Loop Mitigation in Bloom Filter Based Multicast: A Destination-Oriented Approach

Xiaohua Tian
Department of Electronic Engineering
Shanghai Jiao Tong University
xtian@sjtu.edu.cn

Yu Cheng
Department of Electrical and Computer Engineering
Illinois Institute of Technology
cheng@iit.edu

***Abstract*—Recently, several Bloom filter based multicast schemes have been proposed, in which multicast routing information is carried with an in-packet Bloom filter. Since routers have no need to maintain forwarding states on a per-group basis, the Bloom filter based multicast protocols have desirable scalability. However, a critical issue is that these schemes may incur forwarding loops due to the false positive inherent in the Bloom filter. Existing solutions can only conditionally mitigate the probability of the forwarding loop, instead of fully preventing such events which (once occurred) will cause severe damage to the network. In this paper, we resolve this issue in the context of a destination-oriented multicast (DOM) scheme, a Bloom filter based multicast protocol carrying destinations IP addresses with the in-packet Bloom filter. With a theoretical analysis of the loop issue in DOM context developed, we reveal that the DOM design natively supports automatical elimination of permanent forwarding loops in all cases except a subtle one termed as conservation of bits. Based on the conclusion, we derive a probability upper bound on the loop occurrence in DOM. Furthermore, we propose an accurate tree branch pruning scheme, which equips the DOM the capability to completely and efficiently remove the false-positive forwarding loop. We present simulation results over a practical topology to demonstrate the performance of the loop mitigating DOM, with comparison to a representative Bloom filter based multicast scheme FRM and traditional IP multicast.**

I. Introduction

A scalable inter-domain multicast protocol has been an open research issue in recent two decades [1]–[4]. Proposed by Deering in 1988, IP multicast delivers the shared data along a network-layer based tree structure constructed using a distributed multicast routing algorithm [5]–[7], [10]. It is bandwidth efficient in data delivery but poor scalable in managing the multicast tree [1], [2], [10], since each router needs to maintain the multicast forwarding states for every group passing through; the messaging overhead and the memory cost grow linearly with the number of multicast groups being supported by the router. The more recent overlay multicast establishes the data-dissemination structure at the application layer [11], [12], wherein each overlay link is an end-to-end unicast path between two hosts. Although convenient for deployment as the underlying unicast infrastructure needs no modification, overlay multicast induces redundant traffic at the network layer [11]: it is common that separate overlay links pass through the common physical links in the underlying transport network.

Recently, several schemes, e.g., FRM [13], LIPSIN [14] and BloomCast [16], have been proposed to improve the scalability of the network-layer multicast. While these schemes vary in details, they share the same design philosophy: the domain-level multicast tree is carried with an in-packet Bloom filter, and the forwarding router examines each of its neighboring domain-level edges against the in-packet Bloom filter to compute appropriate packet copies and output interfaces. Since routers have no need to maintain forwarding states on a per-group basis, the Bloom filter based multicast protocols have desirable scalability.

FRM is the earliest proposed multicast protocol of this kind [13]. A critical issue of FRM is that the false positive inherent in the Bloom filter may incur forwarding loops, which is taken into consideration by its later variants. In LIPSIN, routers cache the suspect pattern of the in-packet Bloom filter to detect loops [14], but can impose heavy reencoding burden on the data source node; moreover, LIPSIN can not guarantee that the loop previously detected will be completely eliminated. BloomCast proposes a *bit permutation* technique, with which the bit positions in the Bloom filter are re-mapped to a different arrangement at each intermediate router; thus the false positive forwarding at each node is reduced [16]. However, the BloomCast can only work in the symmetric routing environment, and bit permutation can only mitigate the probability of the forwarding loop rather than totally prevent it. For multicast protocols with the FRM-like flavor, the forwarding loop issue has not been completely resolved.

