(Deployable) Reduction of Multicast State with In-packet Bloom Filters

Petri Jokela, Heikki Mahkonen
Ericsson Research, NomadicLab
Kirkkonummi, Finland

Christian Esteve Rothenberg
University of Campinas (UNICAMP)
Campinas, São Paulo, Brazil

Jörg Ott
Aalto University Comnet
Espoo, Finland

***Abstract*—Recent developments in networking technology have enabled massive media distribution in the Internet. However, bandwidth is still a limited resource and unicast-based media distribution from a single source to multiple receivers is inefficient. Multicasting provides traffic replication closer to the receivers allowing more efficient data distribution.**

IP multicast can be used for distributing data streams in IP networks, but the bandwidth saving comes at the cost of increased state in the network routers. The amount of state is directly dependent on the number of multicast groups in use. In this paper, we show how in-packet Bloom filter (iBF) multicast can be used to reduce multicast state in the network. The deployment can be done gradually: during the migration phase, a single AS can replace IP multicast with the proposed iBF-based solution without affecting the rest of the network, and take advantage of the reduced state in its core routers.

I. Introduction

Network usage has changed since the dawn of the Internet as substantial advances in networking technology have enabled new and more diverse applications. Today's Internet performance allows transmission of high-quality audio and video streams over the net. In a typical live media transmission case, a stream is delivered to multiple receivers simultaneously. This dissemination mostly happens via massive fan-outs of unicast connections because network layer multicasting is not widely deployed or enabled – beyond possibly intra-provider IPTV streaming or overlay P2P networks [1]. This makes the media delivery inefficient from the network point of view since the data has to be sent repeatedly over the same links. The need for an efficient multicast solution is obvious. In the current Internet, multicast can be implemented either at the IP layer with IP multicast [2], or with different kinds of application layer solutions [3], such as P2P overlays [1]. Some future networking architectures, e.g., PURSUIT [4], are also considering multicast as the elementary scheme for data transmission.

Application layer multicast protocols [3] use unicast connections to deliver traffic across routers supporting application layer multicast. At each such multicast router, data packets are replicated at the application layer if necessary, and sent to the next hop router(s) over separate unicast connections, in essence forming an overlay somewhat similar to the early Multicast backbone (Mbone). IP multicast [2], in contrast, replicates the data at the branching routers at the IP layer. This mechanism is faster than application layer multicast, but requires on-path router support. While IP multicast establishes state for each group in all routers spanning the multicast tree, application layer multicast limits state creation to a subset of routers.

LIPSIN [5] proposes a new packet forwarding mechanism, which is not dependent on a global addressing and routing scheme. Packet forwarding is based on source routing, using on-path link identifiers. To save space and provide a constant length header for packets, link identifiers are compressed using Bloom filters [6]. To take a forwarding decision, a router checks which (if any) of its outgoing links are included in the Bloom filter in the packet. Since the routing information is stored in the packet instead of the routers, the routers can be considered to be nearly stateless.

In this paper, we propose an incremental deployment scenario for the in-packet Bloom filters (iBF). The main idea is to replace IP multicast with iBF forwarding, e.g., inside a single domain managed by one operator. The iBF multicast is transparent to IP multicast in the neighboring networks. In contrast to commercial P2P tree-based or mesh-pull streaming services [1], iBF multicast is delivered as an in-network data distribution service. We investigate three alternative approaches, utilizing the same basic iBF forwarding, but with slightly different mechanisms to maintain the state.

We carry out extensive simulations and compare the results with IP multicast in terms of state requirements. We also examine the forwarding efficiency of iBF: since Bloom filters are probabilistic data structures and may yield false positives, additional packet replicas may be created in the network. Our simulation results show that the proposed solutions make more efficient usage of the network resources than IP multicast.

The rest of the paper is organized as follows: In Section 2, we describe the multicast protocols related to this work. In Section 3, we present our solution for reducing state in the network and Section 4 evaluates the solution. Section 5 concludes the paper and gives future work directions.

II. Background

Numerous multicast solutions were proposed in the past, all of them aiming at utilizing network resources more efficiently. It is hard to achieve optimal performance, because the network has originally been designed for unicast communication. In this background section, we limit the discussion to those protocols closely relevant for our work.

A. IP Multicast

Deering introduced the idea of IP multicast in the late 1980s [2]. IP multicast replicates packets at the IP layer according to multicast routing tables without the need for higher layer interactions. The forwarding procedure is also relatively fast. Further multicast routing protocols such as DVMRP [7] and MOSPF [8] were developed early on to populate multicast routing tables.

Protocol Independent Multicast (PIM) is a multicast routing protocol family with multiple operation modes, of which PIM Sparse Mode (SM) [9] is the most popular today. In PIM-SM, the multicast tree is built based on requests from the clients willing to receive multicast traffic. Clients send *join* requests towards a rendezvous point for the group to which the sources also register. The routers on the path set up the required forwarding state for delivering data packets towards the clients. When a client wants to leave the multicast group, it sends a *leave* message towards the source. Routers on the path remove the corresponding multicast group state and forward the message further, if the router does not have any receivers left for that multicast group.

Originally, IP multicast groups were identified solely using a group identifier G. This mode is called *Any Source Multicast (ASM)*, because all nodes can send data to that group. The data is delivered to all clients that joined that multicast group. The group identifier is denoted with $(*, G)$, where the asterisk stands as a wildcard for any source. In this case, G must be globally unique (even though *scoped* identifiers exist [10]).

In *Source Specific Multicast (SSM)* [11], the group identifier is created from the IP address of the source node (S) and a Group ID (G), where the G identifies the group at the source node. The multicast group is denoted with (S, G) as a globally unique identifier for the group. Using the source IP address as part of the identifier prevents other nodes sending to the same group and provides a natural rendezvous point for join and leave messages. Only packets carrying the correct source and destination group addresses (S, G) are forwarded in the network.

In our work, we choose PIM-SM with SSM as the IP multicast solution, to which we compare the in-packet Bloom filter based solution.

B. In-packet Bloom Filter Forwarding

IP routing is based on maintaining state information in the network routers. Each router has a routing table, describing the next hop for each of the incoming packets. The growth of the IP routing tables [12] has created a need for solutions, reducing the required amount of routing information.

