Scalable Multicast Concepts

Fabian Keller
University of Stuttgart
Email: hello@fabian-keller.de

***Abstract*—There are many use cases where applied multicast concepts can have a huge impact on application performance. Especially in data centers huge amounts of data can be transferred through the network using multicast techniques. The problem with multicast is scalability - routers have to keep track of multicast forwarding states and often can't handle huge amounts of multicast groups. In this paper we examine different approaches enhancing the overall performance of multicast and compare them to each other. First we are going to look at tree-building multicast algorithms like REUNITE operating from routers. Another approach is injecting the multicast logic into the sender right beneath the application layer like DR. MULTICAST does. Lastly we examine how multicast can be easily approached with software defined networking.**

I. Introduction to IP Multicasting

IP multicast (IPMC) is a protocol for sending IP datagrams to a set of hosts identified by a sole IP address and was first introduced in RFC-1112 by S. Deering. A host can join and leave a multicast group at any time - a multicast group can even be empty. Sending a datagram to a multicast group means sending the datagram to all joined hosts. In contrast to unicast where the sender sends the datagram at least once for each receiver, the datagram is being replicated on the route to the joined hosts. It is trivial to see that this approach can save a lot of bandwidth when sending the same datagram to many hosts.
IPMC defaults to a IP time-to-live (TTL) for all multicast packets of 1. This means that multicast is restricted to a single network by default (see Fig. 1). For inter-networked multicasting "multicast routers" take care of forwarding the packets. Routing between multicast routers is a difficult task and its scalability is the main focus of this paper. This chapter introduces IPMC as described in [1] to provide a basic comprehension for the topic and point out the problem we focus before we will examine multicast routing in the following chapters.

A. Multicast Group Management

In order to support various multicast groups it is necessary to keep track of the hosts membership to the groups. Therefore the IP service interface provides two group management operations:

```
JoinHostGroup( group-address, interface )
LeaveHostGroup( group-address, interface )
```

Both operations require a specific multicast group address and a network interface, as listening to a multicast group is specific to a network interface of the data link layer even though multicast takes place in the transport layer. The IPv4 multicast group addresses range from 224.0.0.0 through 239.255.255.255 [1], whereas IPv6 multicast groups use the address prefix ff00::/8 [2]. After a host joins a multicast group the IP module adds the group address to its list of group memberships. Hence when a host leaves a multicast group the group address is being removed from that list.
The router connecting the hosts does not keep track of which host has joined which group, it only keeps track of which groups have been joined by all connected hosts. Using the Internet Group Management Protocol (IGMP) the router keeps sending a Query message for every multicast group periodically to all connected hosts in order to determine if at least one host has joined the group. A host receiving a Query replies after a randomly chosen delay `D` less than 10 seconds with a Report message stating whether the host is still joined to the group or not. The report is destined to the group address being reported and thus also is a multicast message staying in the same network, as the TTL of the report packet is fixed to 1 [1]. When other hosts waiting for their random delay to time out receive such a report they quietly stop their timer and do not send a report. For the multicast router, receiving a single report is enough information to know that there are still joined hosts in the network. This behaviour prevents flooding the multicast network with unnecessary Report and Query traffic.

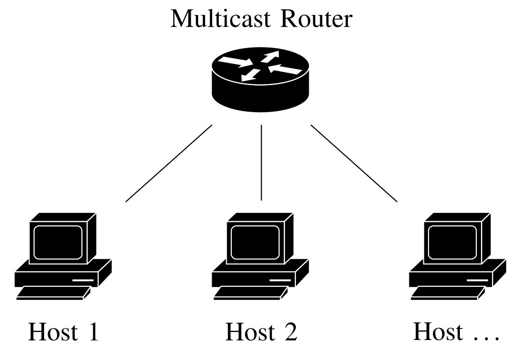


Fig. 1. Network topology in which IPMC is defined. IPMC does not describe multicast forwarding between multicast routers.

B. Sending Messages to a Multicast Group

A source can send a datagram to a multicast group as easy as if it was sending in unicast mode; the only difference is

that the IP destination address has to be a multicast group address. As IPMC is not a reliable communication method no Internet Control Message Protocol (ICMP) messages (e.g. Time Exceeded, Destination Unreachable) are returned from receivers.
To improve performance of multicast transmissions it is possible to map multicast group addresses to certain network addresses (e.g. Ethernet addresses). For IPv4 multicast in Ethernet the 23 low-order bits of the multicast group IP address are added to the Ethernet multicast address 01-00-5E-00-00-00 (hex) leading to tolerated ambiguities as a multicast group address has 28 significant bits. These ambiguities are the cause for receiving packets of a group that is not joined by the receiver on the Ethernet module. When the packets are passed to the IP module they are discarded, as the IP module does the actual (and reliable) group filtering. If Ethernet does not support any filtering at all the filtering would then be done solely by the IP service.

C. Multicast Routing

We now know that hosts can join and leave multicast groups within their local network segment using IGMP to communicate with their multicast router. If the multicast should go beyond the local network the multicast router has to forward the packets to other multicast routers. It is important to prevent cyclic forwarding, as cyclic forwarding unnecessarily strains bandwidth. Thus it is a goal of all multicast routing protocols to have a cycle-free forwarding topology while reaching all joined hosts. As tree topologies are cycle free, it is a common approach to construct a spanning tree reaching all receivers over the network. Tree based approaches can now be divided into two categories: shared-tree algorithms and source-based tree algorithms [7] which both construct a spanning tree over the network connecting all receivers.
Source-tree algorithms like Simple Explicit Multicast (SEM) [9] construct the tree starting with the source as root node. If there are multiple sources for a single group one tree has to be constructed for each source. To minimize calculation overhead shared-tree algorithms like Core Based Trees (CBT) [3] and REUNITE [8] construct a single tree and define a fixed entry point, the so called Core Router [3]. Multiple sources can now send to the network by sending to the Core Router which takes care of forwarding to all joined receivers.

D. Scalability Concerns

Scalability is a term that can be construed in various different ways. In order to talk about scalability of IP multicast we need to exactly define what scalability is. A lot of research has been done on multicast algorithms that scale for a specific use case. Until today there is no overall solution to scalability, but different algorithms scale better under certain circumstances than others. The following aspects define the term "scalability" considering multicast concepts in this paper:

1) The number of joined receivers
2) The number of sources
3) The number of routers in a minimal spanning tree connecting all receivers
4) The number of different multicast groups within the same set of receivers
5) The frequency of join/leave operations
6) The frequency of data transmission through multicast

Scaling aspects 1) through 4) means directly increasing the number of Multicast Forwarding Table (MFT) entries in the multicast routers. To tackle this problem it is a common approach to reduce multicast states. Multicast Tunneling with Branch Filtering (MTBF) [6] aggregates similar multicast forwarding entries to a single entry to decrease the number of entries in the MFT. Distributed Core Multicast (DCM) [4] decentralizes routers with lots of multicast groups passing through, e.g. ingress and egress routers of a backbone network, by tunnelling between multiple ingress and egress routers and keeping the multicast state only in them [5]. REUNITE [8] and SEM [9] eliminate forwarding states in non-branching routers, as non-branching routers only need to know their previous and next neighbour router. In contrast to the previous algorithms, Dr. Multicast [11] intercepts all IPMC calls and modifies/optimizes them based on administrator-specified policies.
Scaling aspect 5 triggers tree maintenance operations frequently, as joining or leaving a group modifies the constructed forwarding tree. Various algorithms tend to optimize traffic forwarding by constructing a highly sophisticated and optimized forwarding tree, but maintaining the tree can become very expensive.
Scaling aspect 6 shows how scalable the forwarding of the packets itself is. Packets have to be examined and replicated in branching routers. Depending on how often this has to be done on the way to the receivers this can have an impact on latency.