In this paper, we resolve the forwarding loop issue in the context of a destination-oriented multicast (DOM) scheme [17]–[19]. The implementation of DOM is also based on the Bloom filter, but the fundamental difference from the FRM-like protocols is that DOM carries destinations IP addresses with the in-packet Bloom filter, instead of the multicast tree. In the DOM design, each DOM-aware router maintains certain amount of local states that are independent of the number of groups to facilitate multicasting. Such a design has the significant advantage in reducing the routing information to be carried by the packet thus the false positives and bandwidth cost in DOM [18], [19]. The states can also be utilized to achieve fast group joining [20], with which the average data access delay of DOM could be reduced. We summarize and compare the major features of DOM and FRM in Table I. The

TABLE I
FEATURES COMPARISON BETWEEN DOM AND FRM

Features	DOM	FRM
Asymmetric routing	Handled with border gateway protocol (BGP) view combined with reverse path forwarding (RPF) based joining [19]	Source routing where source node encodes multicast tree branches [13]
Major states at in-netwrok router	Depends on the number of destination domains can be reached through the router [18], [19]	Depends on the number of neighboring links of the router [13]
Bandwidth efficiency	Depends on the number of destination addresses encoded in the packet [18], [19]	Depends on the number of tree branches encoded in the packet [13]
Joining operation	Can be accelerated with fast joining [20]	Has to be completed at source node
Falsely forwarded traffic	Can be completely blocked	Can be constrained with probability [14], [16]
Forwarding loops	Can be completely removed	Can be constrained with probability [14], [16]

last two features of DOM are to be realized in this paper.

This paper will further exploit the DOM local states to achieve complete forwarding loop elimination. We first develop a theoretical analysis of the loop issue in DOM context, which illustrates the severe damage the forwarding loop (once occur) can cause in large-scale network such as Internet backbone. We then reveal that the DOM design natively supports automatical elimination of permanent forwarding loops in most cases except a subtle one termed as *conservation of bits*. Based on this conclusion, we derive a probability upper bound on loop occurrence in DOM. Furthermore, we propose an accurate tree branch pruning scheme, which can block the falsely forwarded traffic at the root point in the network so that the bandwidth efficiency is maintained. We also show that the false positive forwarding loop in DOM can be completely and efficiently removed with the proposed pruning scheme. Simulation results over a practical topology are presented to demonstrate the performance of the loop mitigating DOM, with comparison to the representative Bloom filter based multicast scheme FRM and traditional IP multicast.

The remainder of this paper is organized as follows. In Section II, we review existing loop prevention schemes and briefly describe the DOM working principles, so that the following contents can be understood. Section III presents the theoretical analysis of forwarding loop issue in DOM context. Section IV proposes an accurate branch pruning scheme for DOM. Performance of DOM is demonstrated through simulation results in Section V. Section VI gives the conclusion remarks and the future work.

II. BACKGROUND

A. Related Work on Loop Prevention

The forwarding loop in the network normally can be eliminated by reducing the value of time-to-live (TTL) field in the IP header. However, the value of TTL field in the inter-domain routing scenario is usually large. If such loops are small, the TTL value reduction may not prevent substantial unnecessary looping over domains [15]. Moreover, the error-prone configuration of inter-domain routing can also cause the innate Internet loop prevention scheme ineffective [16]. Considering all these reasons, it is necessary to purposely design a scheme for the Bloom filter based multicast protocols to eliminate forwarding loops.

LIPSIN proposes to deploy a FRM-like multicast protocol in a publisher/subscriber network fabric [14]. Each link of the network is assigned d IDs. Thus there are d candidate in-packet Bloom filters for a given multicast tree, from which a Bloom filter with the best false positive performance can be selected. When receiving a packet, the router analyzes the in-packet Bloom filter to check if it contains a path that may lead the packet to return. If positive, the packet and its incoming interface will be cached. A loop is detected if the packet with the cached in-packet Bloom filter returns to the router from an interface other than the cached one. Nevertheless, the router caching the suspect packet is not necessarily the origin of the loop; therefore the false positive traffic can not be fully truncated. To deal with the challenge, the caching router has to signal a request upstream towards the data source to insert a different Bloom filter in the packet for the multicast tree, which imposes much burden on the data source node; moreover, there is no way to guarantee that the re-encoded Bloom filter will never incur forwarding loop at a different router in the network.

