Scaling Bloom filter-based multicast via filter switching

Christos Tsilopoulos and George Xylomenos
Mobile Multimedia Laboratory, Department of Informatics
Athens University of Economics and Business
11362 Athens, Greece
Email: tsilochr@aueb.gr, xgeorge@aueb.gr

***Abstract*—Stateless multicast forwarding with in-packet Bloom filters (iBF) has recently been proposed as a highly scalable way for supporting a large number of multicast groups. However, iBF multicast generates redundant traffic due to false positive forwarding decisions and it also scales poorly with multicast group size. In this paper we investigate scaling iBF multicast to arbitrary multicast group sizes, by partially sacrificing the network's fully stateless operation. We propose a switched-iBF multicast scheme that places multicast forwarding state at a few network nodes, so as to minimize redundant traffic regardless of the group size. We evaluate the scheme through simulations and find that switched-iBF multicast can scale to any group size while keeping redundant traffic below 1%-4% at the (minimal) cost of placing state at no more than 0.5%-2.5% of network nodes. We also compare the state requirements of switched-iBF multicast against other multicast schemes. Our evaluation shows that switched-iBF multicast achieves a tremendous reduction of multicast state in the range of 87%-99.6%. Hence, even though the system is no more fully stateless, it remains far more scalable than other approaches.**

***Index Terms*—in-packet Bloom filters, multicast packet forwarding, switched-iBFs**

I. Introduction

During the last decade, developments in the areas of broadband access, always-on Internet connectivity and cloud computing have given rise to a set of new Internet applications and services. In this context, a number of cases arise in which data must be disseminated from a source to a (potentially large) number of receivers. Examples of this communication pattern include the provision of Internet services to users (e.g. on-line shared storage, on-line gaming), internal infrastructure procedures (e.g. bulk data backups, Map-Reduce systems etc.), or real-time multimedia applications such as *Networked Music Performance* (NMP), where multicast can be used for the direct exchange of data between NMP participants, without a centralized server [1]. Such applications can highly benefit from multicast delivery. Even though multicast had been an important research topic for a long time, multicast technologies face significant scalability problems with respect to the number of co-existing multicast groups. This technical constraint, as well as business constraints, have confined multicast to closed, centrally controlled, enterprise networks, where it facilitates a limited number of centrally-provided applications (e.g. IPTV).

Multicast scalability constraints are imposed by the hop-by-hop packet forwarding model: multicast forwarding state is distributed among network routers which maintain *Multicast Forwarding Tables* (MFT). Unlike unicast addresses, multicast addresses are logical and not topological identifiers; thus routers cannot aggregate MFT entries as in unicast IP forwarding, hence MFT sizes grow proportionally to the number of multicast groups traversing a router.

To alleviate these constraints, Bloom filters have been recently proposed as a method for single-source multicast delivery [2]. The idea is quite simple: multicast tree links are encoded into a Bloom filter which is then placed as source-routing information in packet headers; hence the term *in-packet Bloom filter* (iBF). At each hop, the router extracts the iBF, checks which of its outgoing links are encoded in the iBF and transmits the packet over those links. By moving multicast forwarding state from routers to packets, the network can support an arbitrary number of co-existing multicast groups without worrying about MFT size explosion. These benefits, however, do not come for free. iBF multicast is susceptible to redundant packet forwarding decisions (redundant traffic) due to the probabilistic nature of the Bloom filter data structure [3]. The rate of redundant forwarding decisions increases as the group size increases and more tree links are added. Previous studies have shown that once the false positive probability of an iBF exceeds 0.2%, forwarding anomalies arise, causing a sharp decline in bandwidth utilization [2], [4]. In essence, iBF multicast scales poorly with respect to group size.

In this paper we address the issue of iBF multicast scalability with respect to group size, by partially sacrificing the fully stateless operation of iBF multicast and, therefore, its scalability with respect to the number of groups supported. We achieve this by splitting a multicast delivery tree in a *few* sub-trees, installing multicast forwarding state at sub-tree roots and applying an iBF-switching forwarding logic. By doing so, we aim to minimize redundant traffic regardless of the group size. For the remainder of the paper, we call this multicast scheme *switched-iBF multicast*.