One alternative for routing packets in the network is to use source routing. The sending host determines the path that the packet must take and inserts the list of the nodes on the path in the packet header. This mechanism moves some of the forwarding state to the packet itself, and, at the same time, makes the packet header longer. [1]

Bloom filters can be used to store information in a small, fixed-size space. This property, combined with a source routing mechanism, creates interesting possibilities for packet routing. In the following, we present the basic iBF based forwarding idea, utilizing Bloom filters and source routing based upon link identifiers.

1) Bloom Filter: A Bloom filter (BF) is a probabilistic data structure. It is used to store membership information of data items in a fixed-size filter. A BF is an *m*-bit long bit-string, where all bits are initially set to zero. Inserting a data item into the filter, starts by hashing the data with *k* different hash functions, resulting in *k* index values in the range [0..m-1]. Each of the indexed *k* bits are set to one in the BF. The procedure is repeated for all data items in the set. If the bit value at position *k* of the filter is already one, it remains unchanged.

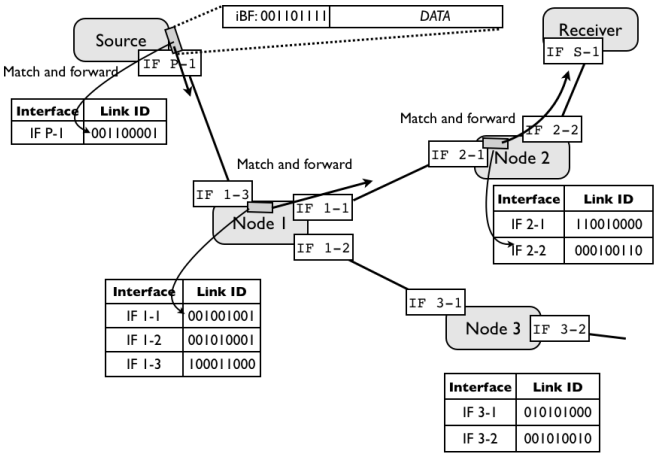


Fig. 1. In-packet Bloom filter forwarding [5]

Once all the items from the set have been inserted into the filter, it can be used for verifying if a certain data item belongs to the set. The verified data item is hashed with the same *k* different hash functions, resulting in *k* values. If all the Bloom filter positions, indexed by these calculated values, are one, the data item is assumed to be in the set. If any of the bits are zero the data item is not part of the set.

The probabilistic nature affects the result accuracy. The verification operation may give a positive answer without a correct match: a *false positive*. The more elements the filter contains, the higher the probability of false positives becomes. Depending on the usage of the filter, this may be a manageable situation. *False negatives* cannot occur with Bloom filters, i.e., if the verification returns a negative result, the item does not belong to the set.

A Bloom filter does not contain any information about the number of inserted data items; a bit set to one at some location can belong to several inserted items. This makes it impossible to remove items from the filter. Since the introduction of Bloom filters, various enhancements have been proposed for it, e.g. for removing items. One of those is the counting Bloom filter [13]. In a counting Bloom filter, each position in the filter is actually represented by a counter instead of a single bit. When an item is added to the filter, the counter value is increased at each of the *k* index points. During the verification process, all index locations having a value greater than zero, are considered to be one. When an item is removed, each of the indexed *k* counters is decreased by one. If the value goes to zero, the corresponding bit in the Bloom filter becomes zero.

2) In-packet Bloom Filters: This probabilistic multicast technique [5] has been proposed as a forwarding fabric for the Information Centric Networking (ICN) architecture in the PURSUIT [4] project. iBF enables fast forwarding decisions in routers and requires no per-flow state in the network.

The iBF forwarding mechanism is not tied to ICN principles and can also be used in other network environments. For example, Multiprotocol Stateless Switching (MPSS) [14] introduces iBF forwarding in the MPLS network, simplifying the forwarding, especially in the multicast case. In optical

[1] It also requires the sender to obtain knowledge of the entire path to the receiver.

routers, iBF can be used to achieve more efficient packet forwarding directly on the optical layer, without the need for Optical-Electric-Optical (O-E-O) conversion [15].

iBF forwarding does not rely on globally routable addresses. Instead, all links between two nodes are identified using a Link Identifier (LID). A LID is an *m*-bit long bit string, with k bits set to one, where $k \ll m$. Typically, *m*=256 and *k*=5. This makes the LID similar to a data item in Bloom filters, where the k different hash functions have been run over the data and the corresponding index values have been generated. In our simulations, the LIDs are generated randomly. It is possible to achieve better results by optimizing the LID values in the network, but the larger the topology is, the harder this is to implement.

The iBF packet forwarding is based on source routing, where all the required forwarding information is included in the header of the packet. A multicast tree consists of a set of interfaces, that the packet needs to pass. All the LIDs of the links forming a multicast tree are collected and combined in the iBF by using logical OR operations. This is similar to inserting a new data item in the Bloom filter.

In the proposed forwarding fabric [5], the Topology Manager (TM) is responsible for collecting and maintaining the network topology information. The TM can be implemented, for example, based on similar principles as OSPF, or by using a centralized manager, such as PCE [16]. The TM creates a delivery tree after receiving a request and calculates the corresponding iBF. The iBF is further delivered to the data source node, which sets the iBF to the data packet header and sends the packet to the network.

The Topology Manager can be replaced with other mechanisms, such as collecting the forwarding iBF en route. In a (multicast join request) packet, a new *m*-bit field is added and initialized to zero. Each router on the path adds the LID of the interface from where the packet arrived to the collector field by logically ORing it with its content. Once the packet arrives to the destination, the collected field has reverse path iBF that can be used for data delivery.

Figure 1 shows the packet forwarding process in the iBF network. When a packet arrives to an iBF router, it verifies all its LIDs on its outgoing interfaces with the iBF in the packet header. The router logically ANDs the LID with the iBF, and if the result equals the LID, the packet is sent out from that interface. Multicast is an inherent property of iBF forwarding, and is implemented simply by adding LIDs of multiple outgoing interfaces of a single router to the iBF. When the router makes the forwarding decision, it sends the packet out on all the matching interfaces.

As discussed in the Bloom filter section, the verification process may return a false positive answer. If this happens during matching of interface LIDs, the packet is falsely forwarded out of the corresponding interface. In our simulations, we calculate the false routing decisions and determine the forwarding efficiency value. The forwarding efficiency describes the share of data transmission over links, that are needed for the data to reach all the destinations, compared to the total number of transmissions over links. I.e. if the forwarding efficiency is 97%, it means that 3% of the transmissions over links are due to false positives and are not required for data delivery.