E. Paper Roadmap

Brevity prevents us to go into detail for all mentioned algorithms. In section II we are going to examine SEM and REUNITE - two different approaches to reduce the multicast forwarding state. The key difference between both algorithms is that SEM constructs a source based tree whereas REUNITE constructs a shared tree.
A new approach especially for data centres is to have a multicast layer between the application layer and the network layer. The reason for this is that the multicast layer can optimize group memberships, tree sharing between groups, multicast address allocation and restrict access to resources in times of peak traffic. We will discover the approach *Dr. Multicast* takes in section III.
In software defined networking (SDN) we have a great advantage over Dr. Multicast: a centralized controller that always knows every detail of the network topology and its load. In this approach we use the information given by the controller to optimize multicast message delivery. The approach also ensures to be more reliable by giving the ability to easily

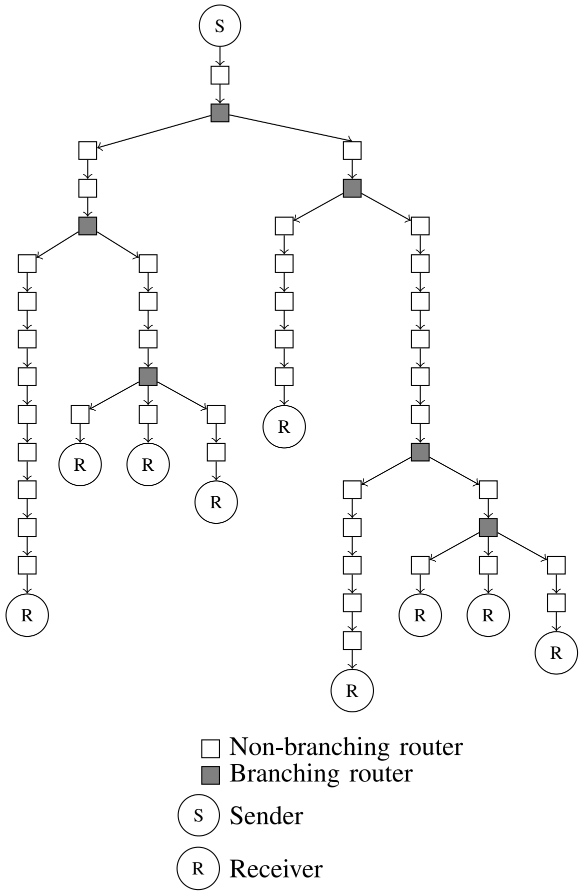


Fig. 2. The topology represents a sample multicast forwarding tree. Messages sent by the Sender have to be replicated at all branching routers for the Receivers to receive them. Non-branching routers just forward the message.

have fall-back trees. In case of node failure the trees can be switched very quickly to keep packet loss at a minimum. We will present the design of the controller in section IV.

II. Multicast State Reducing Algorithms

A commonly seen multicast situation is shown in Fig. 2. A single source sends a multicast message to a set of receivers over a multicast forwarding tree. On the route of the packets there are a few branching routers which need to know how to replicate the message. The non-branching routers just keep forwarding the messages.

A. Simple Explicit Multicast

SEM as described in [9] is a source-based tree construction algorithm and completely removes multicast forwarding states in non-branching routers. The key concept behind this is that all branching routers know their previous branching router and a list of all the next branching routers. Thus, the branching routers themselves do not know anything about the non-branching routers in between. SEM then uses unicast messages between branching routers to transport data.

1) Joining Multicast Groups: If a receiver *R* wants to join a group *(S,G)* with *S* being the source and *G* being the IPMC group address, the receiver sends a IGMP join message destined to *G*. The multicast router of the local network where *R* resides intercepts the join message and sends a SEM-specific *join(S,G)* message to the source. This router is henceforth called the Designated Router *DR* of *R*. If the *DR* does not know the source of the Group *G* it determines the source using IGMPv3 [10] which supports source discovery. As soon as the source receives the *join(S,G)* message, it adds the address of the *DR* to its list of receivers for group *G*.
In order to stay joined the *DR* of a set of joined receivers has to send *alive* messages to its previous branching router. The previous router continues sending a new *alive* message to its previous router until the alive message recursively reaches the source. When a *DR* stops sending *alive* messages to its previous branching router a time-out will trigger the previous branching router to send a *leave(S,G,DR)* message directly to the source, which then removes the *DR* from the list of joined receivers.

2) Tree Construction: To construct the multicast forwarding tree the source partitions the list of joined *DR*s into sublists L_i. The lists according to their next unicast hop router are then sent to their respective next hop router in a *branch(S,G,pB,*L_i*)* message (with the previous branch router *pB=S*). A router receiving a branch message partitions the list of *DR*s the same way and either forwards the branch message if there is only one partition, or it replaces the previous branching router *pB* with itself and sends it to the corresponding next hop routers. A router extracting multiple partitions from the list of *DR*s now is a branching router and sends a *previous_branch* message to the previous branching router. As non-branching routers have simplify forwarded the *branch* message, the address of the previous branching router can be easily determined. When a router receives a *previous_branch* message it knows one of its next branching routers. As soon as this operation finishes there is a route from the source to each DR, which can now be used to send data packets to the group *(S,G)*.

3) Tree Maintenance: As we have already seen above the source receives a *leave(S,G,DR)* message as soon as a *DR* stops sending *alive* messages. If the last branching router before the leaving *DR* has only two descendants, it afterwards has to degrade to a non-branching router (only one descendant left). To fix this the source sends a new *branch* message after receiving the *leave* message and a time-out. This rebuilds the complete forwarding tree.
When a new receiver joins a group, SEM possibly requires a new branching router in its forwarding tree. Again, the source sends a new *branch* message after receiving the *join* message and the same time-out as with *leave* operations. Due to the time-out the tree could possibly be in a state in which forwarding to a subset of the receivers fails. To improve reliability SEM sends data packets in GXcast mode [9] while it knows that the tree might be broken and is being rebuilt.