The contribution of this paper is twofold. First, we present a simple algorithm for the placement of iBF multicast state at network routers and evaluate our scheme through simulations over large, synthetic scale-free graphs. Evaluation results show that switched-iBFs can scale to any group size while keeping redundant traffic below 1%-4%, at the cost of placing state at no more than 0.5%-2.5% of network nodes. Second, we compare the forwarding state requirements of switched-iBF multicast with IP multicast and other multicast schemes proposed to reduce multicast forwarding state. The results show that switched-iBF multicast achieves a tremendous reduction of multicast state in the range of 87%-99.6%, i.e. nearly two levels of magnitude less. Hence, even though we sacrifice the fully stateless operation of iBF multicast, we still get far better scalability with respect to the number of groups.

The remainder of the paper is organized as follows. In Section II we present background work on iBF multicast: we describe the basic method and discuss its advantages and constraints, we briefly present available solutions for scaling iBF capacity without adding multicast state inside the network and finally argue that these solutions are *impractical* for packet forwarding. In Section III we present a switched-iBF multicast scheme and describe the tree-traversal algorithm for selecting sub-trees and stateful nodes. We evaluate the switched-iBF multicast scheme in Section IV and conclude in Section V.

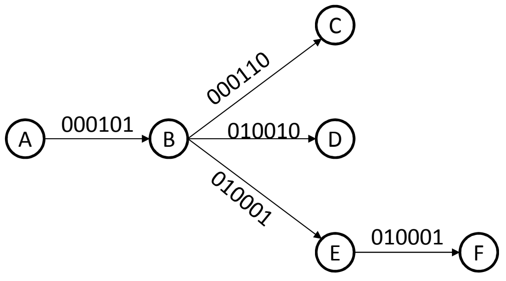


Fig. 1. A simple network with assigned LIDs ($m = 6$ and $k = 2$). LIDs need not be unique.

II. Background and Related Work

A. Multicast forwarding with in-packet Bloom filters

In iBF multicast, the links of the multicast tree are encoded into a Bloom filter which is then placed as a source-route in the packet header; hence the term *in-packet Bloom filter* (iBF). The construction of the iBF may occur at the source node [4] or it may be delegated to a separate routing module [2], [5]. To encode path links into Bloom filters, links are assigned with a *Link Identifier* (LID). An LID is an m-bit string with only k bits set to 1 ($k << m$). The positions of the k bits are determined using k *hash functions*, e.g. by applying the k hash functions on the network adapter's MAC address. LIDs are unidirectional (a bi-directional link is assigned two LIDs, one for each direction) and do not need to be unique. A delivery path is encoded into an iBF by ORing the path LIDs. In the example of Figure 1, the iBF for transmitting data from A to C is $LID_{AB}|LID_{BC} = 000111$. Forwarding elements extract the iBF from packet headers, examine which of their outgoing links are part of the iBF and transmit the packet over those links. If the expression

$$(iBF\&LID_i) == LID_i$$

evaluates to true, then the node *assumes* that LID_i is encoded into the iBF and transmits the packet over link i. In our example, when node B receives a packet with iBF 000111, it will forward it to C, since $iBF\&LID_{BC} = 000110 = LID_{BC}$. For multicast delivery, we OR all LIDs of the tree; the forwarding decision logic remains the same. In the example, the iBF for multicast delivery from A to {C, D} is $LID_{AB}|LID_{BC}|LID_{BD} = 010111$.

There are two significant advantages with iBF forwarding. First, it is very lightweight in terms of the state required at network elements: nodes store only their LID information which is proportional to their node degree. This also applies to multicast: as forwarding information is kept at packet headers and not at routers, the network can support an arbitrary number of multicast sessions without worrying about MFT explosion. The second advantage of iBF forwarding regards its capability for line-speed operation, in contrast to other source-routing methods. The forwarding logic is based on simple bitwise operations and can be easily implemented in hardware [2].

On the other hand, iBF multicast scales poorly with respect to multicast group size. This is because Bloom filters are probabilistic representations of sets: when a Bloom filter is queried whether an item is contained in the set, it may return a *false positive*. During packet forwarding, the iBF may falsely answer that an LID is present in the path, thus the packet will be forwarded over that link. Continuing the example, the iBF for multicasting data from A to {C, D} was 010111. When the multicast packet arrives at B, LID_{BE} also matches the iBF, thus the packet is forwarded to E. In addition, at node E the iBF also matches LID_{EF}, so the packet is also forwarded from E to F. In this example, a multicast delivery intended for 3 links resulted in 5 transmissions, i.e. 2 redundant transmissions. The amount of redundant traffic generated depends on the *false positive probability* (fpp) of the Bloom filter as given by [3]

$$fpp = (1 - e^{-kn/m})^k \tag{1}$$

where m is the size of the iBF, k is the number of hash functions used and n is the number of inserted items. The fpp increases as (i) the size of the iBF decreases or (ii) the number of inserted items increases. In iBF multicast, as group size increases, more links are added, thus the fpp also increases. Previous work on iBF multicast reported that once the fpp exceeds 0.2% there is a sharp rise in the amount of redundant traffic caused by false positives [2], [4].