III. IBF Migration Scenario

Replacing the old Internet forwarding mechanism with a new one is challenging. When the new system requires changes on end-hosts as well as on all the routers, the deployment burden is high. Internet-wide deployment overnight is impossible, leaving incremental deployment as the only way forward. Such gradual deployment must allow adopting operators to take advantage of the new technology, without affecting packet forwarding in the rest of the Internet.

A. Preconditions

In this paper, we propose replacing IP multicast partially with iBF, within a single domain. The purpose is to offer some immediate benefit to the deploying operator compared to using IP multicast. The iBF multicast area must be transparent to the surrounding IP multicast in the neighboring domains. The edge nodes must handle the IP multicast signaling messages and perform correct conversions between the different types of networks.

Instead of using a Topology Manager (as one would in a clean-slate environment), we take advantage of the IP multicast-based signaling also for controlling operation in the iBF network. We simply piggyback the additional control information for the iBF network on the IP multicast control messages. This has some disadvantages compared to a Topology Manager-based solution: firstly, IP routing does not necessarily yield optimal routing and hence no optimization for the iBF forwarding tree and, secondly, we have to assume that IP routing is stable and delivers the signaling messages always using the same route, so that *join* and *leave* operations can be used to properly manage iBF delivery trees. We note that this choice only adds piggybacking to control messages and does not affect the actual multicast data delivery.

To estimate the amount of state in a real world scenario, we use the Shoutcast web site as an example. Shoutcast.com [17] provides access to different Internet radio stations. In addition, the site lists the number of listeners for each of the stations. Figure 2 shows a snapshot of a number of Internet radio station listeners with the number of stations grouped with resolution of hundred listeners. We observe that most of the radio stations have only a small number of listeners.

If all the stations were delivered using IP multicast, the amount of state would be high due to the number of different groups. In other words, there would be a lot of state in the network, but each added piece of state would serve only a few listeners. With large radio stations, the situation would be different since it is more likely that added state would serve more clients: in this case, multicasting would offset its cost in terms of state by eliminating lots of redundant data transmissions from the network that would have been required for unicast delivery.

In contrast, iBF packet forwarding provides the best performance when the number of receivers is not very large. Typically, the more receivers a multicast group has—and thus the more LIDs that have been inserted in the forwarding iBF—the more false positive routing decisions that are made in the network. Therefore, the efficiency is lower when the number of destinations per group increases.

B. Replacing IP Multicast with iBF

1) Network setup: Figure 3 depicts the basic scenario for placing an iBF forwarding network between IP multicast

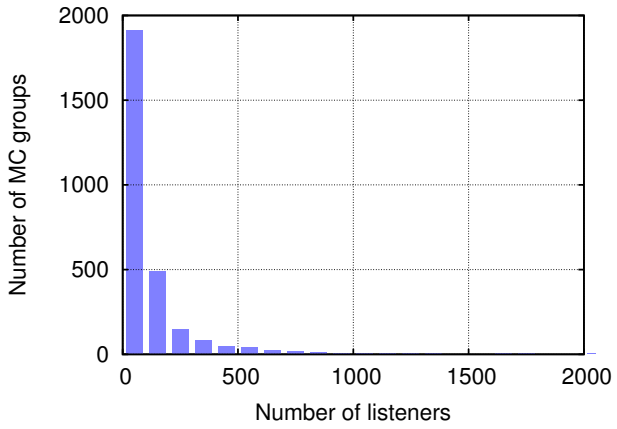


Fig. 2. Distribution of the number of listeners (receivers) per radio stations (modeled as multicast groups) of the popular Shoutcast.com streaming service [17] (May 25, 2012)

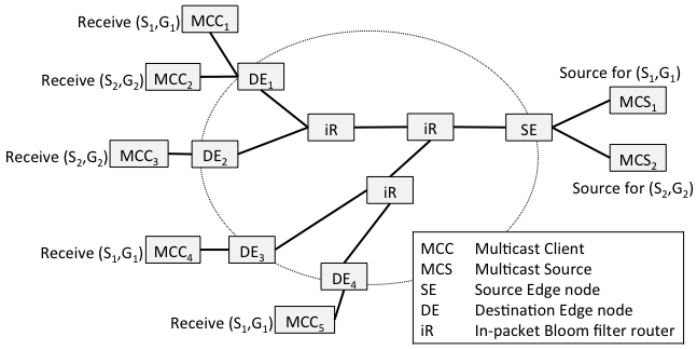


Fig. 3. IP multicast with iBF support

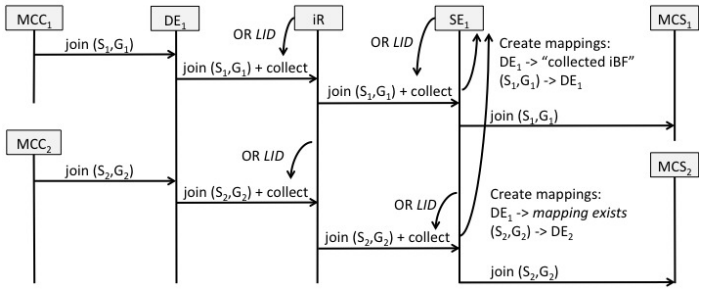


Fig. 4. Creating iBFs

a)

IP Multicast group	iBF of the tree
(S_1, G_1)	$iBF_{S1,G1} = iBF_1 \mid iBF_3 \mid iBF_4$
(S_2, G_2)	$iBF_{S2,G2} = iBF_1 \mid iBF_2$

b)

IP Multicast group	Destination Edges	iBF of the tree
(S_1, G_1)	DE_1, DE_3, DE_4	$iBF_1 \mid iBF_3 \mid iBF_4$
(S_2, G_2)	DE_1, DE_2	

Destination Edge	iBF to the DE
DE_1	iBF_1
DE_2	iBF_2
DE_3	iBF_3
DE_4	iBF_4

Fig. 5. iBF mappings at the SE node

networks. The iBF network is connected to the IP multicast networks via gateway nodes. These gateway nodes can act either in Source Edge (SE) and Destination Edge (DE) roles or in both roles simultaneously. The SE function resides on the edge, behind which the IP multicast source is located. Similarly, the DE function operates on the client side edges.

