Scalable and Load-balanced Data Center Multicast

Wenzhi Cui
Department of Computer Science
University of Texas at Austin, TX
wc8348@cs.utexas.edu

Chen Qian
Department of Computer Science
University of Kentucky, KY
qian@cs.uky.edu

***Abstract*—Data center applications use multicast as an effective method to reduce bandwidth cost. However, traditional multicast protocols designed for IP networks are usually bottlenecked by the limited state capacity on switches. In this paper, we propose a scalable multicast solution on fat tree networks based on the observation that data center multicast traffic has strong heterogeneity. We propose to remove the multicast management logic from switches and use the SDN controller to manage multicast groups. The proposed Dual-structure Multicast (DuSM) determines elephant and mice groups according to their traffic amounts and treats them separately. For each elephant group, the controller installs multicast state to maintain multiple shared trees and the group traffic will be balanced evenly among the trees to avoid congestion. For mice groups, the controller applies state-free multicast that trades bandwidth capacity for state capacity, such as multicast-to-unicast translation. Our experiments using real multicast traffic data show that, DuSM can increase the multicast state capacity to support more number of groups by $> 200\%$ compared to IP multicast. DuSM also achieves better traffic balance among links than IP multicast.**

I. INTRODUCTION

Many Data center applications, such as multimedia and publish-subscribe services used in cloud [17], data backup and updates, and network virtualization [22], [23], rely on multicast to disseminate data to multiple receivers [26]. Multicast benefits the network by reducing bandwidth overhead and latency between group members.

Traditional IP multicast protocols cannot scale in Data Center Networks (DCNs) because switches only support limited multicast state in their forwarding tables [13], [17], [19], [20], [26], [28]. End-host multicast [11], also known as overlay multicast or application-layer multicast, does not store multicast state on routers or switches. However using end-host multicast in DCNs will cause significant bandwidth loss due to large amount of duplicated packets transmitted in the network. Furthermore, most multicast protocols do not utilize the topology of DCNs, e.g., multiple parallel paths connecting any pair of end hosts. For example, multicast protocols similar to the Protocol Independent Multicast Sparse Mode (PIM-SM) [14] usually chooses a single core switch on a Fat Tree as the rendezvous point (RP), i.e., the root of the multicast tree. When groups using the same core switch have traffic bursts at the same time, network congestion may occur.

In this paper, we focus on developing a multicast protocol which can scale multicast in DCNs and balance traffic to reduce congestion. We start our design by analyzing real multicast traffic traces in DCNs. Results show that group traffic in DCNs has strong heterogeneity. A small fraction of groups contribute to the majority of traffic, called *elephant groups*, while most groups have very low traffic volume, called *mice groups*. Motivated by such observation, we propose a Dual-Structure Multicast (DuSM) system which relies on the controller in a software defined networking (SDN) platform [2] to manage multicast groups. The SDN controller collects network information and categorize multicast groups into the two types. For an elephant group, DuSM installs multicast rules on switches to maintain multiple shared trees. Traffic of this group will be balanced among these trees to avoid congestion. For mice groups, which are majority, DuSM applies state-free multicast that trades bandwidth capacity for state capacity, such as multicast-to-unicast translation, and hence saves large multicast state space.

The rest of this paper is organized as follows. Section II introduces background knowledge and observations from data analysis. Section III presents the algorithm design of our multicast system DuSM. We evaluate the performance of DuSM in Section IV. We present related work in Section V. Finally we conclude our work in Section VI.