To better illustrate the performance degradation of iBF multicast when the group size grows, Figure 2 presents simulation results for a synthetic scale-free graph with 500 nodes. The plot shows the forwarding efficiency of multicast delivery for various iBF sizes, defined as

$$\text{forwarding efficiency} = \frac{\#\text{multicast tree links}}{\#\text{total packets transmitted}} \tag{2}$$

If we target a forwarding efficiency of 90% (i.e. 10% of traffic being redundant), we see that multicasting with a 256-bit iBF and $k = 4$ can scale up to roughly 6-7 nodes. A 1024-bit iBF with $k = 6$ scales up to 60 recipient nodes. These results are consistent with previous research [2], [4], [5].

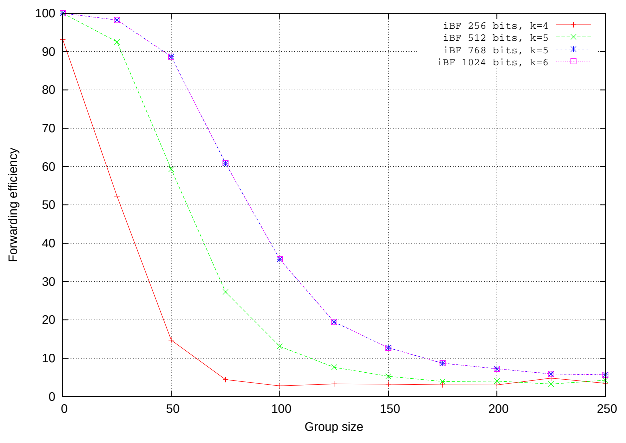


Fig. 2. iBF forwarding efficiency in a network with 500 nodes.

B. Related work

Mitigating forwarding anomalies in iBF multicast: Several research studies focus on mitigating forwarding anomalies such as redundant transmissions and, even, routing loops. LIPSIN proposed to install short-lived packet caches in routers in order to eliminate forwarding loops [2]. When a looped packet arrives at a router, the router compares it against this cache and discards it, thus eliminating the loop. This approach, however, introduces extra state at routers and increases per-packet processing overhead. Särelä et al. proposed a *varying-k* method for computing the LIDs and a *bit-permutation* scheme during iBF construction and forwarding [4]. These schemes mitigate forwarding anomalies but do not provide significant scalability benefits. Going one step further, Särelä et al. proposed *BloomCast*, a protocol for inter-domain iBF multicast [4]. In BloomCast, the iBF is computed based on the inter-domain graph and separate iBFs are used at each individual domain. When packets enter ASes, ingress routers encapsulate packets with a domain-specific iBF and tunnel them towards AS exit points. Although BloomCast shrinks the topology graph by applying the abstraction of inter-domain and intra-domain graphs, it does not solve the scalability problems for a single flat graph. For instance, the Internet is currently reported to contain more than 35000 ASes [6]. Using a single iBF for multicast delivery over the inter-AS graph, BloomCast scales up to 20 AS nodes with a 1024-bit iBF. ESM studied iBF multicast specifically in data center topologies [5]. The authors proposed to use the technique only for small (manageable) multicast groups, resorting to hop-by-hop multicast forwarding for bigger group sizes. In *Hierarchical Tree Splitting* (HST), the multicast tree is split in several sub-trees, *all* rooted at the source, so that the iBF for each sub-tree has a low fpp [7]. The source node maintains several iBFs (one per sub-tree) and transmits packets over all trees. HST preserves the stateless operation of routers, at the expense of additional multicast state at the source node and redundant traffic: the sub-trees may have overlapping links, thus causing multiple data transmissions. Moreover, HST still suffers from excessive redundant traffic in very large trees. Our goal is to minimize redundant traffic for arbitrary tree sizes.