The SE function is responsible for mapping incoming IP multicast traffic to the iBF network traffic and delivering packets further to the iBF network. It maintains a mapping between each IP multicast group (S, G) and a corresponding iBF, defining the delivery tree for that particular group inside the iBF network. At the SE, the IP multicast group (S, G) defines unambiguously the tree on the iBF side. The iBF is added to the packet header, keeping the original packet intact. The DE node is responsible for removing the iBF header, and sending the traffic from the iBF network into the IP multicast network. The SE and DE functionality can also be in core routers to support IP multicast clients or servers in the local network.

2) Joining the multicast tree: Figure 4 depicts the procedure for creating the iBF for relaying traffic between the SE and DE routers. When an IP multicast *join* message for multicast group (S_1, G_1) arrives at the DE router, the router adds two additional iBF fields in the header: *iBF collector* and *DE identifier*. The DE sends the *join* message further to the next-hop iBF-capable router. The procedure is repeated for each multicast group. If additional *joins* arrive at the DE node for the same multicast group, the IP multicast function at the DE notices this and does not send the *join* message further.

The *join* message is routed through the iBF network using IP routing. Each iBF router on the path adds the LID of the receiving interface to the *iBF collector* field in the header. The LID is added by logically ORing it with the existing content of the field.

Finally, the *join* message arrives at the SE node. The SE removes the added iBF fields from the header. The SE uses this information for mapping the (S, G) multicast group to the multicast tree on the iBF network. Finally, the SE sends the *join* message further towards the actual multicast source node in the legacy IP network.

The SE can store (S, G) to iBF multicast tree mappings in different ways. As described in Section II-B1, it is not possible to remove items from a basic Bloom filter. This feature is, however, required for iBF multicast trees: there must be a possibility to remove destinations from the tree when clients leave the multicast group. This affects the design of the SE node tables, for which we present three alternative ways of implementation:

Tree-based iBFs: The SE maintains a single multicast group (S, G) to iBF mapping (see Figure 5a). All individual iBFs received in the *join* messages from the DE nodes and

containing the path information from the SE to the DE are logically ORed into the corresponding iBF. Thus, when a new DE sends a *join* message, a new path is added to the (S, G) mapping. As this solution does not allow removing DEs from the tree directly, we describe an alternative way for the leave process in the next subsection.

Path-based iBFs: When the SE receives a *join* message from a DE, it creates a mapping from the DE identifier to the path iBF, leading from the SE to the DE (Figure 5b). If a previous *join* message has arrived from the same DE for another multicast group, the corresponding mapping already exists, and a new one is not needed.

To create the iBF trees for each multicast group, another mapping table is required. This table contains the mappings from each (S, G) group to all the DE nodes that have joined the multicast group. The SE can calculate the iBF for the tree using the corresponding DE to iBF mappings whenever the set of DEs changes.

Counting iBFs: Using counting Bloom filter at the SE node allows removing entries from the tree iBF. This solution is similar to the *Tree-based iBF* solution, but each of the Bloom filter bits is in fact represented by a counter. Instead of just ORing the incoming iBF to the existing iBF of the tree, for each location where the incoming iBF has a bit set to one, the counter value of the corresponding bit in the iBF of the iBF tree in incremented.

A Counting iBF does not add to the amount of state at nodes inside the network nor in the packet headers, but the required space for the state at the SE node increases. For example, using a 4-bit cell configuration [18] would increase the required space for the iBF $m = 256$ from 256 bits to 1 Kbit.

3) Leaving the multicast tree: The DE has to be removed from the SE's mapping table, when all IP clients behind it have left the group. Once a DE leaves, the SE has to recalculate the iBF using information from the active DE nodes. We first present two alternative ways to do this with a *tree-based iBF*.

To inform the SE node that it has no more listening clients, a DE node forwards the last IP multicast *leave* message to the SE node, which now has to recreate the iBF without that DE node. The SE node sends a request message, requesting the DEs to re-join for the (S, G), using the iBF that is mapped to the (S, G) for which the *leave* message arrived. The request is thus delivered to all DE nodes associated with the (S, G). All DEs, with active receivers will respond with a *join* message, collecting the corresponding path iBF. Once the *joins* have arrived at the SE, it can recalculate the iBF for the active DEs and send further data using the updated iBF.

Another alternative is to allow the DE nodes to send the *join* messages periodically for all the active (S, G) groups. Thus, the SE can recalculate the iBF for the iBF tree periodically. There is no need for the DEs to send any *joins* for updated iBF information.

In all cases, the SE has to receive a *leave* message from all DEs when they do not have clients any longer. Once all the DEs have left, the SE has to send the *leave* towards the IP multicast source to inform the source that it does not want to receive multicast traffic for that group any more.

For *path-based iBFs*, the removal process is simpler. When the *leave* message arrives from the DE node, the SE can remove the DE from the mapping table for that particular (S, G) multicast group (Figure 5) and recalculate the iBF for the tree. There is no need to update the SE to DE iBF mapping table.

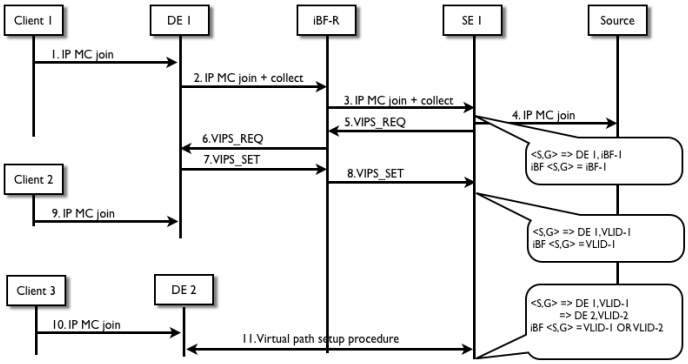


Fig. 6. Virtual Path Setup

The leave process is simple also for the *counting iBF* solution. An iBF collector field is added to the *leave* message, and the iBF path is collected, just like with the *join* message, when the packet is delivered to the SE. The receiving SE removes the collected iBF from the corresponding (S,G) mapping table by decrementing each counter for which which the collected iBF has a one in the bit field.

