders.	$\mathcal{O}(mn^2)$

maintaining the tree for a task, we limit SDM to not use extra nodes beyond its source sender and current receivers. Note that, for each OVS-pair, there might be multiple directly-connected receivers and the traffic from the OVS-pair to them is internal. Accordingly, the core of SDM's control plane is to build multicast trees at the level of OVS-pairs.

1) Receiver Join: On receiving the join request of the new receiver node, SDM finds a parent OVS-pair node among the interested tree's candidate set to let it join, such that the involved multicast cost computed by (1) is minimized. By employing a selected set of, instead of all, the available nodes as the candidate parents, SDM is flexible to support user-specified *filters*. For instance, by filtering out nodes whose out-degrees are larger than a predefined threshold, SDM can control the maximum out-degree of nodes precisely. Obviously, the time complexity of appending a new receiver to a tree already involving n receivers with JOIN is $\mathcal{O}(n)$.

2) Receiver Leave: When a receiver leaves, SDM reconstitutes multicast trees correspondingly. As Table I shows, SDM supports two strategies, named DELTA and FRESH, respectively. Basically, DELTA only reconstructs the involved tree: it first initializes the tree to only contain the source node, then repeatedly selects the receiver that would involve the minimum multicast cost from all unprocessed remaining receivers and add this receiver to the tree with JOIN, until all receivers are joined. In contrast, FRESH is more thorough and reconstructs all active trees by using JOIN to process all active receivers in their arrival orders. SDM can make combined use of them: call DELTA to deal with a receiver's leave immediately and then call FRESH periodically. Assume that there are m active multicast trees and the largest one involves n receivers. Accordingly, the time complexity Delta and Fresh involve is about $\mathcal{O}(n^2)$ and $\mathcal{O}(mn^2)$, respectively.

V. PERFORMANCE EVALUATION

In this section, we study the performance of SDM control plane algorithms through numerical simulations. Results show that SDM could construct trees with near-optimal multicast throughput and latency.

A. Methodology

1) Metrics and baselines: For multicast tasks, we mainly care about their throughput and maximum latency, which mainly depend on the number of multicast transfers going through the same link, and the weighted height of their multicast trees, respectively. To highlight the performance of SDM, we mainly consider two baselines, which are supposed to achieve the best multicast throughput and latency in theory:

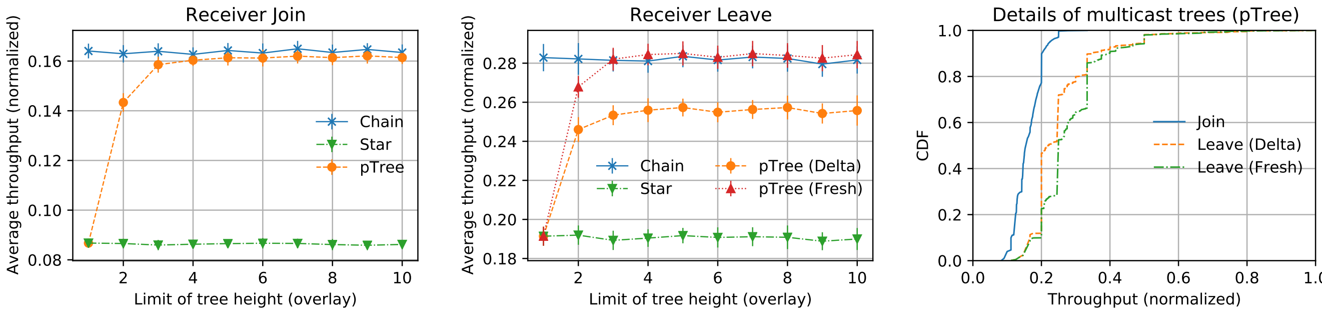
- **Star**: active receivers connect to the sender directly without caring about the limits of K_u; such a design achieves the best multicast latency (This is exactly the scheme employed by [5, 7]).
- **Chain**: active receivers form a chain in their arrival orders without caring about the latency/height requirements and weights; such a design is likely to achieve the best multicast throughput in theory.

More specifically, for each multicast tree, its throughput is determined by the slowest receiver; we assume that all flows in the network share link capacities in the manner of perfect per-flow max-min fairness. As for the multicast latency, we use the involved tree's weighted height as the metric.

2) Network and Workloads: To drive the analysis, we consider that SDM needs to maintain trees for 300 randomly generated multicast tasks in a cloud cluster involving 1000 servers. These servers are interconnected via a flat, non-blocking data center network abstracted as one big switch, in which bandwidth competition appears only at the ingresses or egresses of the servers and all ports have the same normalized *unit* capacity. For each server, we assume that it launches one OVS-pair and the corresponding K_u parameter is large enough to support any multicast trees. Both the source and the receivers we consider here are OVS-pairs (i.e., servers). In the case of receiver join, we assume that the source of each multicast task needs to multicast a long-live stream to other n randomly selected servers; and in the case of receiver leave, after all aforementioned receivers have joined, half randomly selected receivers in the cloud would leave then. During the test, for all multicast trees, we vary τ, the height limit of their multicast trees, from 1 to n, and measure the throughput and latency of each active multicast task when all joins or leaves have been conducted. As the workload is synthesized randomly, we perform 40 trials for each parameter setting. We vary the value of n from several to hundreds and obtain consistent results. The results we show in this article are these when $n = 10$.