Increasing the capacity of iBFs: Other studies focus on increasing iBF capacity without placing multicast state at routers. In general, the approaches for increasing the capacity of a iBFs are (i) to vary the size m of the Bloom filter or (ii) to compute the optimal value for k so that fpp remains low [3]. Varying m can be realized in two ways. First, by using a Bloom filter that grows dynamically as items are inserted [8], [9]. Second, by computing a value for m that minimizes the fpp, provided that we know n beforehand, i.e., we must first determine the multicast tree size and then compute the size of the iBF. In both cases, varying m means using different iBF sizes on a per tree size basis, which is not practical in the context of packet forwarding. First, m cannot grow without bound: there are limits to the size of the iBF imposed by MTU packet sizes, of which iBFs should be a relatively small part. Second, using variable length identifiers is *unfriendly* to line-speed hardware-level implementations of iBF forwarding.

The second approach for increasing the capacity of a Bloom filter is to keep m fixed and vary k [7]. This approach requires first determining the number of multicast tree links and then computing k. The value for k that minimizes fpp is [3]

$$k_{opt} = \frac{9m}{13n} \tag{3}$$

Varying k is impractical because it does not guarantee that the fpp will remain below a desired threshold. For example, if $m = 256$ and bits and $n = 200$ tree links, then $k_{opt} = 0.88$. In practice, however, k cannot be less than 1; at least one bit needs to be set in LIDs, otherwise packet forwarding will fail. In this case, if we set $k = 1$ and apply (1) we get $fpp \approx 18\%$, which is far too big compared to the suggested 0.2% [4]. Moreover, varying k is also impractical as it would require computing and setting LIDs in nodes on a per multicast tree size basis. In contrast, having a unique, predefined value for k allows LIDs to be computed and installed once, during network bootstrap.

Reducing IP Multicast Forwarding State: REUNITE proposed installing multicast forwarding state only at branching points of the multicast tree; non-branching points forward packets using existing unicast routing information [10]. This significantly reduces multicast forwarding state for sparse trees, but its benefits are minor in dense trees. Explicit Multicast (Xcast) proposed including the addresses of multicast receivers in packet headers [11], with routers forwarding multicast packets based on that information only. Xcast is a stateless scheme and therefore scales with respect to the number of multicast groups. However, due to the limited capacity of the packet header, it is only suitable for small groups. Yang and Liao proposed a semi-stateful version of Xcast [12] where multicast forwarding state is installed in a few, carefully selected routers, and packets are forwarded among these routers via Xcast. This is quite similar to switched-iBF multicast with two notable differences: (i) in Xcast packet headers contain host addresses for multicast receivers while in iBF tree links are encoded and (ii) in Xcast pre-existing unicast forwarding state is required. Our evaluation (see section IV) shows that due to the high compression capabilities of Bloom filters, switched-iBF multicast requires installing significantly less multicast state, thus providing better scalability properties. Finally, MAD identified that multicast group sizes follow a Zipf distribution and proposed using native IP multicast for the few large groups and overlay multicast for the smaller groups [13]. In our work, we show that even large groups can be accommodated with far less state than with IP multicast, thus further increasing the system's capacity for the large groups.

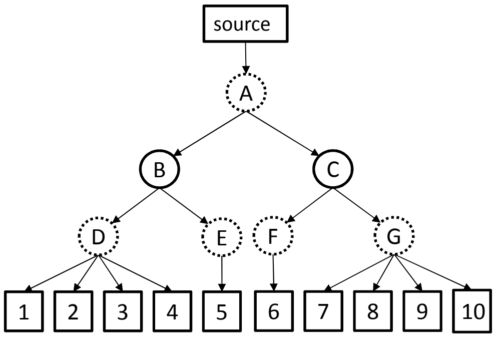


Fig. 3. An example delivery tree with 10 receivers. Dashed-line nodes are stateless. Solid-line nodes contains sub-tree iBFs.

III. Switched-iBF Multicast

A. Overview

The basic idea in switched-iBF multicast is to break the initial delivery tree into several sub-trees, compute the iBF for each sub-tree and place the respective iBFs as multicast forwarding state at sub-tree roots. At the data plane, multicast packets are forwarded using the same forwarding logic with one differentiation: upon reaching a stateful node, the node switches the packet's iBF with the stored iBF and further relays the packet using the new iBF. Sub-trees are selected so that the resulting iBFs will have a very low *false positive probability* (fpp). Figure 3 shows an example of a multicast tree with two stateful nodes. The source node uses an iBF to multicast data to the next immediate stateful nodes, i.e. B and C, which apply iBF-switching and further push packets down to receivers.