4) SEM Example: Figure 3 illustrates how the SEM protocol works. *A,B,C,D,E* and *F* are all receivers using IGMP

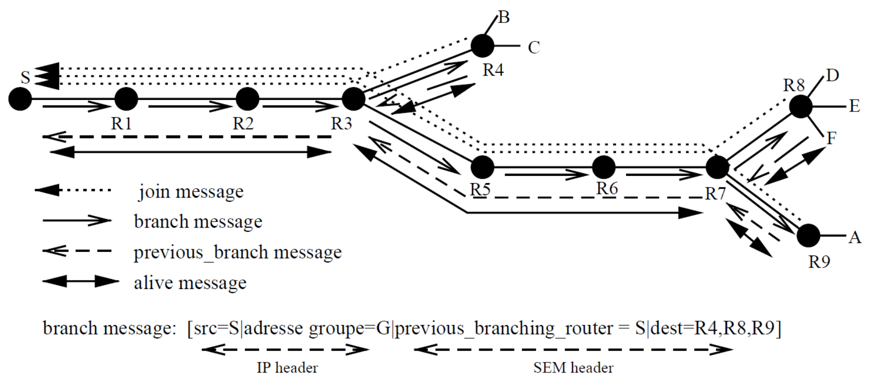


Fig. 3. SEM tree construction. Taken from Ref. [9]

to send a *join* message to their corresponding DR - *B* and *C* send to *R4*, *D,E* and *F* send to *R8* and *A* sends to *R9*. The DRs then send a SEM *join(S,G,Rx)* message to *S* where *Rx* represents the individual router name. The source *S* adds the set of routers to its receivers list. It then sends a *branch(S,G,S,$\{$R4,R8,R9$\}$)* to R1. This branch message is being forwarded hop-by-hop until R3. R3 partitions the list of DRs and sends two branch messages: *branch(S,G,R3,$\{$R4$\}$)* to *R4* and *branch(S,G,R3,$\{$R8,R9$\}$)* to *R5*. A further partitioning of the last branch message then occurs at *R7*. All branching routers then send a *previous_branch(S,G,Rx)* message to their corresponding previous branch after finishing the branch operation, where *Rx* is the name of the router sending the message: *R3* to *S*, *R4* to *R3*, *R7* to *R3*, *R8* to *R7* and *R9* to *R7*.
The forwarding tree is now constructed and S can start sending data to all receivers. The topology contains 9 routers: 2 branching routers, 3 DRs and 4 non-branching routers. To join the 6 receivers and to construct the tree, SEM used 6 IGMP join messages, 3 SEM specific *join* messages, 9 *branch* messages and 5 *previous_branch* messages, which is a total of 17 SEM-specific control messages.

B. SEM scalability

Having understood the key concepts of SEM we now have a quick glance at it's scalability. We back our considerations on the analysis of SEM in Ref. [9]. Figure 4 shows an excerpt of their analysis. Fig. 4a and Fig. 4b show the number of group members on the horizontal axis and the amount of transmitted data quantity in the core network in bytes on the vertical axis. Measurements were taken in the Internet2 topology of the Abilene[1] network with 90, 140 and 190 receivers. The P-X lines represent a measuring where X% of the traffic are transmitted using SEM and the remaining percentage being transmitted using GXcast. SEM uses GXcast during tree construction to prevent loss of packets.

1) Scaling the number of joined receivers: Scaling the number of receivers strongly depends on the network topology. SEM scales good if there is a large amount of non-branching routers. In [9] the French internet topology in 2005 was examined and calculations states that using SEM could reduce multicast forwarding table size in such a topology of up to 66%. Regarding 4a and 4b the P-100 (pure SEM without GXcast) scales very good between 90 and 190 receivers, in fact not a real increase in transmitted data quantity can be measured. The measurement does not include the traffic being generated between the DRs and the real receivers.

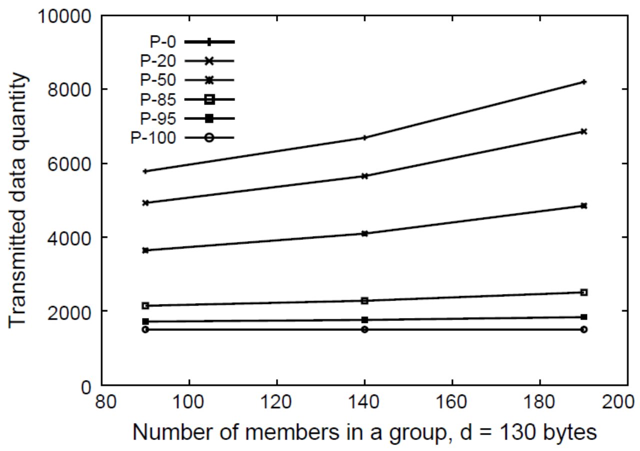
2) Scaling the number of sources: As SEM has to build a forwarding tree for every source, it does not scale good at all for multiple sources. Having *n* sources for a group G the branching routers need to maintain *n* different MFT entries, as entries are identified by *(S,G)*, which is like having *n* different multicast groups.

3) Scaling the number of routers in a minimal spanning tree connecting all receivers: SEM scales perfectly as long as additional routers are non-branching routers. They do not need to have a single multicast table entry for the groups passing them. On the other hand if we have lots of branching routers SEM had to send a lot more alive messages over shorter distances. Thus having a sparse network topology is ideal for SEM.

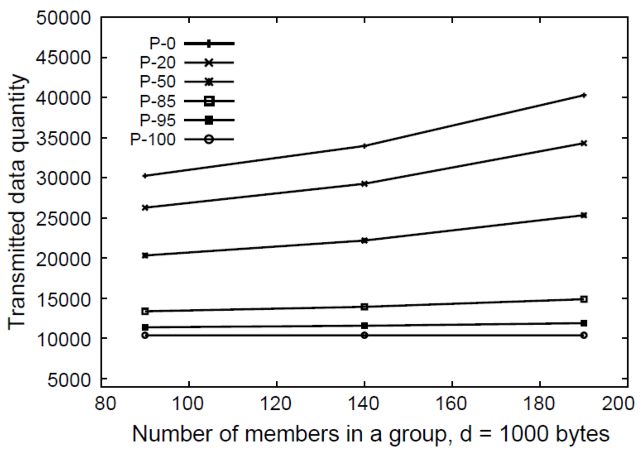
4) Scaling the number of different multicast groups within the same set of receivers: SEM scales linear to the number of groups as no similar forwarding states are aggregated in the multicast forwarding tables. Taking 140 receivers in Fig. 4a, a total of $130bytes * 140 = 18200bytes$ have to be transmitted to the receivers. Most of the data replication process happens between the DR and the real receiver thus with P-100 and a transmitted data quantity of roughly $1700bytes$ (taken from the image) this is a traffic overhead of nearly 10%. Taking into account that every further multicast group also has an overhead of 10%, bandwidth limits scaling the number of groups at a certain point.

5) Scaling the frequency of join/leave operations: Join and *leave* operations are the most expensive operations in SEM as they trigger a complete rebuild of the forwarding tree. This happens only once in a certain time-out interval, but sending in GXcast in between imposes a high computation load on branching and non-branching routers on all routes. In Fig. 4a and Fig. 4b it is obviously clear that GXcast does not

[1]abilene.internet2.edu



(a) 130 bytes of transmitted data



(b) 1000 bytes of transmitted data

Fig. 4. The transmitted volume in the core network with protocols GXcast and SEM. Fig. 4a and Fig. 4b have been taken from Ref. [9].

scale well. P-0 (pure GXcast) puts roughly 3-4 times more transmission load on the network than P-100 (pure SEM).