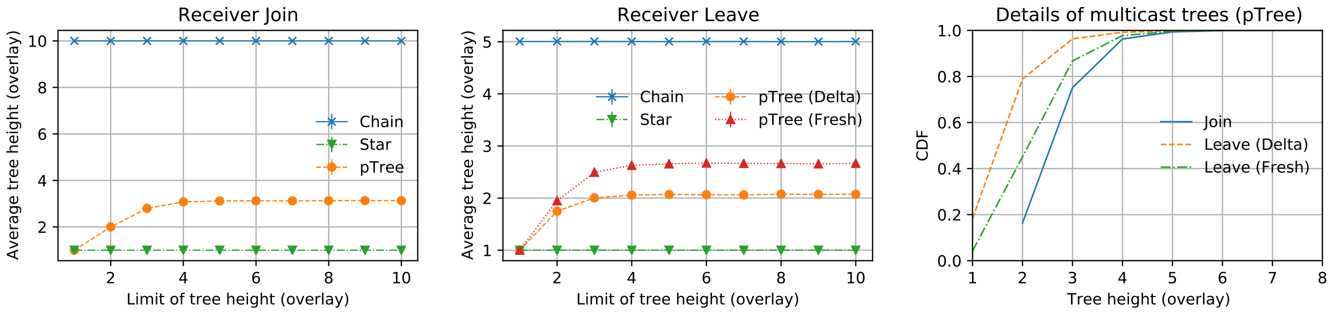
B. Results

Figure 3 shows the details of how the average throughput of active multicast tasks changes with the increase of the limit of tree height (τ). Obviously, as Figure 3a indicates, when the limit of tree height is set to 1, pTree and Star achieve the same performance; however, the multicast throughput SDM's pTree algorithm achieves increases rapidly with the release of τ; and in the test, once the height limit is larger than 3, the average throughput reaches a performance point very close to that of the Chain. Similar results are observed in the case when half of the receivers left as Figure 3b sketches. We find that when dealing with receiver leave, the throughput achieved by pTree (Fresh) is better than that of pTree (Delta); this is reasonable since pTree (Fresh) reconstructs all active multicast



(a) Throughputs when all receivers joined (b) Throughputs when half of the receivers left (c) Without the limit of tree height

Fig. 3: The multicast trees constructed by pTree achieve near-optimal throughput concerning the limit of tree height. Note that, the limits of tree height do not take effect on Star and Chain, according to their designs.



(a) Tree heights when all receivers joined (b) Tree heights when half of the receivers left (c) Without the limit of tree height

Fig. 4: The heights of most multicast trees constructed by pTree are small. Note that, the limits of tree height do not take effect on Star and Chain, according to their designs.

trees, resulting in increased throughput for most active tasks as the throughput distributions in Figure 3c demonstrates. In this test case, $\tau = 10$; i.e., the limit of tree height is totally removed as a multicast tree's height would never exceed the number of its receivers. Since reconstructing all multicast trees would involve a lot of network updates, in practice, pTree (Fresh) can be configured to run periodically.

We also obviate that, in Figure 3b, once the limit of tree height is larger than 3, pTree (Fresh) even outperforms Chain. Note that, after half of the receivers left, each task involves about 5 receivers on average. Then, there are only about $5 * 300/1000 = 1.5$ multicast-translated unicast flows on each egress or ingress, indicating that the performance of pTree-constructed trees outperforms the Chain when the number of concurrent active flows is small. All these phenomena indicate: even with greedy algorithm designs, pTree achieves near-optimal multicast throughput concerning the height limits specified by requests.

Despite the limit of tree height is set to τ, the actual tree height constructed by pTree is far from this limit, especially when τ is large, as Figure 4 shows. Basically, once τ is larger than 3, the multicast tasks' average tree height seems to stay consistent, implying that the multicast latency of trees generated by pTree is well-optimized as well. From Figure 4b, we also notice that when dealing with receiver leaves, the multicast trees conducted by pTree (Fresh) involve higher heights than those constructed by pTree (Delta), indicating that the gain of throughput achieved by pTree (Fresh) over pTree (Delta) shown in Figure 3b, is with the cost of higher tree heights (i.e., larger multicast latency). In the end, Figure 4c shows the distributions of multicast trees' heights constructed by pTree when the limit of τ is removed (i.e., set to 10). Detailed results confirm that SDM achieves optimized tree heights by keeping the height of most trees small.