B. Selection of iBF-switching points

To reduce redundant traffic we need to maintain the Bloom filter's fpp below a certain threshold. According to (1), fpp depends on m, k and n. The first two parameters are fixed in practice, hence the only *tunable* parameter is n, i.e. the number of sub-tree links. We define a desired fpp_{max} threshold and use (1) to obtain the maximum number of sub-tree links

$$n_{max} = -ln(1 - fpp_{max}^{-\frac{1}{k}})(\frac{m}{k})$$

We traverse the multicast tree in a *bottom-up post-order* fashion, breaking it into sub-trees containing n_i links, so that $n_i \to n_{max}$ as shown in Algorithm 1. In the algorithm we denote n_i as the number of links of the sub-tree rooted at node i, C_i as the set of i's children in the delivery tree and iBF_i as the iBF for the sub-tree rooted at node i. For example, consider the tree of Figure 3 and assume $n_{max} = 6$. When we visit nodes D and E we have $n_D = 4$ and $n_E = 1$, so we move to node B where $n_B = 6 = n_{max}$. B becomes an iBF-switching point. We compute iBF_B, install it at B, prune the sub-trees rooted at B and reset n_B to 0. Similarly, node C is selected as an intermediate iBF-switching point. Finally, at the source node we compute the iBF for multicasting data to nodes B and C.

Algorithm 1 Sub-tree selection

```
***Method***: TreeTraverse
***Input***: $t$: multicast tree;
        $s$: tree source
        $fpp\_thres$: maximum $fpp$;
$n_{max}$ := computeNmax($fpp\_thres, m, k$);
$n_{source}$ := sub_tree_traverse ($t, s, n_{max}$);
$iBF_s$ := compute_iBF($t, s$);
return $iBF_s$;
***end method***

***Method***: sub_tree_traverse
***Input***: $t$: multicast tree;
        $i$: current root node;
        $n$: maximum number of nodes;
$n_i$ := 0;
for ($j$ in $C_i$) {
        $n_i$ := $n_i$ + 1 + sub_tree_traverse ($t, j, n$); //recursion
}
if ($n_i \geq n_{max}$) {
        $iBF_i$ := compute_iBF($t, i$);
        installState($iBF_i, i$);
        removeSubtree($t, C_i$);
        $n_i$ = 0;
}
return $n_i$;
***end method***
```

C. Multicast tree construction

Our focus in this paper is on the scalability properties and not on the signaling details of the iBF multicast scheme. We will therefore only outline how our scheme can be combined with different multicast tree construction algorithms. Bloom filter-based multicast systems are generally classified in two categories, depending on how the multicast tree is constructed. In the first category, the tree is constructed in a distributed manner and the iBF is computed at the source node [4], [7]. In these systems, users issue multicast JOIN messages which are forwarded by routers towards the specified source. As the JOIN messages are propagated, they collect reverse path information. When a JOIN reaches its destination, the source node extracts the reverse path, appends it to the preexisting delivery tree (if any) and computes the respective iBF. A basic assumption in these systems is that nodes have sufficient topological information to forward JOIN messages and compute the iBFs. In the second category, the multicast tree and the iBF are *centrally* computed by a separate routing module that resides in a dedicated server [2], [5]. In these systems, JOIN messages are delivered directly to the routing module which computes the multicast tree based on topological information (e.g. link delays), constructs the iBF and then sends it to the group source. In these systems, the JOIN delay is increased, due to the need to also communicate with the routing module.

Our switched-iBF scheme is compatible with both options. The only extension required is that the node computing the iBF must also (i) decide where multicast state should be placed and (ii) instruct the selected nodes to install the appropriate state. Note that iBF multicast is, in general, suitable for applications with low group dynamics (e.g. orchestrated data delivery, data backup) and not for highly dynamic multicast groups (e.g. IPTV). This limitation holds for our scheme as well.

IV. Evaluation

We evaluated the scalability properties of switched-iBF multicast via extensive simulation tests on a custom-made simulator. We used synthetic scale-free graphs generated with the Barabási-Albert algorithm [14] ranging from 1000 to 5000 nodes. LIDs for links were constructed using *Double Hashing* [3] with SHA-1 and MD5. For each tested graph of size s we generated s different multicast groups, e.g. for a network of 5000 nodes we generated 5000 multicast groups. The size of each multicast group is uniformly distributed in $[10, s-10]$, i.e, groups have at least 10 and at most $s-10$ nodes. For each multicast group, group members are randomly selected among all nodes. For multicasting we used the *shortest-path trees*, i.e. the union of the shortest paths between the source and each receiver. We saw similar behavior in all graph sizes, hence, due to lack of space, we only present results for 5000 node graphs.