4) Data delivery: For the incoming multicast traffic, the SE retrieves the active iBF for the iBF tree and encapsulates the packet with the iBF header. Using the basic iBF forwarding described in Section II, the data packet is delivered to the destination DE nodes. Each DE decapsulates the packet and sends it towards the clients. If the packet arrives at the DE due to a false positive, there is no state for that (S, G) pair at the DE node, and the data packet is discarded.

C. Virtual paths

As an attempt to reduce the number of false positive forwarding decisions, we utilize the virtual tree concept, defined in [5]. A virtual tree is a multicast delivery tree, that has a unique Virtual LID (VLID) value, which is configured on each of the interfaces along the delivery tree. Adding a virtual tree adds one piece of state in each of these interfaces.

Creating optimal virtual trees in the network is not trivial. When new trees are introduced, the existing trees should be verified and new trees created so that the state increase is minimal. Instead of using full trees, we establish a virtual tree only between two edge nodes. We refer to these trees as virtual paths. Each virtual path adds one piece of state in each router on the path. Only one virtual path VLID is needed for each SE–DE pair, and the same VLID can be reused for all multicast groups using that path.

Figure 6 shows the virtual path setup process. The SE can initiate the process after receiving a *join* message by sending a VIPS_REQ message to the DE (Fig 6, messages 5 and 6). The DE creates a VLID for the virtual path, prepares a VIPS_SET message, and sends it back to the SE (message 7). Each router on the path sets the VLID as an additional link identifier on the incoming interface (receivers of messages 7 and 8). Once completed, the virtual path between SE and DE has been established.

IV. Evaluation

We evaluated the proposed iBF migration solution using the ns-3 network simulator [19]. During the simulations, we concentrated on the amount of state in the network nodes. Each piece of state requires a different size of storage, depending, e.g., on the address size. Thus, we also estimated the required amount of memory in different scenarios.

A. Network Node State

The evaluated solutions behave differently and create the needed state in different nodes. For example, the tree-based iBF solution creates state only on the edge routers, while IP multicast needs state in every router belonging to the delivery tree. Thus, it is not feasible to focus on the amount of state on some particular node. Instead, we calculate the total state maintained in the network. In the simulations, we count each piece of state as one, not considering the bytes consumed.

In this section, we present the theoretical basis for calculating the amount of state in the network nodes. In the equations, we see the various aspects of the method under investigation. The equations do not count any state for the outgoing traffic at the DE function because the required state is the same for all the solutions: a DE requires only the IP multicast state needed to deliver packets to the next hop IP multicast nodes in neighboring networks.

Equation 1 shows the state count for IP multicast. In the equation, g is the number of multicast groups and t_i is the number of core routers that are part of the delivery tree, excluding the edge routers. The first term in the equation gives the core router state and g is directly the amount of state in the SE nodes.

$$State_{Multicast} = \sum_{i=1}^{g} t_i + g \tag{1}$$

For path-based and tree-based iBF multicast, the forwarding nodes do not keep any additional state. For tree-based iBFs, only a single mapping between the incoming IP multicast group and the outgoing iBF is needed on the SE routers (see Figure 5a.). Equation 2 gives the amount of state for this solution, and the only term is the number of groups.

$$State_{iBF-tree} = g \tag{2}$$

For path-based iBFs, the situation is slightly more complex, as was shown in Fig. 5b. Equation 3 shows the state requirement for path-based iBFs. The first term describes the number of mappings between (S, G) and DEs. The second term, in turn, gives the amount of state at the SE routers for the DE to iBF mappings. E_{D_i} is the number of DE nodes at the SE node i, including all the groups, c_i is the number of destination edge routers for group i, and E_S is the number of active SE nodes.

$$State_{iBF-path} = \sum_{i=1}^{g} c_i + \sum_{i=1}^{E_s} E_{D_i} \tag{3}$$

Finally, equation 4 shows the amount of state for the virtual path enhanced iBF multicast. The only difference to the path-based iBF is the third term in the equation, describing the nodes having the VLID set. p_{ij} gives the number of routers on the path from $\mathrm{SE_i}$ to $\mathrm{DE_j}$.

$$State_{VLID} = \sum_{i=1}^{g} c_i + \sum_{i=1}^{E_s} E_{D_i} + \sum_{i=1}^{E_s} \sum_{j=1}^{E_{D_i}} p_{ij} \tag{4}$$

In tree-based solution, the state saving comes with the cost of added signaling, when a DE is leaving the group. In all cases, the DE must inform the SE about leaving the group, and if that is the last DE leaving, the SE must send the *leave* message to the IP network, towards the source node. However, considering that the DE node leaving does not happen very often, or it can be time limited, the amount of required signaling messages is low compared to, e.g., the transmission of a video stream.

In addition to the amount of state, we calculate the overall state reduction ratio. This indicates the percentage of state reduction relative to IP multicast.

B. Simulator

The iBF packet forwarding simulator was implemented in the PSIRP [20] project for ns-3 [19], version 3.6. We enhanced the original iBF simulator with the functionality specified in Section III.

Due to the missing IP multicast functionality in ns-3, the required functions and packet structures were implemented. To simulate the IP multicast source and receiver nodes, we modified the ns-3 UDP echo server and client software. The client and server applications were handling the sending of *join* messages, as well as the data traffic in the IP based nodes. In all iBF nodes, we implemented the statistics collecting operations for calculating the number of IP multicast groups, VLIDs, as well as false positive routing decisions that were made in the network.

The edge routers are responsible for the conversion between the two forwarding technologies. The IP multicast *join* message is the key signaling message used to create the required forwarding information in the iBF network. Thus, the *join* message handling was implemented at the edge routers, as well as at the iBF forwarding nodes. The edge routers implement the required traffic mapping information functions, and the core routers the required iBF collecting operations.

C. Performed simulations

We selected the Rocketfuel [21] AS6461 topology for evaluating the different multicast solutions. We simulated both the path-based iBF and the virtual path solution. IP multicast state information was collected at the nodes based on the *join* messages passing through the routers, and the required state for the tree-based iBF was calculated.

We used the network with two different setups. 1) We used all the 138 nodes in the topology as the potential edge routers, i.e., all the routers were connected either to external IP networks or they were local receivers or senders of multicast traffic. Note that there is no difference from the simulation point of view whether the SE or DE functionality on a node serves a local IP multicast client or has a complete network behind it. 2) We selected all the nodes with three or less connections as the edge routers.