BloomCast proposes a *bit permutation* technique to reduce the Bloom filter false positive effect. Different from FRM's directly encoding tree branches at source node, BloomCast let joining messages record each hop they traveled starting from leaves of the tree, encode the hop in a Bloom filter and re-map the Bloom filter to a different arrangement at each intermediate router. A unique reverse shortest path tree (SPT) is then created at the data source node by ORing all cumulatively permuted Bloom filters in joining messages. During the forwarding, the falsely delivered packet can not be correctly de-mapped through the bit permutation at each hop, so the packet with no matched output interfaces will be dropped. Unfortunately, although BloomCast works smoothly under the symmetric routing assumption, where the shortest path from node A to B is the same one used to go from B to A, the inter-domain routing is usually asymmetric for the administrative reasons [2]. Further, BloomCast still can not identify the origin of the forwarding loop once it occurs, and bit permutation can only mitigate the probability of the forwarding loop rather than totally prevent it.

The proposed loop mitigation scheme will completely eliminate the forwarding loop incurred by the false positive,

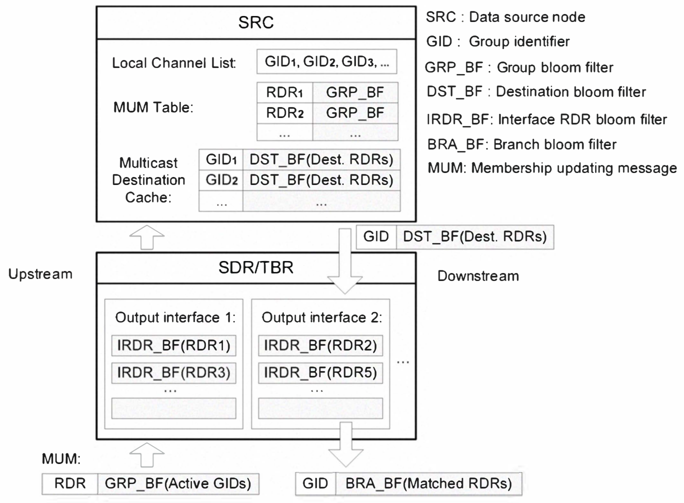


Fig. 1. Bloom-filter based design of DOM.

which is to be theoretically proved in the paper. By utilizing the forwarding states installed in the intermediate routers, the origin of the forwarding loop (once occurred) will be accurately identified. Then the pruning state is installed at the origin node of the loop to fully truncate the traffic falsely delivered, without bothering the data source node. Our design can work in both symmetric and asymmetric routing scenario, which can accommodate the asymmetric inter-domain routing environment.

B. Destination-Oriented Multicast (DOM)

We are to describe the Bloom-filter based design of DOM according to the upstream procedure (i.e., states establishment) and downstream procedure (i.e., data forwarding), as illustrated in Fig. 1, where Bloom filters are illustrated as shadowed areas.

1) Forwarding States Establishment: The left side of Fig. 1 shows how forwarding states are established at routers. With DOM, a border router of a stub autonomous system (AS) domain is selected as the *designated router* (DR). For convenience, we use RDR (SDR) to denote the DR of a receiver-side (source-side) AS domain, and use TBR to denote the transit-domain border router between SDR and RDR. The RDR basically needs to implement the internet group management protocol (IGMP) [9] to discover the active groups within its domain. When new groups are activated, the RDR is triggered to send *membership updating messages* (MUMs) to the *data source node* (SRC) in the format as (RDR: GID_1, GID_2, $\cdots$, GID_n), where RDR represents a domain prefix and GID represents the group ID.