II. BACKGROUND AND DATA ANALYSIS

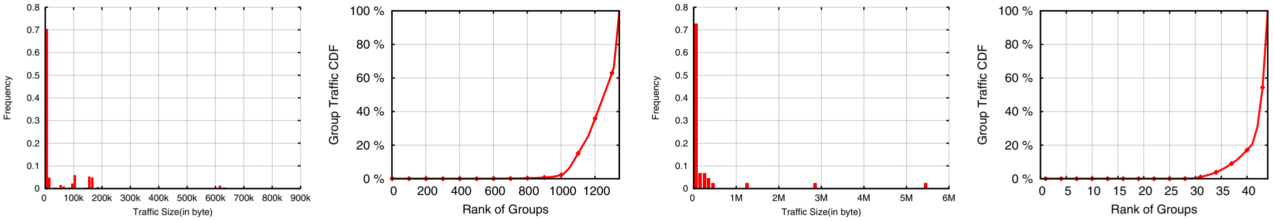
A. Data Center Topologies

Today's data center networks often use *multi-rooted hierarchical tree* topologies (e.g., fat tree [5] and Clos [12]) to exploit multiple parallel paths between any pair of hosts. A standard fat tree topology has three vertical layers: edge, aggregate, and core. A *pod* is a management unit down from the core layer, which consists of a set of interconnected end hosts and a set of edge and aggregate switches that connect these hosts. The current design of DuSM focuses on a fat tree topology. However, we believe our design can be easily extended to other hierarchical topologies.

B. Group Traffic Distribution

We analyze real data center multicast traffic data from two sources, IBM WebSphere Virtual Enterprise (WVE) [1] that is widely used for other multicast design [20], [26] and another data center network of a large telecommunication corporation (we use Telecom as the name of this trace). Due to privacy concerns, all traffics are samples by an unknown rate in each set of data. Hence we only focus on the traffic distribution and pay less attention to the absolute traffic volumes. The WVE trace has 127 hosts in 1364 groups and the Telecom trace has

978-1-4799-5952-5/15/$31.00 ©2015 IEEE



(a) Traffic histogram of WVE (b) Cumulative traffic of WVE (c) Traffic histogram of Telecom (d) Cumulative traffic of Telecom

Fig. 1. Group traffic distribution in real data center networks

512 Virtual Machines (VMs) in 44 groups. For each group, we calculated the total amount of traffic samples in bytes.

Figure 1(a) and Figure 1(c) show the traffic distribution of the two multicast traces in frequency histograms. We find similar observations from the two sets of data. Although the maximum traffic size in both traces reach a large value, around 70% groups only have extremely small multicast traffic. Figure 1(b) and Figure 1(d) show that the sum of traffic of these groups account to less than 10% of the total traffic. In addition, a small proportion of groups contribute to the majority of traffic. We find strong heterogeneity of multicast traffic in both DCNs. Our observation also validates the assumption made by other multicast design that multicast group traffic follows a power law distribution [19], [21].

According to our results, we also find the following relations between the multicast group size and the group traffic amount. If a group has a large amount of traffic, its group size is also very large among all groups. However, some large groups may have very little traffic.

C. Multicast Management by SDN

A traditional multicast solution uses a decentralized protocol like IGMP [15] for group management. As a consequence, multicast routing and group management are usually bundled by listening to IGMP messages and maintaining a multicast forwarding table accordingly. However, emerging trends in SDN provide opportunities to move group management logic from switches or routers to the controller. Recent work has suggested to use the SDN controller for multicast management such as group join and leave [20], [24]. Our unique contribution is to let the SDN controller utilize the traffic heterogeneity property and data center topologies for scalable multicast system design.

Centralized group controller can manage multicast group membership in different ways: for example the controller can intercept IGMP messages at edge switches or VM hypervisors for backward compatibility. Although in this work we consider the controller as centralized, DuSM can also deploy multiple controllers in a distributed manner for load balancing and fault tolerance purposes. For example, we can partition the multicast address space into multiple blocks and assign each controller different blocks. The SDN controller can also keep track of group traffic volumes and dynamics using measurement tools on switches [27], [29] so that appropriate actions can be applied for different groups. Our protocol uses features that can be implemented in most commodity switches, like ECMP routing [8], multicast forwarding, IP over IP tunnelling [25] and GRE encapsulation [16]. Furthermore, we have a discussion on how to implement the protocol on OpenFlow-enabled switches in Section III-E.