VI. Conclusions and Discussions

This article presents SDM, a software-defined system that provides virtualized "native" IP multicast support to distributed applications in the cloud by performing *multicast-to-unicast* and *unicast-to-multicast* translations at edge servers. SDM is easy to deploy and use, as its data plane directly builds upon the OVS switches widely existing in data center servers. Also, the performance of SDM is near-optimal since its control plane algorithm pTree constructs throughput and latency/height optimized trees with respect to the requirements of multicast tasks.

Indeed, the high-level idea of the SDM data plane is generic and not bounded to OVS; it is possible to port the design to other OpenFlow- or P4- compatible switches [10, 12]. For instance, if OpenFlow-supported hardware switches are available at the edge, the M2U and U2M translations original conducted by OVS-T can be re-implemented in the corresponding egress and ingress pipelines, respectively. A complete design is beyond the scope of this article and left as future work.

Acknowledgments

We are grateful to the anonymous reviewers for their constructive feedback. The work of Shouxi Luo was supported by the NSFC under Grant 62002300 and the China Postdoctoral Science Foundation under Grant 2019M663552. The work of Pingzhi Fan was supported by the NSFC under Grant 61731017 and the 111 Project under Grant 111-2-14.

References

[1] M. Shahbaz, L. Suresh, J. Rexford, N. Feamster, O. Rottenstreich, and M. Hira, "Elmo: Source routed multicast for public clouds," *IEEE/ACM Transactions on Networking*, vol. 28, no. 6, pp. 2587–2600, 2020.

[2] X. Li and M. J. Freedman, "Scaling ip multicast on datacenter topologies," in *CoNEXT*, 2013, pp. 61–72.

[3] S. Luo, H. Yu, K. Li, and H. Xing, "Efficient file dissemination in data center networks with priority-based adaptive multicast," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 6, pp. 1161–1175, 2020.

[4] W. Jia, "A scalable multicast source routing architecture for data center networks," *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 1, pp. 116–123, January 2014.

[5] W. Cui and C. Qian, "Scalable and load-balanced data center multicast," in *GLOBECOM*, Dec 2015, pp. 1–6.

[6] B. Pfaff, J. Pettit, T. Koponen, E. Jackson, A. Zhou, J. Rajahalme, J. Gross, A. Wang, J. Stringer, P. Shelar, K. Amidon, and M. Casado, "The design and implementation of open vswitch," in *NSDI*, May 2015, pp. 117–130.

[7] Y. Vigfusson, H. Abu-Libdeh, M. Balakrishnan, K. Birman, R. Burgess, G. Chockler, H. Li, and Y. Tock, "Dr. multicast: Rx for data center communication scalability," in *5th EuroSys*, 2010, pp. 349–362.

[8] R. Zhu, D. Niu, B. Li, and Z. Li, "Optimal multicast in virtualized datacenter networks with software switches," in *IEEE INFOCOM*, 2017, pp. 1–9.

[9] N. Handigol, B. Heller, V. Jeyakumar, B. Lantz, and N. McKeown, "Reproducible network experiments using container-based emulation," in *8th CoNEXT*, 2012, pp. 253–264.

[10] Z. Latif, K. Sharif, F. Li, M. M. Karim, S. Biswas, and Y. Wang, "A comprehensive survey of interface protocols for software defined networks," *Journal of Network and Computer Applications*, vol. 156, p. 102563, 2020.

[11] X. Jin, H. H. Liu, R. Gandhi, S. Kandula, R. Mahajan, M. Zhang, J. Rexford, and R. Wattenhofer, "Dynamic scheduling of network updates," in *SIGCOMM*, 2014, pp. 539–550.

[12] O. Michel, R. Bifulco, G. Rétvári, and S. Schmid, "The programmable data plane: Abstractions, architectures, algorithms, and applications," *ACM Comput. Surv.*, vol. 54, no. 4, May 2021.

[13] P. Emmerich, D. Raumer, S. Gallenmüller, F. Wohlfart, and G. Carle, "Throughput and latency of virtual switching with open vswitch: A quantitative analysis," *J. Netw. Syst. Manage.*, vol. 26, no. 2, pp. 314–338, Apr. 2018.

[14] S. Floyd and E. Kohler, "Internet Group Management Protocol, Version 3," RFC 3376, 2002.

[15] M. Besta, M. Schneider, M. Konieczny, K. Cynk, E. Henriksson, S. D. Girolamo, A. Singla, and T. Hoefler, "Fatpaths: Routing in supercomputers and data centers when shortest paths fall short," in *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, 2020, pp. 1–18.

Shouxi Luo is currently a Lecturer with the School of Computing and Artificial Intelligence, Southwest Jiaotong University, China. His research interests include data center networks, software-defined networking, and networked systems.

Huanlai Xing is currently an Associate Professor with the School of Computing and Artificial Intelligence, Southwest Jiaotong University, China. His research interests include mobile-edge computing, network function virtualization, software-defined networking, and evolutionary computation.

Pingzhi Fan is currently a Distinguished Professor and the Director of the Institute of Mobile Communications, Southwest Jiaotong University, China. His research interests include vehicular communications, wireless networks for big data, signal design & coding, etc. He is a Fellow of IEEE, IET, CIE, and CIC.