6) Scaling the frequency of data transmission through multicast: Scaling data transmission has little impact on SEM's scalability as forwarding occurs in constant time once the forwarding tree has been set up. To transmit $130bytes$ to 140 receivers in Fig. 4a with P-100, an overhead of about $1700bytes$ is required. As we see in Fig. 4b, to transmit $1000bytes$ to 140 receivers an overhead of about $11000bytes$ is required. The $\frac{transmitted\ data}{overhead}$ quotient is between 7% and 10% when considering the other group sizes as well.

C. REUNITE

REUNITE [8] is a shared-tree algorithm with multicast state reduction. Like SEM it distinguishes between branching and non-branching routers. REUNITE requires to define a root node which then functions as tree root. This does not have to be the multicast sender, but with a single sender it stands to reason to take the sender as root. Additionally, if only a subset of all routers in the network implement REUNITE it, will still work properly. Next to that REUNITE supports load balancing and sender access control.
The key concept is to maintain two tables in all routers. A multicast control table (MCT) and a multicast forwarding table (MFT). The MCT is used to store tree state information in the following format: *((root_addr, root_port), (dst))*. The first tuple identifies the group and *dst* is used to store the IP address of the first receiver that joined the group on that particular downstream route. The MCT is only used by tree maintenance and construction operations and therefore, lookups in the MCT table are only required when evaluating REUNITE specific control messages. For plain data forwarding only the MFT table is consulted which holds entries of the following format: *((root_addr, root_port), (dst, stale),* $\{$*(rcv*$_1$*, alive*$_1$*), ..., (rcv*$_n$*, alive*$_n$*)*$\}$*)*. The first tuple again identifies the group and the second tuple *(dst, stale)* contains the IP address of the first receiver that has joined the group on that particular downstream route. Then a list of all receivers *rcv*$_i$ is stored to which the router will send replicated messages. The *alive* and *stale* boolean flags store whether the receivers are alive and whether the route is about to become stale. Due to this functional difference MCT tables are used in non-branching routers and MFT tables are used in branching routers.
Furthermore REUNITE uses unicast IP addresses for data sending and group addressing. The group address is defined by the tuple *(root_IP_addr, root_port_number)* which prevents group address interference when using multicast with a large number of applications over the same network.

1) Joining multicast groups: To join a multicast group, a receiver *R* sends a *join* message destined to the root node. The first router on the way belonging to the group distribution tree (either MFT or MCT entries present) intercepts the join message. If the intercepting router already has a MFT entry it adds *R* to the list of receivers obtaining duplicated messages. A router modifying its MFT entry through a *join* message always discards the join message afterwards. Otherwise if the router has a valid MCT entry for that group the router will discard the MCT entry and create a corresponding MFT entry setting *dst*=*R* to become a branching router. This happens only if the *dst* value in the MCT entry is not equal to the IP address of the receiver sending the *join* message. If they are equal, the receiver sending the *join* message is already part of the distribution tree and sends the *join* message to stay alive. The root always adds incoming *join* requests to the list of receiver in its MFT. In the example in Fig. 5 a) R1 sends a join message which propagates to the source *S* as no router on the way has any MCT entries.

2) Tree construction: Having at least one receiver, the source starts sending periodic unicast *tree* messages destined to each joined receiver found in the MFT. All routers intercept *tree* messages. If a router has a MCT entry, it forwards the tree message. Branching routers with MFT entries replicate

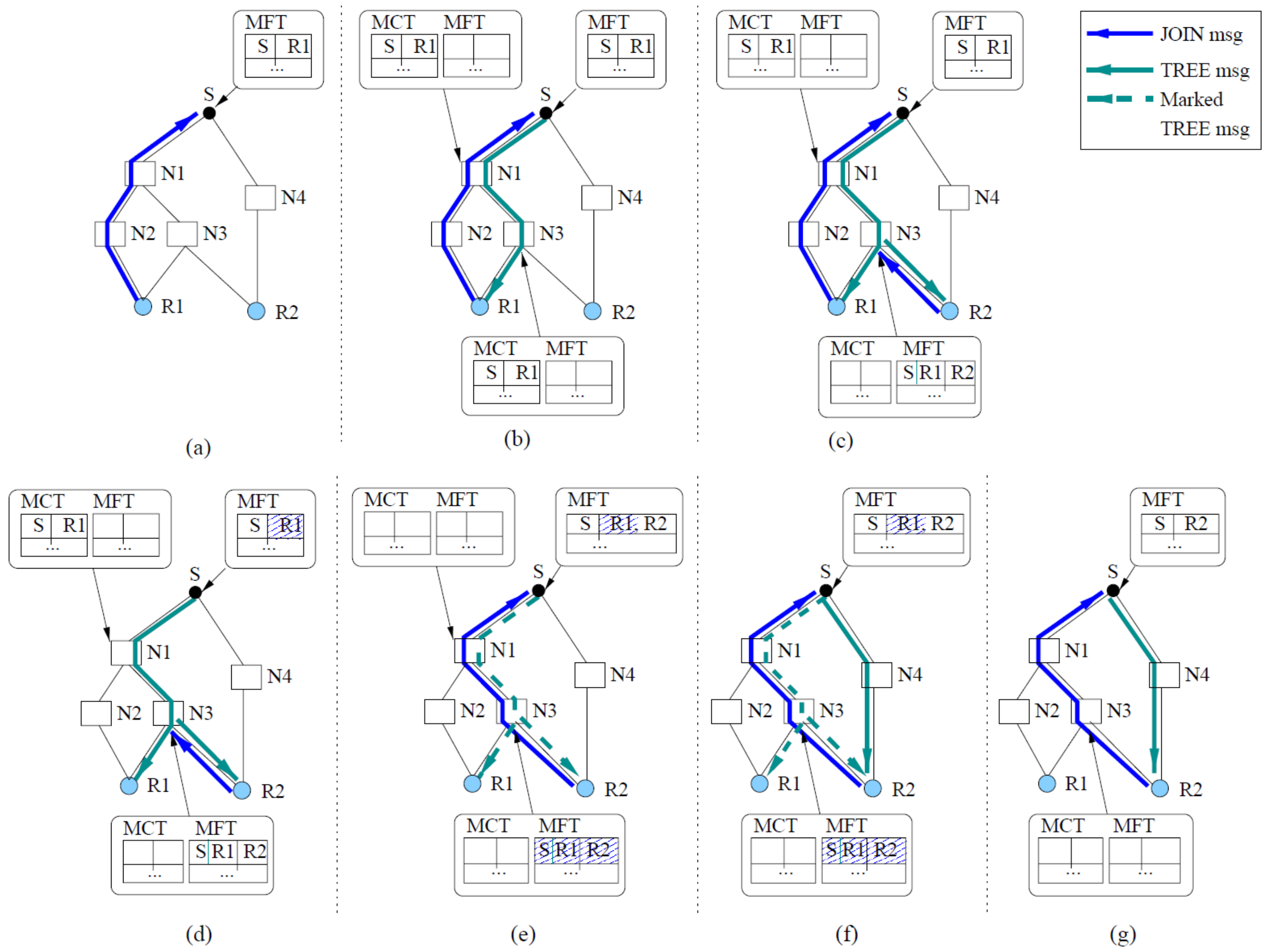


Fig. 5. Illustrates basic REUNITE operations with S being the source and root node. a) R1 joins b) *tree* message c) R2 joins d) R1 leaves e) stale *tree* message, R2 join proceeding to source S f) *tree* discovers new route to R2 g) Completing *leave* operation of R1. Taken from Ref. [8].

the *tree* message for each receiver. If a router has no MCT entry for *(root_addr, root_port, dst)* it creates one setting *dst* to the destination of the *tree* message. In Fig. 5 b) S sends a tree message destined to R1. As REUNITE uses unicast to send the messages, the *tree* message might find a route that differs from the *join* message. As data flows through the routes discovered by *tree* messages these are ideal routes optimized by unicast routing.
In Fig. 5 c) R2 sends a *join* message destined to S. N3 intercepts that message and adds R2 to its list of receivers. Tree messages sent by the source and destined to R1 are replicated by N3 and sent to all registered receivers, as the source does only keep track of a subset of receivers.