III. DUSM DESIGN

A. Group Classification

The essential idea of DuSM is to use state-free multicast for the majority of mice groups that contribute very little traffic, and use a state-based or state-heavy method to optimize the multicast paths of elephant groups. In this paper, a multicast group is identified by a unique class D IP address, which is compatible with current sparse-mode multicast protocols like PIM-SM [14]. Motivated by our observation on group traffic distribution, the controller first classifies multicast groups based on their traffic amounts and apply different strategies to elephant and mice groups. In DuSM, elephants are large groups whose traffic amount is higher than a threshold. The other groups are called mice groups. As discussed in Section II-C, most groups would be classifies as mice if the threshold is set appropriately.

The group classification module of the controller keeps track of group traffic by periodically collecting flow statistics from switches using existing measurement tools [27]. All new groups are treated as mice. When the traffic of a group traffic exceeds a pre-defined threshold, this group will be promoted to elephant and new multicast rules will be installed on corresponding switches.

B. Multicast for Mice Groups

DuSM applies state-free multicast to mice groups by using techniques that trade bandwidth capacity for state capacity. We here provide two possible solutions.

(1) *Multicast to unicast translation.* We allow the hypervisor of a server to obtain information of all mice groups in which its VMs participate, by consulting the controller. When a hypervisor receives a multicast packet from one of its VMs, it produces multiple duplicate packets, each of which is sent to a single receiver. Mice group packets will then be forwarded by unicast forwarding rules on switches. When the hypervisor of a receiver gets the packet, it can translate the packet back to a multicast packet for backward compatibility. There are many existing techniques to translate a multicast header to unicast-like packet header by rewriting or overloading socket

operations [26], IP over IP tunnelling [25], or Generic Routing Encapsulation (GRE) [16] to encapsulate multicast packets in the network layer. Our current implementation uses this approach.

(2) *Bloom filter based forwarding.* Bloom filter [7] is a data structure that uses a relatively small space to a group of members. It has been proposed to scale multicast [17] [19] [18]. There are two main approaches, namely in-switch and in-packet Bloom filters. In-switch Bloom filter based multicast allows each switch to maintain a Bloom filter for each outgoing port. If a packet can pass the check of a Bloom filter it will be sent to the corresponding port. In-packet Bloom filter are carried in data packet headers. If an outgoing port pass the Bloom filter, the packet will be forwarded to the port. Both types of Bloom filter may cause false positives and a packet can be falsely forwarded to a port which does not lead to any receiver. Therefore extra bandwidth will be spent. Bloom filter based methods are good fit to deliver packets of mice groups because of two reasons. First, mice groups contribute to very little traffic, hence false positives will cause little bandwidth lost. Second, Bloom filters can be considered as abstract of forwarding state that only cost much less memory compared to the storage of multicast state.

C. Multicast for Elephant Groups

When the controller detects that the traffic of a group exceeds a pre-defined threshold, it treats the group as an elephant group. If all group members are in a same pod, the controller uses a single shared tree for the group. Otherwise, the controller constructs multiple shared trees for the group. Given a core switch as the RP we can uniquely and trivially determine a Steiner tree for a group on a fat tree. Hence the controller randomly selects multiple cores and computes a Steiner tree for each of them. The controller then installs multicast rules on the switches of these trees. A packet will be matched to one of the trees and delivered along the tree to all receivers. We expect that all packets of the group can be matched evenly on these trees to balance bandwidth cost. Note that for TCP-based unicast, splitting packets on different paths is undesired because it will cause out-of-order delivery. However most multicast applications do not use TCP as the transport protocol, thus in-order delivery is not required.

In this paper, we present the preliminary design of multicast for elephant groups as follows. For an elephant group, the controller first computes k Steiner trees for it, each of which is rooted at a core switch. For each group member, the controller installs k rules on the edge switch for these trees. When the edge switch receives a multicast packet, it matches the packet to one Steiner tree and forwards the packet along the tree. To guarantee the correctness of delivery, we should ensure that the matching method is static, i.e., a packet will always be matched to a particular Steiner tree on different switches. Such static matching is implemented as hashing a small piece of packet-related data (such as the checksum) to k bins, each of which corresponds to a tree. The packet will then be forwarded along the tree to all receivers. Hash functions have been deployed on current switches [27] and hence are ubiquitous. In addition, according to the results of [27], hashing does not have strong negative impact to switch throughput.