To reduce the bandwidth overhead for membership updating, the list of active groups in the MUM message is encoded with a *group Bloom filter* (GRP_BF). When an MUM message reaches an upstream TBR/SDR router, the router will retrieve the RDR prefix, and store it as a local forwarding state at the output interface corresponding to the MUM incoming interface; the local states will later be used for reverse path forwarding. By continuously observing the MUMs, each related interface of the TBR/SDR will memorize all the destination domains that can be reached through it. At an output interface, each RDR is stored as a separate Bloom filter, termed as *interface RDR Bloom filter* (IRDR_BF). These IRDR_BFs actually establish a reverse shortest path tree (SPT) from the SRC to subscribing RDRs, which will be used to facilitate multicast forwarding. DOM incorporates the border gateway protocol (BGP) [8] into the revers path forwarding to make sure the joining scheme can establish a correct reverse SPT even in the asymmetric inter-domain routing environment [19], [20].

The upstream MUM messages will finally reach the SRC node, and each message will be stored as a record of the MUM table. The SRC node should have a *local channel list* indicating the multicast groups it provisions. By checking each GID against the MUM table and identifying the matched GRP_BF, the SRC can detect the destination prefixes for a given group. The destinations information under the group ID will be encoded into a *destination Bloom filter* (DST_BF) and stored into the multicast destination cache.

2) Multicast Data Forwarding: The right side of Fig. 1 illustrates how multicast packets are forwarded. At the SRC node, the DST_BF for a group will be inserted as the destination information into each multicast packet. When receiving a multicast packet, a DOM router performs the following processing: first, compare the packet's DST_BF with IRDR_BFs at each interface. If the DST_BF matches any IRDR_BF installed at the interface, tag this IRDR_BF; second, replicate the packet's payload for each unique interface that has any IRDR_BF tagged in the first step; third, perform OR operation on all tagged IRDR_BFs found in the first step to form a new Bloom filter termed as *branch Bloom filter* (BRA_BF) for each interface; fourth, equip the replicated packet's payload with the yielded BRA_BF at each interface and delivered the new packet along the interface. The BRA_BF will serve as the destination information DST_BF for further downstream forwarding.

The downstream procedure actually follows the *reverse path forwarding (RPF) concept: the data packet is forwarded along the path that is reverse to the MUM joining path.* At each interface, the DOM router remove unnecessary RDRs from the DST_BF, so that the downstream router will not generate unnecessary packet copies for those RDRs that have been delivered over other sibling subtrees.

III. Theoretical Analysis of DOM on Loops

This section first illustrates how the forwarding loop could happen in DOM and how serious the consequence of the loop will be. We then show that design of DOM could facilitate preventing the permanent forwarding loop in all cases except a subtle conservation of bits one, based on which we present the upper bound of the probability for the permanent loop in DOM.

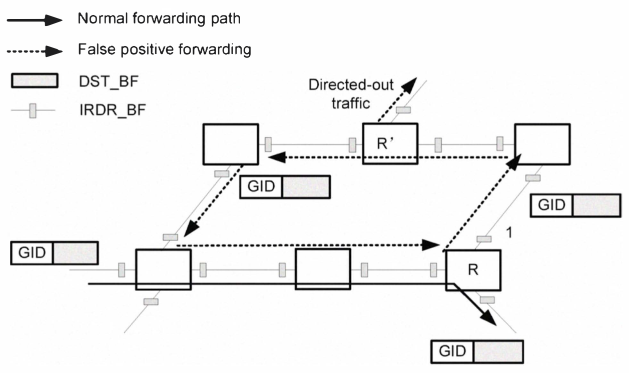


Fig. 2. Loop caused by false positive forwarding.