3) Tree maintenance: The receivers periodically keep sending *join* messages destined to the root node and the root node periodically keeps sending *tree* messages. If a receiver wants to leave a group it stops sending *join* messages. The intercepting router will notice that and mark the entry in its MFT as not being alive. In Fig. 5 d) R1 stopped sending the *join* message and is marked as not being alive in S.
If a receiver R is not marked as alive, all *tree* messages destined to R have a *stale* bit set. This indicates that the branch is about to be dropped. As shown in Fig. 5 e) N3 receives a stale tree message and now forwards all incoming *join* messages, as its about to be degraded to a non-branching router. In Fig. 5 f) the source sends a stale *tree* message destined to R1 and a normal *tree* message destined to R2 as it has received a *join* message from R2. In Fig. 5 g) the stale branch has been dropped and a new branch to R2 has been established. This means that all MFT and MCT tables of N1, N2 and N3 are empty and that the MCT table of N4 contains a new entry for the group.
A receivers alive status is set to not alive after a time-out occurs. Receiving a *message* from that receiver resets the time-out. As soon as a receiver is marked as not alive, the corresponding tree messages will be marked as stale to inform any receivers not visible to the branching router that the branch will be dropped. Stale tree messages are sent until a second

time-out expires, then the branch will be dropped.

D. REUNITE scalability in comparison to SEM

In section II-B we examined the scalability of SEM. We are now going to examine the scalability of REUNITE, compare it to SEM and explain why they scale different.

1) Scaling the number of joined receivers: Scaling the number of joined receivers also mainly depends on the network topology as group membership is stored across all routers in the tree. Scaling the topology is discussed below.

2) Scaling the number of sources: REUNITE does not make any difference having one or multiple sources. Multiple sources in REUNITE are managed by tunnelling them through the root node of the forwarding tree. While this node is the bottle neck of multiple senders, it is the only reliable way to perform access control. All sources send their messages to the root node using unicast, which then starts distributing after successful authentication. This small trade-off concerning latency has been accepted for having a reliable access control. Compared to REUNITE, SEM needs an additional MFT entry in all tree routers for each additional sender. SEM does then not have a latency as the trees are constructed using the sender as root node, but requires a lot more MFT entries.

3) Scaling the number of routers in a minimal spanning tree connecting all receivers: In all non-branching routers one entry in the MCT of the router has to be installed. In branching routers an entry for every branch has to be created. REUNITE also supports the deployment to a mixed topology with REUNITE aware nodes and not REUNITE aware nodes. As only unicast messages are used, non REUNITE aware routers simply forward the messages according to their normal routing tables, which does not influence the correctness of the protocol. If the MFT or MCT of a router is overloaded, the router can pretend to be a non REUNITE aware router and forward the incoming messages. The message delivery will still be correct, but not as efficient as it could be. If the root node was the only REUNITE aware router in the network, it would send a unicast message to every joined receiver.

4) Scaling the number of different multicast groups within the same set of receivers: REUNITE cannot share trees between multiple multicast groups and thus a new multicast tree has to be constructed for every additional group. If a router is overloaded with MCT or MFT entries, it behaves as if it was a non REUNITE aware router for new additional multicast groups. This leads to a linear and fault tolerant group scalability.
SEM is not as fault tolerant as REUNITE, but saves entries in all non-branching routers. Nevertheless the bottleneck of both algorithms are the branching routers, as for every new group the branching routes MFT entry count increases by the number of outgoing branches. All in all, SEM still has a slight disadvantage as it cannot cope with an overloaded MFT.

5) Scaling the frequency of join/leave operations: As there is no global node which keeps track of all group memberships, every receiver has one router in the tree that keeps track of its group membership. Depending on how the tree was constructed this could be any router between the designated router of the receiver and the root node. Join and leave operations will only affect the branch below that certain node. In comparison, SEM sends a branch message from the source down the whole tree after join and leave operations. Therefore SEM does not require to send tree messages to keep track of the tree state as only alive messages between branching routers are exchanged. Join and leave operations are hence more expensive.

6) Scaling the frequency of data transmission through multicast: REUNITE uses unicast to deliver multicast data packets between two branching routers like SEM does. REUNITE always sends data packets destined to a receiver. All branching routers on the way intercept data packets and send copies to the corresponding branches. SEM sends data packets destined to the next branching router. When a router receives such a packet, it sends copies to all the next branching routers in the list. The data forwarding does differ slightly but performance mainly depends on the implementation, as lookups in the MFT table are possible in constant time, as no hierarchical lookup has to be performed.

III. Multicast in Data Centres

The previously mentioned multicast protocols often scale well in a sparse network topology, having few receivers scattered over a large topology. This differs in data centres, as there is a highly linked and dense network topology. Most multicast algorithms (like SEM and REUNITE) scale quite well in a sparse environment, but in a dense environment the amount of non-branching routers decreases drastically. Next to that, data centres have to scale well with an increasing number of multicast groups and moreover have to be fault-tolerant. Especially multicast-storms are the antagonists of a working data center, which can occur in times of high bandwidth usage. Receivers might detect a loss of packets and request them again which fans the embers.
A simple experiment [11] reveals a common bottleneck. Two computers communicating through various multicast groups over a single switch. A sender is located on one computer sending 15.000 packets/s. A receiver is located on the other computer. As the sender sends packets to twice as many multicast groups as the receiver receives from, one expects the receiver to receive 7500 packets/s. Figure 6 shows that IPMC scales up to 100 joined groups in this simple setup. After that the packet loss rate starts rising. With 350 joined groups we reach a packet loss rate of 25%.

A. Introducing Dr. Multicast (MCDC)

MCDC as describe in [11] tackles the problem the following way: It provides a new *sockets.h* library in which common IP multicast methods are overridden by MCDC. The idea behind this library is to motivate developers to use standard IPMC operations in their applications. MCDC intercepts all those calls, optimizes them in the kernel and then sends them to the network.
A mapping and a library module are the core modules of

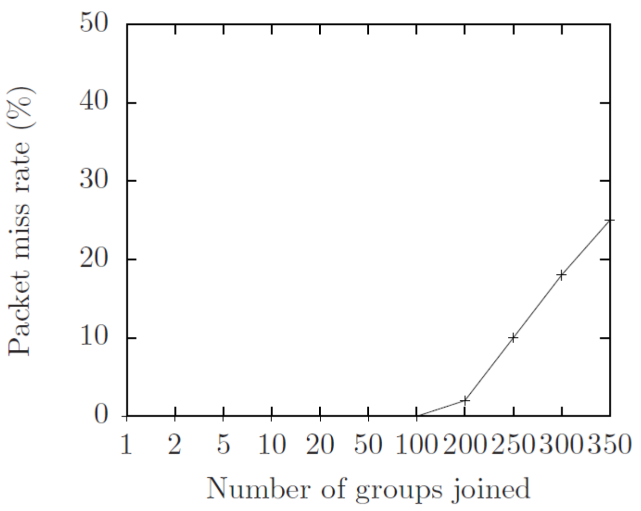


Fig. 6. Receiver packet miss rate vs. number of IPMC groups joined. Taken from Ref. [11].