D. Extension to general topologies

Although DuSM can be easily extended to other hierarchical networks, we are also interested in whether we can apply this system to general topologies. In fact the major problem for elephant groups is to determine the multiple shared multicast trees in an arbitrary topology. The RP (or core) selection problem of DuSM is more complicated than core selection of IP multicast [9], because multiple RPs are required to determine for an elephant group. In addition, the Steiner tree computation is NP-hard in an arbitrary topology. For mice groups, the problem is how to allow the unicast packets to avoid bottleneck links. We propose an iterative heuristic algorithm for k-Steiner Tree Problem. In each iteration, we first build a Steiner Tree by invoking a heuristic solution for standard Steiner Tree Problem, then for each edge in the computed Steiner Tree, we increase the weight by a penalty factor $\alpha(\alpha > 1)$ to avoid the bottleneck created by repeatedly selected edge. This procedure terminates when all Steiner Trees are computed or the weight of some Steiner Tree is larger than a predefined threshold. The second condition is necessary for sparse networks where only a few effective Steiner Trees can be computed. In our proposal we use α times the minimal weight of calculated Steiner Trees as the termination threshold.

E. Discussion on OpenFlow Implementation

In this section we will discuss possible approaches to implement our protocol on an OpenFlow-enabled platform. Note that DuSM is not restricted to OpenFlow and can be implemented on any SDN platform that provides centralized control.

An OpenFlow switch usually has one or more flow tables. Each flow table contains a set of flow entries with matching fields, count and forwarding actions. For switches supporting OpenFlow 1.1 [4] or higher, actions in the flow table can direct packets to a group table for advanced forwarding actions like multicast. Multicast rules representing the Steiner trees computed by the SDN controller are installed in the group table. All group management requests such as group join and leave are directed to the controller, and the controller can update multicast rules accordingly. The packet translation module for mice groups is installed to the hypervisors on physical servers. For each receiver of the group, this module simply creates a duplicate unicast packet and write the receiver address onto the destination field. A simpler way is to apply existing protocols such as IP over IP tunnelling and GRE. In fact, GRE/L3 tunnelling has been listed in OpenFlow 1.2 proposals [3]. This packet will then be forwarded by unicast rules of switches. Since unicast rules can always be aggregated to wildcard rules [5], they usually do not have scalability problem.

For switches without group tables, one can still specify multiple forwarding actions in flow tables to implement multicast. In summary, multicast forwarding operation is supported by most OpenFlow switches.

IV. EVALUATION

A. Methodology

We evaluate the performance of the preliminary design of DuSM on fat tree topologies. Since the IBM WVE and Telecom traces only have limited traffic data. We manipulate the real data by enlarging the traffic sizes and group numbers, while still keep the traffic distribution among groups. We implement the DuSM system, including operation modules on switches, hypervisors, and the controller on the ns-2 simulator. When mapping a group member (VM) to a physical server, we consider two different VMs placement strategies: (i) Random: group members are placed randomly across the network and the traffic has no locality, and (ii) Nearby: members of a same group are placed on nearby servers using a very simple heuristic algorithm, reflecting some level of locality.

In our experiments, we vary the threshold used to separate mice and elephant groups by using three different values: 10 KB, 100 KB and 1 MB. When the threshold is set to 1 MB, DuSM treats most groups as mice. We compare DuSM with PIM-SM [14].

We focus on three performance metrics: multicast state overhead, balance of network traffic among different links, and number of switch updates under group dynamics.