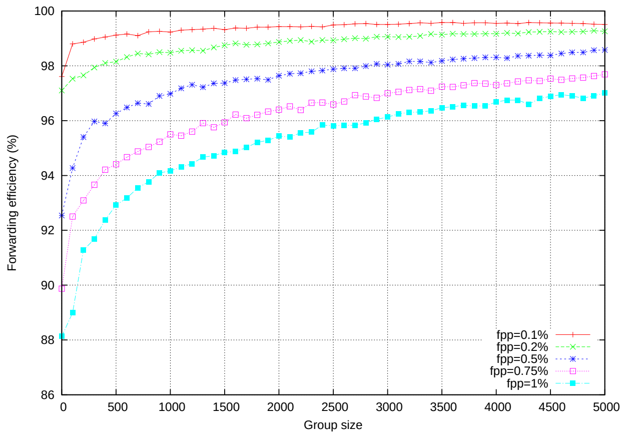


Fig. 4. Forwarding efficiency of switched-iBF multicast. Network size 5000, iBF size 256 bits, $k = 4$, variable fpp.

A. Forwarding efficiency and state requirements

Figure 4 shows the forwarding efficiency defined in (2) as a function of group size in a graph of 5000 nodes with $m = 256$ and $k = 4$. When $fpp = 0.1\%$, forwarding efficiency is kept above 99%, i.e. false positives account for less than 1% of overall traffic, regardless of group size. Forwarding efficiency is not constant: it starts from a low value and increases quickly. In small groups several false positives occur, but as group size grows and more links are added, fewer links can count as false positives. Overall, the forwarding efficiency of switched-iBF multicast increases as the group size increases, i.e. switched-iBF multicast behaves better in larger multicast groups.

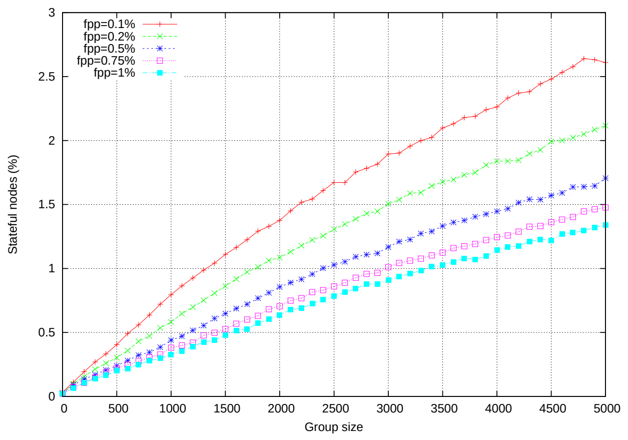


Fig. 5. State requirements of switched-iBF multicast. Network size 5000, iBF size 256 bits, $k = 4$, variable fpp.

Figure 5 shows the state requirements of switched-IBF multicast as a function of group size, with the same parameters as above. The Y-axis shows the percentage of nodes where state must be installed. The fraction of stateful nodes grows linearly with multicast group size. As fpp increases, the size of sub-trees also increases; hence the number of stateful nodes decreases. In spite of the linear growth of state requirements, notice that state requirements are overall quite low. For instance, multicasting to a group of 2500 nodes with $fpp = 0.5\%$ (forwarding efficiency at 98%) requires multicast state at only 1%, i.e. $5000 \times 1\% = 50$ nodes. For the remainder of the paper, we present results for $fpp = 0.5\%$ as it provides a good trade-off between forwarding efficiency (quickly passes 96% and converges to 98%) and state requirements (requires half the nodes compared to $fpp = 0.1\%$).

In Figure 6, we show the state requirements for various values of m and k with $fpp = 0.5\%$ in a 5000 node graph. As m increases, the fraction of stateful nodes decreases as expected: as iBF size grows, more tree links can be added to the iBF without the fpp passing the selected threshold. Notice that with $m = 1024$ bits, switched-iBF multicast requires state at around 0.4% of the nodes, i.e. we can almost broadcast (e.g. transmit a firmware update to all routers) with multicast forwarding state at only $5000 \times 0.4\% = 20$ nodes.