MCDC. The library module intercepts and modifies the messages going through the network layer. The mapping module optimizes multicast usage through various methods as it has access to a gossip layer which provides health information of the complete network.
As the mapping module knows the different bottlenecks of the network and all multicast groups, it can cluster and aggregate different multicast calls to use a single multicast address. Moreover it can switch to simple unicast calls for small groups and use multicast only for large groups. Next to that the administrator can define a maximum limit of group allocations dependent on the network capability, he can limit the overall multicast bandwidth and allow or deny different applications or hosts to use multicast.
We will now have a look at how the mapping module works and afterwards consider MCDCs scalability.

B. Mapping Module

The mapping module basically works as a group membership service and as an IPMC address allocation service. It therefore uses a gossip-based control plane to keep track of the system state. As the mapping module has to run on every node, synchronizing the system state across all nodes is the most important task.
In order to keep a consistent state only a certain node acting as leader can allocate IPMC addresses. All other nodes only need a global snapshot in order to map multicast calls to any group appropriately. Theoretically, any node having a consistent snapshot of the global state could allocate IPMC addresses. This is used to increase MCDCs reliability. If the leader fails, any other node takes control of the leader role. Again, choosing a new leader node is an administrative task and thus can be defined by the administrator, but defaults to using the oldest node in the system.
When group memberships update it is necessary to deploy the new global state to all nodes. A node does this by maintaining a global table in which for each node of the system a heartbeat value is stored. After a time-out of T milliseconds the node sets its own heartbeat to the current system time and then randomly selects a node from the system with which it will exchange the global table. Exchanging is done in a very efficient matter by hashing the heartbeat values of the two tables and looking for different hashes. If different hashes are found, an updating patch is sent to whichever node has the more outdated version. This way the new global state will eventually propagate through the whole system, but especially for applications that require fast group joining it is important to minimize the latency it takes to update the global table in all nodes in the network. There are two strategies to reduce the latency, which also can be configured by the administrator as they sit on top of the normal gossip propagation. One strategy is to update all nodes by sending a broadcast message to all nodes with the required join information. The other strategy is to send unicast messages to all senders, as the senders addresses can easily be determined by considering the global table. If the strategies fail it does not matter, as the gossip layer still operates in the usual way.

C. Scalability of MCDC

MCDC is a completely different approach compared to SEM and REUNITE. A key feature of MCDC is to highly optimize multicast calls instead of optimizing multicast forwarding. In this section we will point out when MCDC scales good and when it doesn't.

1) Scaling the number of joined receivers: Scaling the number of joined receivers adds a new node for every receiver to the network and to the global table, which has to be kept in the memory of each node. According to [11] the global table of a 1000-node topology can easily be stored within a few MB of RAM in each node.

2) Scaling the number of sources: As with scaling joined receivers, scaling sources also increases the global table size. Nevertheless MCDC automatically optimizes resource usage by the senders. Frequently addressed multicast groups are assigned a real IPMC address and the fewer frequent groups are handled using unicast. MCDC can determine the current bandwidth usage automatically and adjust the group addressing structure on-the-fly to optimize performance.

3) Scaling the number of routers in a minimal spanning tree connecting all receivers: Scaling the routers does not have an impact on MCDC at all, because MCDC is located only in senders and receivers. As MCDC uses multicast addresses throughout the network it depends on how the routers handle the multicast protocol, but this is not a direct concern of MCDC.

4) Scaling the number of different multicast groups within the same set of receivers: MCDC tackles scaling of multicast group by aggregating groups intelligently. The algorithm first creates logical multicast groups called *Topics*. It tries to group similar multicast groups into *Topics* by analysing traffic reports. Afterwards it sorts the *Topics* by reported traffic size and node count in descending order and assigns highly trafficked *Topics* an IPMC address, while using unicast for the ones at the bottom of the list.

5) Scaling the frequency of join/leave operations: Scaling the frequency of join/leave operations is the current bottleneck of MCDC. With every join/leave operation the global table in every node has to be updated over the network. Due to network latency no consistent state can be reached in a node if join/leave operations occur in less time than it takes for the gossip layer to synchronize at least two nodes. With more and more nodes the gossip layer will take even longer to synchronize and using the alternate strategies to decrease join latencies should be handled with care as that has an impact on overall network utilization.

6) Scaling the frequency of data transmission through multicast: Due to its optimization algorithms MCDC uses the exact capability of the network hardware.

IV. Multicast in Software Defined Networking

Using software defined networking (SDN) we can leverage the approach MCDC (see Section III-A) takes: by centralizing tree optimization. Next to tree optimization an important aspect is to be more fault tolerant and reliable as in IP-based solutions. With SDN it is easy to compute and install multiple redundant and fallback multicast trees.
In this section we will explain a method as seen in [12] to implement an IP multicast network using an OpenFlow Controller. The controller supports multiple trees with fast tree switching, so that the receivers never receive duplicate packets. As we will see, also the packet loss is very minimal when switching the tree. We will also present architectural decisions which boost the overall performance of the controller and enable to react quickly to *join* and *leave* operations.

A. Using Multiple Trees for a Single Multicast Group

A problem by using two multicast trees is that receivers might receive duplicate packets, as distinguishing between the two trees is not a trivial task. Therefore we introduce a tree ID, which is unique for a tree within a single group. All data packets traversing over a certain tree carry the tree ID.
Now we can create different delivery trees and install the flows in the appropriate switches. To switch the delivery tree fast and without packet loss, it is required to install all flows of the new tree, except the one in the switch to which the source is connected. After ensuring that all flows are installed the last flow can be deployed to the source, which then uses the new tree straight away. As the old tree is still intact no packages are lost nor duplicated. After the new tree is in use, the old tree can safely be removed if not in use any more.
This also enables us to prevent a large loss of packets through a single point of failure. Multiple trees can be computed and set up according to the above technique. An occurring failure is being reported to the controller as soon as it is detected. If the failure is not part of the alternative tree, the controller can

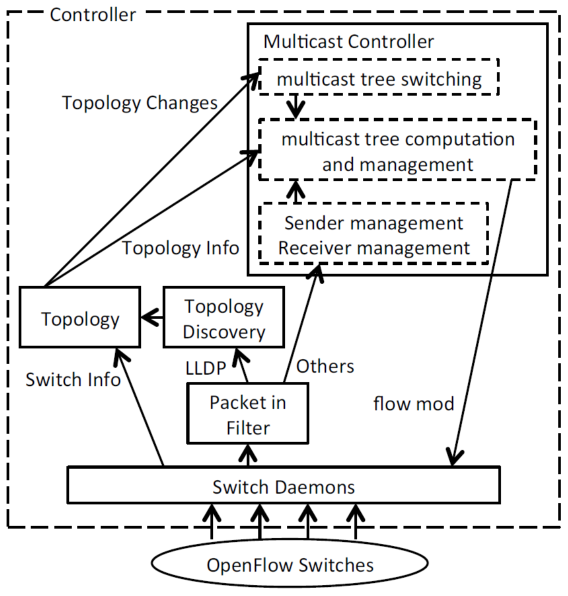


Fig. 7. An architectural overview of the OpenFlow controller. Taken from Ref. [12]

deploy a single flow into the source switch and the new tree works immediately. According to the experiments in [12] it takes few milliseconds for the whole procedure and the packet loss in that time is very low. Depending on the packet rate and the speed of error delivery, results may vary. This also opens an easy way for load balancing at peak traffic, if a switch is overstrained.