B. Multicast State Overhead

Figure 2 shows the number of multicast rules of 16K groups on edge, aggregate, and core switches of a 1024-server fat tree with Random placement, where **whiskers represent the min and max value; edges of the boxes show the 25th, 50th (median), and 75th percentiles; and stars show the average values**. In general, the multicast state overhead shrinks as the threshold gets larger. When the threshold is 10 KB and 100 KB, our algorithm can save around 60% and 80% rules respectively. We find that the state capacity bottleneck is on the edge switches. Assuming the multicast state capacity of a commodity switch is 1000 according to [20], PIM-SM can support around 15K groups, consistent to the results in [20]. Using the 10 KB and 100 KB thresholds, DuSM can support around 32K (100% more) and 64K (300% more) groups respectively. Hence when a proper threshold is chosen (somewhere between 10 and 100 KB), DuSM can increase the multicast state capacity to support more number of groups by 300% compared to IP multicast.

Figure 3 shows the number of multicast rules of 64K groups on edge, aggregate, and core switches in a 1024-server fat tree with Nearby placement. Similarly DuSM can significantly reduce memory cost on switches. Nearby placement allows the network to support more multicast groups due to traffic locality. Assuming the multicast capacity is 1000, PIM-SM can support less than 60K groups. Using the 10 KB and 100 KB thresholds, DuSM can support more than 120K and 200K groups respectively.

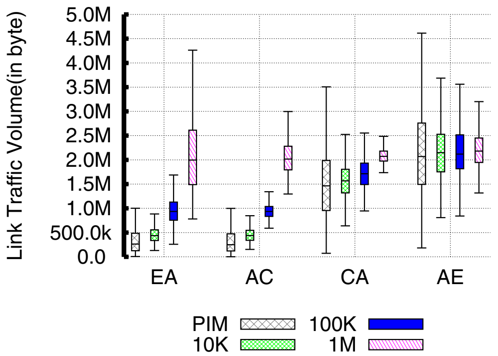


Fig. 4. Traffic rate distribution of 16K groups on a 1024-server fat tree

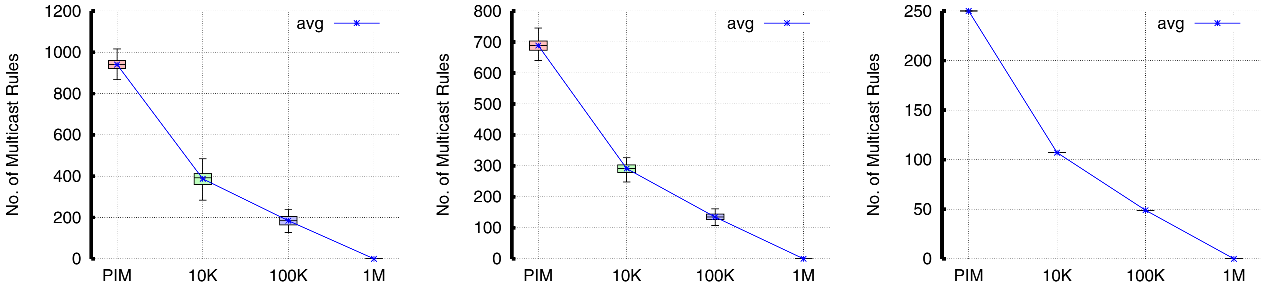
Figure 2 and Figure 3 also show that VM placement strategies can also affect multicast state capacity. When group members are distributed randomly across the network, they have lower probability of sharing the same edge switch therefore a large portion of states(75% in our simulation) could be wasted for small size groups.

C. Network traffic distribution

Figure 4 shows the traffic distribution of 16K groups on different links of a 1024-host fat tree with Random placement. Here we consider four types of links: Edge to Aggregate (EA), Aggregate to Core (AC), Core to Aggregate (CA), and Aggregate to Edge (AE). Note that multicast traffic on EA and AC links includes no replicate copies because EA and AC links are on the path from a sender to the root of the multicast tree. Traffic on CA and AE includes replicate copies to different receivers, and hence is higher than that on EA and AC. Therefore CA and AE links are easier to become bottlenecks. When DuSM is used, extra traffic overhead is introduced by the multicast/unicast translation of mice groups. However DuSM (10 KB and 100 KB) only increases the average traffic on CA and AE links very little (less than 10%). More importantly, we can find that the maximum traffic rate for CA and AE links decreases as the threshold gets larger. Compared to PIM, DuSM (10 KB and 100 KB) can achieve better traffic balance by assigning packets to multiple shared trees and reduce the occurrence of network congestion. Even if our static RP selection algorithm can distribute multicast states across core switches evenly, bandwidth loss on CA and AE links may be caused by hash collisions of two elephant groups. However, our multicast to unicast translation can distribute multicast traffic evenly by employing a multipath unicast protocol like ECMP.