We placed clients behind the edge routers using approximately the same distribution observed in the shoutcast.com statistics (see Figure 2). The actual distribution for the DE

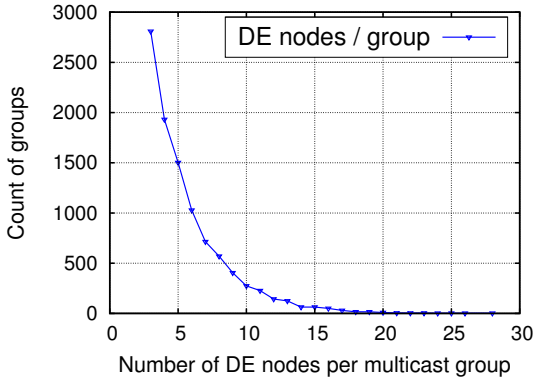


Fig. 7. Number of destination edge nodes per multicast group

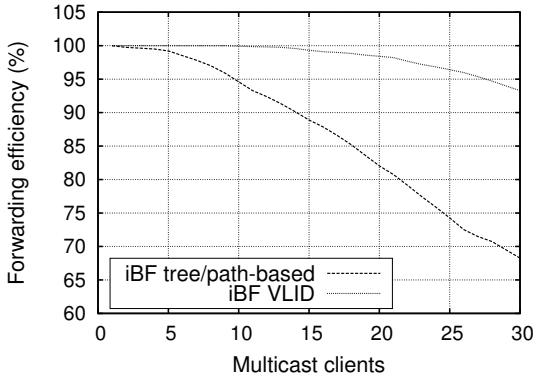


Fig. 10. Forwarding efficiency, 1 server, 1-35 clients

nodes per multicast group is shown in Fig. 7. For the simulation purposes, the number of DE nodes per group is significant. This number defines the branches for each multicast delivery tree inside the iBF network. The number of clients receiving the same multicast group behind a single DE does not affect the signaling or the data transmission inside the iBF network.

We ran the simulations with 10,000 servers, each sending multicast data to one multicast group. During the simulation, the servers were placed one by one, and all the clients for that corresponding multicast group were placed simultaneously. All the clients sent the *join* messages to the multicast group, and the server started to send data to the clients. The network nodes collected the statistics of the network usage.

D. Results

Figure 8(a) shows the amount of state with 26 edge routers. When running the simulation using 10,000 multicast groups, the IP multicast requires more than two times the state than the path-based iBF solution. When compared to the tree-based iBF solution, the amount of IP multicast state is approximately ten times higher. The Virtual Path solution performs slightly worse in the beginning, when the state is added to the core routers. While there are 26 edges, the virtual paths are quickly established between all combinations, after which the amount of state grows in line with the path-based solution.

Figure 8(b) shows the corresponding overall state reduction ratio, compared to IP multicast. The tree-based solution performs best, with a state reduction ratio of over 90%. Both path-based and virtual path solutions get close to a 60% reduction ratio.

Figure 8(c) shows the forwarding efficiency. The iBF (both path-based and tree-based) solutions provide an approximately 96% forwarding efficiency, i.e. roughly four percent of data packets delivered over links are not needed for data delivery, but are forwarded because of a false positive match. However, virtual paths provide nearly 100% efficiency, i.e. the amount of false positive routing decisions is very small.

Figures 9(a), 9(b), and 9(c) show the corresponding results when all the 138 nodes act as edge nodes. In this scenario, there are more potential edge-to-edge connections; Either the network is connected to many neighbors, or the nodes are themselves multicast source or destination nodes.

IP multicast uses roughly nine times more state than tree-based iBF, while the forwarding efficiency is slightly over 95%. Path-based iBF saves roughly 30% of the state compared to IP multicast. However, virtual paths use more state than IP multicast. The reason for this is that all the SE–DE pairs have not yet installed the virtual paths. Looking at the curve in Fig. 9(a), we can see that the saturation point is close, and when more groups are added to the system, the IP multicast state will exceed the virtual paths.

Counting bloom filters have the same amount of state as the tree-based iBFs, but there is no need to do re-joins from the DE nodes when DE nodes are leaving the group. The counter consumes more memory than the plain iBF and the consumption is estimated to be somewhere between IPv4 and IPv6 multicast memory consumption.

E. False Positives in iBF Forwarding

In LIPSIN [5], the authors discuss the false positives and the related forwarding efficiency. They show that the forwarding efficiency gets better when the number of LIDs is increased, and the used iBF is selected from a set of candidate iBFs. The authors claim also that the additional bandwidth used by false positive routing decisions, is not always a disadvantage. For example, ICN networks can take advantage of these additional packet replicas by caching them.

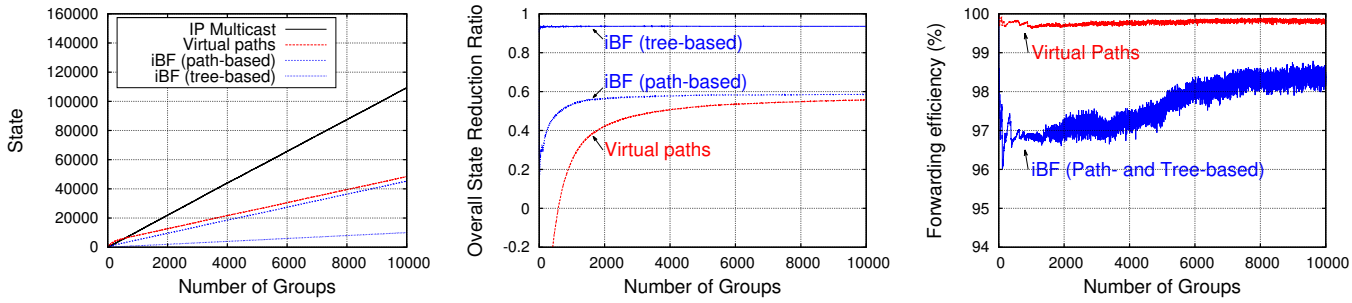
In Switching with in-packet Bloom Filters (SiBF) [22], the authors show that it is possible to forward packets using iBFs with a zero false positive rate in a data center, using only 96-bit iBFs. Their solution performs careful LID creation based upon the MAC addresses of the neighboring nodes.

When comparing pure iBF and Virtual path iBF solutions, we saw in Figures 8(c) and 9(c), that Virtual paths provide better forwarding efficiency when the number of edge routers is small. However, adding edge routers increases the number of VLIDs in the core routers and lowers the forwarding efficiency. Still, the efficiency stays above 95%.