B. Controller Architecture

In this section we will explain what the different modules of the OpenFlow controller, as seen in Figure 7, are used for. The Switch Daemons are a simple wrapper to communicate with the networking hardware through the OpenFlow standard. TCP connections to the switches are used to gather relevant information in the controller. Through these the Topology is discovered and all IGMP messages are forwarded to the Sender management and the Receiver management modules of the Multicast Controller. Incoming *join* and *leave* messages trigger the multicast tree computation if the state of a receiver changes.
The multicast tree computation and management module is a highly sophisticated module for efficient tree construction. The key idea is to cache constructed trees, as constructing is an expensive task. Dijkstra's algorithm has to create a tree covering all the switches regardless of whether they are used or not. Having a cached version of the tree now allows to mark each switch as *used* or *unused*. When a group membership change occurs, the switches in which changes have to be applied can easily be determined and the module only deploys a patch rather than redeploying the whole tree. This reduces required control messages, minimizes packet loss and is quicker.
As seen in Fig. 8 a) the Controller stores a cached tree with

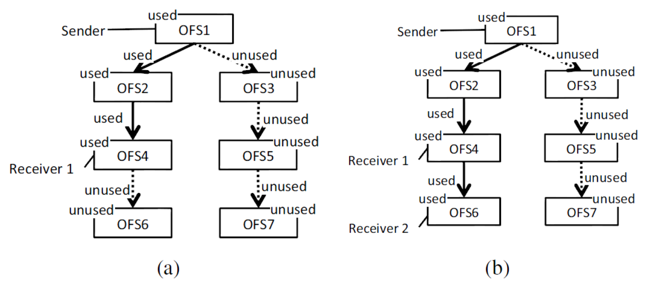


Fig. 8. Visualizes how the multicast tree computation and management module determines the patch to update the flows in the network using the cached tree. (a) One receiver at OFS4 (b) A second receiver at OFS6. Taken from Ref. [12]

Receiver 1 joined to the multicast group. When a join message of Receiver 2, connected to OFS6, reaches the Controller it can quickly determine the required flows to establish by starting at OFS6 and following the tree branch to the root node until a used switch is found. As the link between OFS4 and OFS6 is marked as unused, the controller knows to install a forwarding flow in OFS4 and to install a flow in OFS6 which forwards the multicast messages to the local network of OFS6, where Receiver 2 is located in. After the patch is installed, the Controller marks the link between OFS4 and OFS6 and OFS6 itself as used for further calculations as seen in Fig. 8 b).

The multicast tree switching module is enhancing the reliability of the network. Using the multicast tree computation it ensures to have an installed fall-back tree. As soon as the topology reports a port or link down it ensures that the fall-back tree is deployed. Therefore it determines the affected groups and checks if the fall-back tree is not affected by the problem. If so, it installs the new according flows in the switches to which the senders are connected to.

To further optimize message delivery, the controller assigns an Ethernet address to each tree. The address is of the following format: 03:00:01:00:[First octet of multicast address]:[tree ID]. The sender continues to send messages to the multicast address and the switch where the sender is connected to rewrites the packets Ethernet destination address. The packet can now be transmitted through the network with maximum performance, as very little lookups have to be made for Ethernet address routing. The last switch before the sender reverts the changes and sends the multicast packet to its local subnet.

C. Scalability of the SDN Approach

Compared to MCDC, SDN has the great advantage of being more centralized. Retrieving metrics of the whole topology is included in the OpenFlow standard. Therefore it is not necessary for the SDN approach to implement a control plane that keeps gathering information, as it is already present in an efficient manner. Moreover SDN does not have the problem of synchronizing the global state within all the nodes, as trees are computed and maintained centrally by the controller. In SDN, routing tables can easily be customized to express the intention of the controller in a perfect way. With former network hardware the application had to instead adjust their messages to how the network behaves as the application had no influence on it. This is the main reason why the scalability of MCDC and the SDN approach differ a lot, as we will see in the following considerations.

1) Scaling the number of joined receivers: Scaling the number of joined receivers depends on how good they can be aggregated by the controller. Each switch that sends to a receiver has to store at least one flow. If we connect multiple receivers to one switch, it still has to store one flow only, as the switch can use IPMC to deliver the packet to all connected receivers.

2) Scaling the number of sources: As with the number of joined receivers, every switch that has a connected multicast source requires one flow entry per group to be installed. If multiple sources are in the subnet, the switch still processes all messages of the senders, as they are destined to an IPMC address.

3) Scaling the number of switches in a minimal spanning tree connecting all receivers: Scaling the number of switches is not a problem at all. For a single group, branching switches require one flow entry for every branch and non-branching switches require one flow to forward incoming messages. As the messages are mapped to Ethernet addresses processing time in the switches is also at a minimum level and cannot be faster than in usual networking topologies.

OpenFlow supports group entries as of version 1.3.0 [13]. Group entries allow us to use a single entry for multicast forwarding in branching routers by using action buckets (a set of ordered actions).

4) Scaling the number of different multicast groups within the same set of receivers: Every multicast group has at least one designated logical tree in the network. Having n groups within the same set of receivers requires n times as much flow entries as for a single group (assuming the tree generation algorithm is deterministic). If every group also has a backup tree, the flow count would double (but not necessarily in the same switches). Thus, scalability mainly depends on how many flow entries are supported by the switch (common switches easily support 100.000 flow entries). Overall, flow entries scale linear to the group count.

5) Scaling the frequency of join/leave operations: The presented controller has an optimized way of handling join and leave operations by storing a tree for the whole topology and looking up required branches for the current set of joined hosts. Leaving or joining is then a task of looking at a single branch in a tree. Starting with the node where the join or leave occurred and proceeding to the root. If a *join* occurred, every switch that is marked as *unused* on the way will now be marked as *used* and the appropriate flow will be installed. If a *leave* occurred, every switch that is marked as *used* and has no other receivers left can be marked as *unused* and the flows can safely be removed from the switches. In the worst case for a *join* operation, the controller had to go through all switches in the network (tree degraded to a linear list), but

TABLE I
THIS TABLE GIVES A BRIEF OVERVIEW OF ALL EXAMINED MULTICAST ROUTING APPROACHES IN THIS PAPER, WHEN SCALING THE DIFFERENT SCALABILITY ASPECTS INTRODUCED IN SECTION I-D.