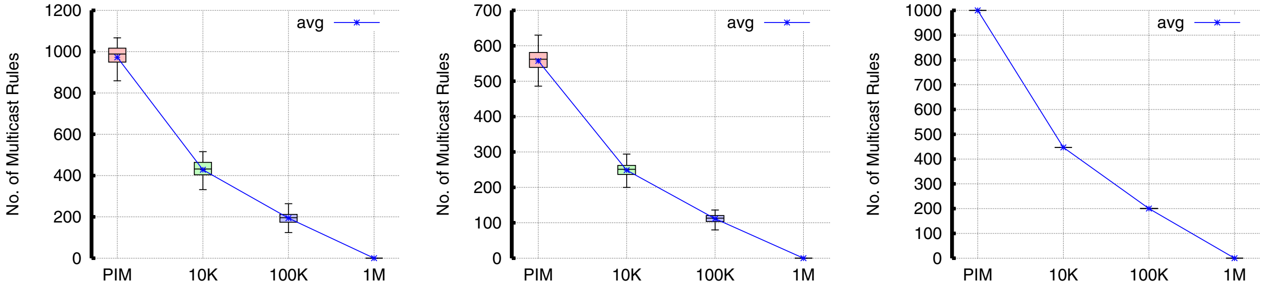
D. Group Membership Dynamics

We evaluate the influence of group membership dynamics by measuring the frequency of switch updates triggered by join and leave events. As discussed in Section II-C, the centralized controller will be informed of the dynamic event first and then disseminates corresponding actions on related switches.



(a) Edge state overhead (b) Aggr state overhead (c) Core state overhead

Fig. 2. Multicast state overhead of 16K groups on a 1024-server fat tree with Random placement



(a) Edge State Capacity (b) Aggr State Capacity (c) Core State Capacity

Fig. 3. Multicast state overhead of 64K groups on a 1024-server fat tree with Nearby placement

#	PIM	10K	100K
Edge updates	2096.66	207.98	85.24
Aggr updates	1787.91	185.20	77.96
Core updates	2483.64	259.52	109.52
Overall aver updates	2050.56	406.67	169.02

TABLE I
AVER. NO. OF SWITCH UPDATES (RANDOM)

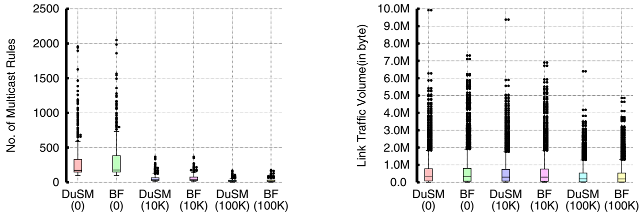
#	PIM	10K	100K
Edge updates	2111.67	198.45	81.97
Aggr updates	598.97	540.80	225.33
Core updates	793.66	554.81	230.50
Overall aver updates	1242.98	181.61	79.22

TABLE II
AVER. NO. OF SWITCH UPDATES (NEARBY)

Generally speaking, only forwarding rules for elephant groups will be installed or removed. Table I and Table II show the switch updates for 16K groups on a 1024-server fat tree with different placement strategies. In general, we can conclude that DuSM can save around 90% switch updates when we set the threshold to 10K, which is proportionate to the saved state capacity in Figure 2.