For comparison, we also simulated a single multicast group case, where 35 clients joined one by one, and we calculated the forwarding efficiency for each step. Figure 10 shows the result. We can see that with pure iBF the efficiency drops below 95% when more than ten clients join the group, while with virtual paths this happens only after 27 clients.

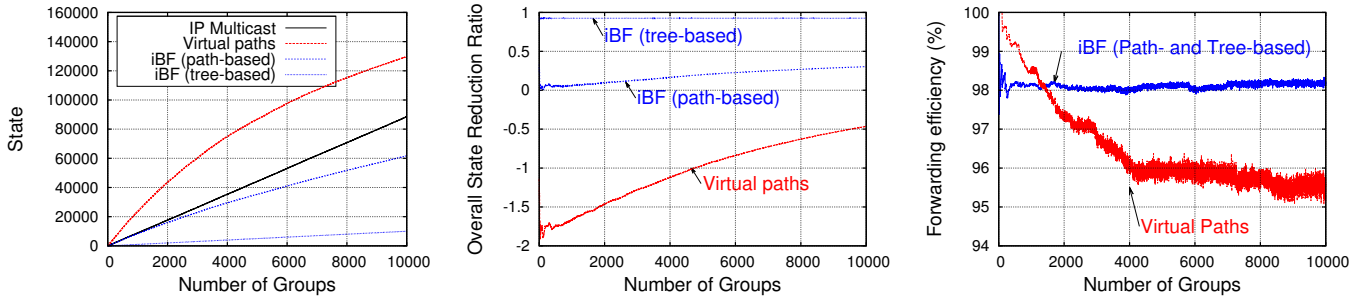
F. Memory Requirement

The evaluated solutions consume different amounts of memory when storing the state information. Table I summarizes the calculated theoretical minimum values for the different solutions. For the IP multicast case, the table shows



(a) State Count (b) Overall state reduction (c) Forwarding efficiency

Fig. 8. Simulation results for 26 edge routers.



(a) State Count (b) Overall state reduction (c) Forwarding efficiency

Fig. 9. Simulation results for 138 edge routers.

values calculated using data structures from the Linux IP multicast implementation. These are given inside parenthesis.

With IP multicast, the group identifier (S, G) size is 64 bits with IPv4, and 256 bits with IPv6. This group ID is mapped to the MAC address of the next hop router. In addition, the router needs to verify the incoming interface before the packet is forwarded to prevent injection attacks. The mandatory information takes $(32 + 32 + 16 + 48) = 128$ bits for IPv4, and $(128 + 128 + 16 + 48) = 320$ bits for IPv6.

Linux IP multicast data structures are defined in the kernel sources (file: `mroute.h`). The actual structure for the group entry takes 368 bits for IPv4 and 560 bits for IPv6. The structures contain some additional information, e.g., byte counters, that are not needed for multicast forwarding.

In tree-based iBF, each multicast group requires space for the multicast group id and the corresponding iBF (Figure 5). For IPv4, the required memory is $(2 \times 32 + 256) = 320$ bits and, for IPv6, $(2 \times 128 + 256) = 512$ bits. Using counting Bloom filters with 4-bit counters, each LID takes 4×256 bits. With IPv4, the total size is 1,088 bits and, with IPv6, 1,280 bits.

The path-based iBF solution state depends both on the total number of incoming multicast groups and on the number of DE nodes per multicast group. In the simulations, we had roughly 46,000 multicast tree end points in the iBF network. On average, there are 4.6 end points per multicast group. The total number can be used to calculate the required memory at the SE nodes. If we use four bytes for the DE node identifier (NId), the total amount of used memory is $46,000 \times 32 + 10,000 \times 2 \times sizeof(IP\ address)$ for the (S, G) to DE list mappings. For the DE to iBF mappings, we need maximum of $(n - 1) \times (NIdsize + iBFsize)$ at all n edge nodes. This results in $137 \times (32 + 256) \times 138 = 0.65$ MB, or $25 \times (32 + 256) \times 26 = 0.02$ MB for the edge router state.

TABLE I. MEMORY REQUIREMENT (MEGABYTES), 10,000 IP MULTICAST GROUPS

	IPv4		IPv6	
	26 DEs	138 DE*10000s	26 DEs	138 DEs
IP Multicast	1.46 (4.80)	1.18 (3.88)	5.21 (7.30)	4.22 (5.91)
iBF: tree	0.38	0.38	0.61	0.61
iBF: counting	1.30	1.30	1.53	1.53
iBF: path	0.57	1.21	0.81	1.43
iBF: Virt. path	0.67	3.28	0.91	3.50

With virtual paths, using a similar kind of SE state arrangement as with the path-based solution, the only additional consumed memory comes from the VLIDs installed on the forwarding nodes in the network. The amount of state in the nodes can be calculated using the total amount of state with virtual paths and the amount of state with path-based solution. This difference yields the required amount of state in the forwarding nodes.

V. CONCLUSIONS AND FUTURE WORK

In this paper, we compared the performance of IP multicast with in-packet Bloom filter based multicast. We evaluated three different iBF-based solutions and used the required amount of state as the evaluation criterion. In addition, we estimated the memory usage of the different solutions.

The tree-based iBF approach provides the best results in terms of required state and memory consumption, at the cost of a negligible amount of signaling. But the signaling does not create a huge burden: First, the size of a signaling message is small compared to the amount of data typically streamed.

Second, the destination edges do not leave a group very often. It is also possible to limit the leaving process to occur, e.g., at most in one minute intervals.

The path-based solution, in turn, adds some state at the SE node, but reduces the required signaling. Still, it is more efficient, both from the state and memory consumption point of view, than the IP multicast solutions.

Virtual paths use more state, but they also provide better forwarding efficiency. The gain of virtual paths becomes clear, when the number of multicast groups transmitted is high and the virtual paths can be used efficiently for multiple groups simultaneously. Our simulations showed that, when the number of edges is large, the number of multicast groups should also be large.

In the simulations, we showed that an iBF-based packet forwarding solution can be used to partially replace IP multicast in the network. The replacement provides enhanced operation in the network, and the deploying operator can benefit from introducing this technology.