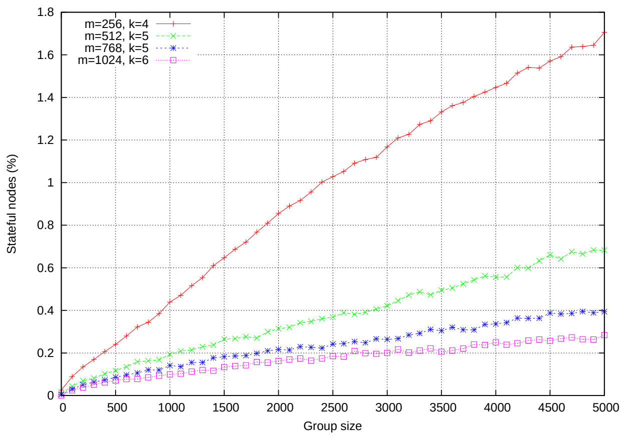
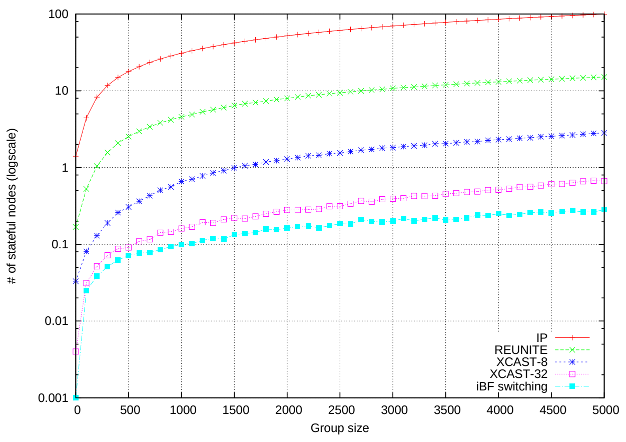


Fig. 6. State requirements of switched-iBF multicast. Network size 5000, $fpp = 0.5\%$, variable m and k.

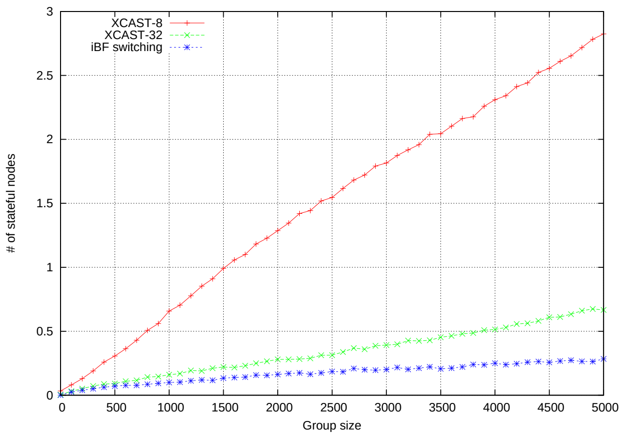
B. Comparison against IP multicast forwarding schemes

We then compare multicast forwarding state requirements between switched-iBF multicast and various schemes proposed for reducing IP multicast forwarding state. Specifically, we compare switched-iBF multicast against REUNITE [10], the semi-stateful Xcast [12] and plain IP Multicast. REUNITE reduces state by installing multicast forwarding information only at branching points of the multicast tree. The semi-stateful Xcast scheme installs multicast state at a few network nodes and multicasts packets among them using Xcast [11]. Plain IP multicast places multicast forwarding state at each node of the multicast tree.

Figure 7(a) presents the fraction of stateful nodes as a function of multicast group size in a 5000-sized graph. We chose $m = 1024$ bits and $fpp = 0.5\%$ for switched-iBF multicast. For semi-stateful Xcast we considered a header of the same size, i.e. 1024 bits, and then considered two variants: (i) in XCAST-8 the header may contain up to 8x128-bit host addresses and (ii) in XCAST-32 the header may contain up to 32x32-bit host addresses. XCAST-8 targets a possible deployment of Xcast in IPv6 networks while XCAST-32 targets a hypothetical deployment in IPv4 networks. XCAST-32 is impossible to realize of course, as IPv4 headers cannot exceed 40 bytes, but we present the results for completeness. The simulation setup is the same as in the previous section. Due to wide difference between the various schemes, the Y-axis in Figure 7(b) is in log scale. In all schemes, multicast forwarding state grows linearly with respect to group size. However, switched-iBF achieves a large state reduction compared to the other schemes. For instance, for a 2000 node group, switched-iBFs reduce the number of stateful nodes by 99.6% compared to IP multicast, 98% compared to REUNITE, 87% compared to XCAST-8 and 42% compared to XCAST-32. Figure 7(b) presents the same data but only for semi-stateful Xcast and switched-iBF multicast, with the Y axis in linear scale. Although switched-iBF and semi-stateful Xcast both use a combination of source-routing and stateful nodes, switched-iBF requires far less state due to the efficient encoding of the Bloom filter, even though we restrict the number of tree links so that $fpp = 0.5\%$. Even XCAST-32 requires double the state of switched-iBF. Taking into account that REUNITE and Xcast also require unicast forwarding state, it is clear that an iBF-based forwarding node requires far less overall state. Hence, although we sacrificed the fully stateless nature of iBF multicast, it both scales with respect to multicast group size and remains a far more scalable solution in terms of the number of multicast groups supported.