E. Group Fairness

In this section, we examine whether our algorithm will block the transmission of some flows. We define the transmission fairness for a certain group is the ratio of the delivery time to transmission time. The transmission time is the time span between sending the first message and receiving the last message and the transmission time is the time span from sending the first message to sending the last message. Table III show the transmission fairness for 64K groups on a 1024-host fat tree. Although there is a slight increase in average group fairness, we can still conclude that DuSM has negligible effect on transmission fairness.



(a) Multicast State (b) Link Traffic Volume

Fig. 5. DuSM vs Brute Force(BF) on rock1239 topology

F. k-Steiner Tree Heuristic

In this section we compare our heuristic for k-Steiner Tree Problem to brute force approach using ISP topologies produced by rocketfeul. Like in Fat-tree topologies, we assume traffic volume and group size follow power law distribution and group members are randomly placed across the network. Considering the network size, we choose $k = 3$ and $\alpha = 2$ for our heuristic procedure. Figure 5 shows the multicast state and traffic volume distribution using different k-Steiner Tree Solution for *rock1239* topology, where **boxplot covers 95% data points and the left 5% are presented by solid bullets**. Since our heuristic will terminate when the weight of some Steiner Tree exceeds the threshold, it will return less than k trees sometimes and hence result in slightly less multicast state occupation. Although the brute force approach will compute optimal Steiner Tree Set with least shared links for each group,

#	0	10K	1M
Avg Group Fairness	1.000003	1.000027	1.000027
Min Group Fairness	1.000001	1.000001	1.000001
Max Group Fairness	1.000203	1.000227	1.000227

TABLE III
GROUP FAIRNESS

we can see from Figure 5(b) that our solution can achieve comparable results for most links(more than 95%).

G. Summary of Evaluation

In this section, we show that DuSM can support much more multicast groups than traditional IP multicast. The overall traffic volume does increase but the maximum traffic overhead reduces, indicating better load balance among links. In a hierarchical-tree topology with multiple paralleled paths, network congestion is usually caused by traffic imbalance rather than the overall traffic volume [6]. Hence DuSM does not hurt network scalability on bandwidth. DuSM also reduces the controller overhead for switch updates, which could be another potential scalability bottleneck [20]. DuSM requires computing and memory resource on the hypervisor to perform multicast-to-unicast translation. We leave the evaluation of this cost to future work.

V. RELATED WORKS

Jokela *et al.* propose to encode multicast tree information into in-packet bloom filters and hence eliminate the use of multicast states [17]. ESM [19] enhances this approach by implementing source routing in DCNs with switch-based Bloom Filters. However, the overhead of these approaches comes from unnecessary multicast transmission caused by bloom filters. Some other protocols improve scalability by locally aggregating multicast states at bottleneck switches [20] or globally merging similar multicast groups [26]. DataCast [10] leverages in-network packet caching and multiple Steiner trees in a DCN to address the scalability issues of IP multicast.

Unicast routing is widely adopted as a complementary technique for multicast protocols. Narada [11] organizes end systems into an overlay structure and runs a distance vector protocol on top of the overlay for end-system multicast. Dr. Multicast [26] uses a combination of point-to-point unicast and traditional IP Multicast by overloading the relevant socket operations, for large data centers.

VI. CONCLUSIONS

Software defined networking enables the separation of multicast group management and forwarding. This paper proposes a dual-structure multicast system DuSM by utilizing the heterogeneity property of multicast traffic and centralized control for data center networks. DuSM uses state-free multicast for mice groups, and uses multiple shared Steiner tree for a small number of elephant groups to balance their traffic in the network. Experimental results show that DuSM has great potential as a data center multicast system: it can support much more groups using commodity switches and achieves better load balance, compared to IP multicast. We will explore how DuSM can support dynamic group classification in future work.

VII. ACKNOWLEDGEMENT

Chen Qian is supported by University of Kentucky College of Engineering Faculty Startup Grant and National Science Foundation grant CNS-1464335.

REFERENCES

[1] Ibm websphere. http://www-01.ibm.com/software/webservers/appserv/was/.

[2] Open networking foundation. www.opennetworking.org.