	1) The number of joined receivers	*2) The number of sources*	*3) The number of routers in a minimal spanning tree connecting all receivers*
SEM	The source has to maintain a list of all joined receivers.	SEM uses source-based trees. Having n sources for a group requires maintaining n multicast trees in the MFT of all branching routers.	No entries are kept in non-branching routers, but an entry in the MFT is created for each branch in a branching router.
REUNITE	Group membership of receivers is stored across all routers in the tree.	As packets sent by multiple sources are all tunnelled through the root node, the root node clearly is the bottleneck.	REUNITE stores one entry in the MCT of non-branching routers and an entry in the MFT for every branch in branching routers, but provides a method to cope with overloaded tables.
Dr. Multicast	Every receiver has to keep a synchronized version of the global table. Scaling the receivers exacerbates fast table synchronization.	Every sender has to keep a synchronized version of the global table. Scaling the senders exacerbates fast table synchronization.	No information has to be kept in routers, thus this is totally indifferent to MCDC.
SDN	Needs exactly one group entry in all switches of the spanning tree. The controller maintains group membership.	The switch where the source is connected to needs one additional flow entry for the source.	Branching and non-branching routers need exactly one group entry for multicast operations.

	4) The number of different multicast groups within the same set of receivers	*5) The frequency of join/leave operations*	*6) The frequency of data transmission through multicast*
SEM	SEM does not aggregate forwarding states of different groups and thus scales linear to the number of groups.	After join/leave operations the source sends a *branch* message to rebuild the whole tree. While rebuilding, SEM sends multicast packets through GXcast.	Packets are sent using unicast communication between branching routers, which then replicate the packet for every branch.
REUNITE	REUNITE does not aggregate forwarding states of different groups and thus scales linear to the number of groups.	Join and leave operations only affect the branch below the router in which the group membership state of the affected receiver is kept.	REUNITE uses unicast packets destined to the receivers, which are intercepted by branching routers and replicated for their branches.
Dr. Multicast	MCDC can aggregate multicast groups and uses multicast for large groups and unicast for small groups.	MCDC needs to synchronize the global table in all nodes after join and leave operations which is a very time and bandwidth consuming process.	MCDC optimizes data transmission through its various algorithms and reacts to changing demands on-the-fly.
SDN	The controller does not aggregate any multicast groups, thus we need one additional group entry in every switch for every group.	The controller implements a sophisticated caching mechanism to efficiently recompute trees.	Supports very fast routing due to the mapping to Ethernet addresses. The limit is the bandwidth of the network.

this still remains in linear complexity. If for every switch the number of joined receivers is stored in a way, that the number can be looked up in constant time, this also is true for the *leave* operation.

6) Scaling the frequency of data transmission through multicast: The limit for scaling the data transmission is limited by the bandwidth of used hardware as forwarding in SDN switches is very fast due to the mapping to Ethernet addresses.

V. CONCLUSION

In this paper we examined different approaches to multicast routing concerning their scalability. The presented algorithms were invented to solve a particular problem in a particular use case. While performing good in their intended setup, the algorithms quickly show their downsides when changing the initial constants. Table I show a brief overview of how and why the different algorithms scale for the six scalability aspects defined in section I-D.

SEM and REUNITE are strong and efficient multicast algorithms for sparse networks. They both have a complex tree creation and maintenance algorithm, but are competitive when it comes to data delivery. SEM is a good idea for single-source applications, while REUNITE has a great advantage for multiple-source applications. Furthermore REUNITE can be deployed on a mixed topology for incremental deployment of REUNITE aware routers.

Dr. Multicast is a powerful approach when dealing with lots of overlapping multicast groups in an existing network. MCDC does not even require anything to be installed on routers, instead all senders and receivers use the provided networking library. The drawback is that the global table has to be replicated and synchronized between all nodes. When the group memberships are relatively steady MCDC is a powerful tool for good load balancing and efficient bandwidth usage. Thus MCDC is suited best for data centres in dense networks.

The multicast routing approach using software defined networking is a highly scalable one. While SDN is still a new technology the overall performance is magnificent. The largest drawback is the hardware required to implement a network with. But when planning a data center or even planning an upgrade this definitely is the approach that scales best. With OpenFlow Switch Version 1.3.0 [13] it is even possible to use only one group entry for multicast forwarding of packets. As the OpenFlow Controller is still under active development future versions might enable even better scalability, because the current Controller already leverages MCDC but does not provide forwarding state aggregation.

A direction for future works is aggregating various multicast groups in the OpenFlow Controller and using the same forwarding tree for multiple groups.

REFERENCES

[1] Deering, S.: Host extensions for IP multicasting. RFC-1112 (1989)

[2] Hinden, R.; Nokia; Deering, S.; Cisco Systems: IP Version 6 Addressing Architecture. RFC-4291 (2006) 5

[3] Ballardie, A.: Core Based Trees (CBT) Multicast Routing Architecture. RFC-2201 (1997)

[4] Blazević, L.; Boudec, J.Y.: Distributed Core Multicast (DCM): a multicast routing protocol for many groups with few receivers. Newsletter ACM SIGCOMM Computer Communication Review, Volume 29 Issue 5, October 1999, 6-21.

[5] Wong, T.; Katz, R.: An Analysis of Multicast Forwarding State Scalability. Network Protocols, 2000. Proceedings. 2000 International Conference on (2000) 105-115.

[6] Song, S.; Zhang, Z.L.; Choi, B.Y.; Du, D.: Protocol Independent Multicast Group Aggregation Scheme for the Global Area Multicast. Global Telecommunications Conference, 2000. GLOBECOM '00. IEEE (Volume:1) 370-375.

[7] Minoli, D.: Multicast Addressing for Payload. In: IP Multicast with Applications to IPTV and Mobile DVB-H pp. 26-38. Wiley-IEEE Press, ISBN: 9780470260876. (2008)

[8] Stoica, I.; Eugene Ng, T.S.; Zhang, Hui: REUNITE: A Recursive Unicast Approach to Multicast. INFOCOM 2000, Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings, IEEE (Volume:3) (2000) 1644-1653.

[9] Boudani, A.; Cousin, B.: An hybrid explicit multicast/recursive unicast approach for multicast routing. Journal Computer Communications, Volume 28 Issue 16, October 2005, 1814-1834.

[10] Cain, B.; Cereva Networks; Deering, S.; Kouvelas, I.; Cisco Systems; Fenner, B.; AT&T Labs - Research; Thyagarajan, A.; Ericsson: Internet Group Management Protocol, Version 3. RFC-3376 (2002)

[11] Vigfusson, Y.; Abu-Libdeh, H.; Balakrishnan, M.; Birman, K.; Tock, Y.: Dr. Multicast: Rx for Data Center Communication Scalability. EuroSys '10 Proceedings of the 5th European conference on Computer systems (2010) 349-362.

[12] Kotani, D.;Suzuki, K.; Shimonishi, H.: A Design and Implementation of OpenFlow Controller Handling IP Multicast with Fast Tree Switching. IEEE/IPSJ 12th International Symposium on Applications and the Internet (2012) 60-67.

[13] Open Networking Foundation: OpenFlow Switch Specification, Version 1.3.0, June 25, 2012.