(a) IP Multicast, REUNITE, semi-stateful Xcast and switched-iBF multicast (log scale).



(b) Semi-stateful Xcast and switched-iBF multicast (linear scale).

Fig. 7. Fraction (%) of stateful nodes against group size.

V. Conclusion

In this paper we addressed the scalability issues of iBF multicast with respect to multicast group size. We examined the option of sacrificing the fully stateless operation of iBF multicast in order to minimize the redundant traffic caused by false positives in large multicast groups. We presented a switched-iBF multicast scheme and an algorithm for the selection of iBF switching points. Our evaluation through simulation showed that our solution scales to groups of arbitrary size with a (minimal) cost of placing multicast forwarding state at no more than 0.5%-2.5% of network nodes. Moreover, compared with other multicast schemes that reduce forwarding state, switched-iBF multicast achieves state reductions varying from 87% to 99.6%. Hence, even though switched-iBF multicast places state inside the network, it provides far better scalability properties with respect to the number of groups supported, compared to related solutions.

ACKNOWLEDGEMENT

This research has been co-financed by the European Union (European Social Fund - ESF) and Greek national funds through the Operational Program "Education and Lifelong Learning" of the National Strategic Reference Framework (NSRF) - Research Funding Program: THALIS - University of Crete - MUSINET.

REFERENCES

[1] C. Stais, Y. Thomas, G. Xylomenos, and C. Tsilopoulos, "Networked Music Performance over Information-Centric Networks," in Proc. IEE IIMC Workshop, June 2013.

[2] P. Jokela, A. Zahemszky, S. Arianfar, P. Nikander, and C. Esteve, "LIPSIN: Line speed publish/subscribe inter-networking," in Proc. ACM SIGCOMM, Aug. 2009.

[3] S. Tarkoma, C. E. Rothenberg, and E. Lagerspetz, "Theory and Practice of Bloom Filters for Distributed Systems," IEEE Communications Surveys & Tutorials, vol.14, no.1, pp.131-155, First Quarter 2012.

[4] M. Särelä, C. Rothenberg, T. Aura, A. Zahemszky, P. Nikander, and J. Ott, "Forwarding anomalies in bloom filter-based multicast," in Proc. IEEE INFOCOM, Apr. 2011.

[5] D. Li, Y. Li, J. Wu, S. Su, and J. Yu, "ESM: efficient and scalable data center multicast routing," IEEE/ACM Transactions on Networking, vol. 20, n. 3, pp. 944-955, June 2012.

[6] CAIDA, September 2012. [Online]. Available: http://www.caida.org

[7] S.Rizvi, A Zahemszky, and T. Aura, "Scaling Bloom filter based multicast with hierarchical tree splitting," in Proc. IEEE ICC, June 2012.

[8] P. S. Almeida, C. Baquero, N. Preguia, and D. Hutchison, "Scalable Bloom filters," Inf. Process. Lett., vol. 101, no. 6, pp. 255-261, 2007.

[9] D. Guo, J. Wu, H. Chen, Y. Yuan, and X. Luo, "The dynamic Bloom filters," IEEE Transactions on Knowledge and Data Engineering, vol. 22, no. 1, pp. 120-133, 2010.

[10] I. Stoica, T.S.E. Ng and H.Zhang, "REUNITE: a recursive unicast approach to multicast," in Proc. IEEE INFOCOM, March 2000.

[11] R. Boivie, N. Feldman, Y. Imai, W. Livens, D. Ooms, and O. Paridaens, "Explicit multicast (Xcast) basic specification", IETF Internet Draft, 2000.

[12] De-Nian Yang and W. Liao, "Protocol design for scalable and adaptive multicast for group communications," in Proc. IEEE ICNP, Oct. 2008.

[13] T. Cho, M. Rabinovich, K. Ramakrishnan, D. Srivastava, and Y. Zhang, "Enabling Content Dissemination Using Efficient and Scalable Multicast," in Proc. IEEE INFOCOM, April 2009.

[14] A. Barabási and R. Albert, "Emergence of Scaling in Random Networks," Science, vol. 286, pp. 509-512, 1999.