[3] Openflow 1.2 proposals. http://archive.openflow.org/wk/index.php/OpenFlow_1_2_proposal.

[4] Openflow switch specification version 1.1.0. http://archive.openflow.org/documents/openflow-spec-v1.1.0.pdf.

[5] M. Al-Fares, A. Loukissas, and A. Vahdat. A scalable, commodity data center network architecture. In *Proc. of ACM SIGCOMM*, 2008.

[6] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat. Hedera: dynamic flow scheduling for data center networks. In *Proc. of USENIX NSDI*, 2010.

[7] B. H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426, 1970.

[8] Y. Cai, L. Wei, H. Ou, V. Arya, and S. Jethwani. Protocol Independent Multicast Equal-Cost Multipath (ECMP) Redirect. RFC 6754, 2012.

[9] K. Calvert, E. Zegura, and M. Donahoo. Core selection methods for multicast routing. In *Proc. of Computer Communications and Networks*, 1995.

[10] J. Cao, C. Guo, G. Lu, Y. Xiong, Y. Zheng, Y. Zhang, Y. Zhu, and C. Chen. Datacast: A scalable and efficient reliable group data delivery service for data centers. In *Proc. of ACM CoNEXT*, 2012.

[11] Y.-h. Chu, S. G. Rao, and H. Zhang. A case for end system multicast. In *Proc. of the ACM SIGMETRICS*, 2000.

[12] C. Clos. A study of non-blocking switching networks. *Bell System Technical Journal*, 1953.

[13] W. Cui and C. Qian. Difs: Distributed flow scheduling for adaptive routing in hierarchical data center networks. In *Proc. of ACM/IEEE ANCS*, 2014.

[14] D. Estrin et al. Protocol Independent Multicast-Sparse Mode (PIM-SM): Protocol Specification. RFC 2117, 1997.

[15] B. Fenner. IANA Considerations for IPv4 Internet Group Management Protocol (IGMP). RFC 3228, 2002.

[16] S. Hanks, T. Li, D. Farinacci, and P. Traina. Generic Routing Encapsulation (GRE). RFC 1701, 1994.

[17] P. Jokela, A. Zahemszky, C. Esteve Rothenberg, S. Arianfar, and P. Nikander. Lipsin: Line speed publish/subscribe inter-networking. In *Proc. of ACM SIGCOMM*, 2009.

[18] D. Li, H. Cui, Y. Hu, Y. Xia, and X. Wang. Scalable data center multicast using multi-class bloom filter. In *Proc. of IEEE ICNP*, 2011.

[19] D. Li, Y. Li, J. Wu, S. Su, and J. Yu. Esm: Efficient and scalable data center multicast routing. *IEEE/ACM Transactions on Networking*, 2012.

[20] X. Li and M. J. Freedman. Scaling ip multicast on datacenter topologies. In *Proc. of ACM CoNEXT*, 2013.

[21] X. Li and C. Qian. Low-complexity multi-resource packet scheduling for network functions virtualization. In *Proceedings of IEEE INFOCOM*, 2015.

[22] X. Li and C. Qian. Traffic and failure aware vm placement for multi-tenant cloud computing. In *Proceedings of IEEE IWQoS*, 2015.

[23] M. Mahalingam et al. VXLAN: A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks. IETF Internet-Draft, 2013.

[24] Y. Nakagawa et al. Domainflow: practical flow management method using multiple flow tables in commodity switches. In *Proc. of ACM CoNEXT*, 2013.

[25] W. Simpson. IP in IP Tunneling. RFC 1853, 1995.

[26] 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 *Proc. of EuroSys*, 2010.

[27] M. Yu, L. Jose, and R. Miao. Software defined traffic measurement with opensketch. In *Proc. of NSDI*, 2013.

[28] Y. Yu and C. Qian. Space shuffle: A scalable, flexible, and high-bandwidth data center network. In *Proceedings of IEEE ICNP*, 2014.

[29] Y. Yu, C. Qian, and X. Li. Distributed collaborative monitoring in software defined networks. In *Proc. of ACM HotSDN*, 2014.