IBF-based packet forwarding significantly reduces the amount of state when compared to IP multicast. While iBF causes some false positive forwarding decisions, we can still maintain forwarding efficiency at very high performance levels. The efficiency depends on the number of links encoded in the iBF, and stayed between 95% and 100% in our simulations. Virtual paths may be used to improve maintain this efficiency as the number of group members grows.

Some of the assumptions and selected solutions leave room for improvements. For future work, we can identify a couple of interesting topics:

Using virtual trees may result in better forwarding efficiency and less state than that used in the naive virtual path solution. Implementing the mechanism in a real network environment, would provide us better understanding of the efficiency of the solution.

Instead of using counting Bloom filters for dynamically removing elements from the iBF, we intend to investigate the use of the deletable Bloom filter [23] data structure as a low overhead probabilistic alternative.

After showing the feasibility of an incrementally deployable solution, we are working on a proof of concept prototype implementation based on OpenFlow 1.2. With OpenFlow, the flow matching operations using arbitrary bitmasks on source and destination IPv6 addresses allow operating with 256-bit iBF that can be pushed (and popped) at the source and destination edge nodes.

In our work, we did not consider security issues. As the presented scenario targets deploying iBF in a single AS, the risk of attacks is low. The actual iBF operations are performed completely inside the operator's network. However, the potential threats, especially in multi-domain iBF cases, will be investigated.

Yet another interesting line of work would be to investigate how iBF multicast forwarding may become an ISP-friendly mechanism to minimize traffic stresses for the case of next generation P2P IPTV services [1] delivered in either a tree-push, mesh-pull or push-pull fashion.

ACKNOWLEDGMENT

The authors would like to thank the IFIP Networking reviewers, our shepherd Michael Welzl, David Hayes, and Tero Kauppinen for their comments and suggestions that helped us to improve the paper. The work presented in this paper was supported by the European Union FP7 PURSUIT project under contract FP7-ICT-2010-257217.

REFERENCES

[1] X. Hei, Y. Liu, and K. Ross, "Iptv over p2p streaming networks: the mesh-pull approach," *Communications Magazine, IEEE*, vol. 46, no. 2, pp. 86–92, February 2008.

[2] S. E. Deering and D. R. Cheriton, "Multicast routing in datagram internetworks and extended LANs," *ACM Transactions on Computer Systems*, 1990.

[3] M. Hosseini, D. T. Ahmed, S. Shirmohammadi, and N. D. Georganas, "A survey of application-layer multicast protocols," *IEEE Commun. Surveys and Tutiruals*, 2007.

[4] "EU FP7 project PURSUIT," http://www.fp7-pursuit.eu, accessed: 12.1.2013.

[5] P. Jokela, A. Zahemszky, C. Esteve Rothenberg, S. Arianfar, and P. Nikander, "Lipsin: line speed publish/subscribe inter-networking," in *Proceedings of the ACM SIGCOMM 2009 conference on Data communication*, ser. SIGCOMM '09. New York, NY, USA: ACM, 2009, pp. 195–206.

[6] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Commun. ACM*, vol. 13, no. 7, pp. 422–426, Jul. 1970.

[7] D. Waitzman, C. Partridge, and S. Deering, "Distance Vector Multicast Routing Protocol," RFC 1075 (Experimental), Internet Engineering Task Force, Nov. 1988.

[8] J. Moy, "Multicast Extensions to OSPF," RFC 1584 (Historic), Internet Engineering Task Force, Mar. 1994.

[9] B. Fenner, M. Handley, H. Holbrook, and I. Kouvelas, "Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised)," RFC 4601 (Proposed Standard), Internet Engineering Task Force, Aug. 2006, updated by RFCs 5059, 5796, 6226.

[10] D. Meyer, "Administratively Scoped IP Multicast," RFC 2365 (Best Current Practice), Internet Engineering Task Force, Jul. 1998.

[11] H. Holbrook and B. Cain, "Source-Specific Multicast for IP," RFC 4607 (Proposed Standard), Internet Engineering Task Force, Aug. 2006.

[12] D. Meyer, L. Zhang, and K. Fall, "Report from the IAB Workshop on Routing and Addressing," RFC 4984 (Informational), Internet Engineering Task Force, Sep. 2007.

[13] L. Fan, P. Cao, J. Almeida, and A. Z. Broder, "Summary cache: a scalable wide-area web cache sharing protocol," *IEEE/ACM Trans. Netw.*, vol. 8, no. 3, pp. 281–293, Jun. 2000.

[14] A. Zahemszky, P. Jokela, M. Särelä, S. Ruponen, J. Kempf, and P. Nikander, "MPSS: Multiprotocol Stateless Switching," in *Global Internet Symposium 2010*, 2010.

[15] M. AL-Naday, J. Almeida, R.C., K. Guild, and M. Reed, "Design proposal of a photonic multicast bloom filter node," *Photonic Network Communications*, vol. 24, pp. 132–137, 2012.

[16] A. Farrel, J.-P. Vasseur, and J. Ash, "A Path Computation Element (PCE)-Based Architecture," RFC 4655 (Informational), Internet Engineering Task Force, Aug. 2006.

[17] "Free internet radio stations," http://shoutcast.com, accessed: 12.1.2013.

[18] S. Tarkoma, E. R. C., and E. Lagerspetz, "Theory and practice of bloom filters for distributed systems," *IEEE Communications Surveys and Tutorials*, vol. 14, no. 1, 2012.

[19] T. R. Henderson, M. Lacage, G. F. Riley, C. Dowell, and J. B. Kopena, "Network simulations with the ns-3 simulator," SIGCOMM'08 Demos, 2008, code available: http://www.nsnam.org/releases/ns-3.1.tar.bz2.

[20] "EU FP7 project PSIRP," http://www.psirp.org, accessed: 12.1.2013.

[21] "Rocketfuel ISP topology data." http://www.cs.washington.edu/research/networking/rocketfuel/maps/weights-dist.tar.gz.

[22] C. E. Rothenberg, C. Macapuna, F. Verdi, M. Magalhes, and A. Zahemszky, "Data center networking with in-packet bloom filters," in *SBRC 2010*, May 2010.

[23] C. E. Rothenberg, C. A. B. Macapuna, F. L. Verdi, and M. F. Magalhães, "The deletable bloom filter: a new member of the bloom family," *Comm. Letters.*, vol. 14, no. 6, pp. 557–559, Jun. 2010.