A. Forwarding Loop Issue in DOM Context

In the DOM forwarding process, some RDR represented by the IRDR_BF may be falsely detected in the DST_BF, thus a packet copy will be falsely forwarded via the corresponding interface. Let $f(n_1, 1)$ denote the false positive rate for bit matching between two Bloom filters, one containing n_1 elements and the other 1 element. The false positive forwarding over an interface with DOM is:

$$F_{DOM} = 1 - (1 - f(n_1, 1))^{n_3} \approx n_3 f(n_1, 1), \qquad (1)$$

where n_1 indicates the number of RDRs contained in the DST_BF, which is compared to n_3 IRDR_BFs stored in the interface under consideration, with each containing 1 RDR, to search for a match. The term $(1-f(n_1, 1))^{n_3}$ is the probability that none of n_3 IRDR_BFs installed at the interface incur false positive when performing bit matching with the DST_BF in the packet. The expression of $f(n_1, 1)$ can be found in our previous work [18], where the false positive rate of bit matching between two general Bloom filters is analyzed.

The false positive forwarding can happen at each interface independently, and the forwarding loop is formed when the falsely forwarded packet keeps mismatching with neighbor edge states installed along the interfaces that constitute the loop topology, as the dotted lines with arrows depicted in Fig. 2.

The consequence of the forwarding loop incurred by the false positive forwarding can be serious. Consider a packet being forwarded by a node within the loop and assume the packet will be circulated back to the node after T seconds, as illustrated in Fig. 3. The loop circulation means that a single packet will lead to a input flow with rate of $1/T$ packets/second. Given a link capacity of C, just $C \cdot T$ data packets will totally block the link, which could impact some multimedia applications requiring extremely low packet loss rate or even lead to the partial break-down of the network.

B. Automatical Loop Elimination with DOM

Lemma 1: The DOM downstream forwarding can eliminate forwarding loops incurred by the false positive forwarding under the conditions i) the domains associated with the SDR and RDRs are stub domains of the multicast group under consideration; ii) the number of 1-bit positions in the DST_BF will decrease if the multicast flow diverges at some node.

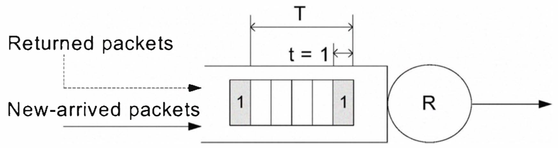


Fig. 3. Effect of the loop on network performance.

Proof: Because of condition i), the loop can only happen among TBRs [18]. Similar to the case in Fig. 2, the forwarding loop in DOM is formed if some falsely forwarded packet keeps mismatching its updated DST_BF with IRDR_BFs along the interfaces that constitute the loop topology. However, as the DOM protocol incorporates the DST_BF updating operation, the number of 1-bit positions in the DST_BF will continuously decrease as the packet travels around the loop. Specifically, for a given IRDR_BF that is falsely incorporated into the updated DST_BF, there must be a TBR in the loop, which will direct at least one of the packet copy out of the loop and remove the corresponding IRDR_BF from the original DST_BF. This is because any IRDR_BFs can not be installed to form a loop according to the forwarding states establishment process in DOM. In addition, the updated DST_BF for the packet copy that remains in the loop can only contain less 1-bit positions than the original DST_BF according to condition ii). Consequently, the packet copy remains in the loop will have fewer and fewer 1-bit positions in its DST_BF. When the remained 1-bit positions can not match any IRDR_BF in a TBR, the packet is dropped and the forwarding loop is eliminated. ■

The condition ii) of Lemma 1 seems to be redundant, because if the multicast flow diverges, the corresponding BRA_BF along each tree branch will contain less RDRs, and the number of 1-bits representing those RDRs will be set to 0. However, when the false positive comes into play, this may not be the case. Consider the subtle case illustrated in Fig. 4. The packet should have gone through interface 2 of TBR B, but the false positive forwarding occurs along interface 1; moreover, the yielded BRA_BF happens to set all 1-bit positions the same as in the incoming packet DST_BF. If the yielded data packet keeps mismatching along a loop topology and the all 1-bit positions remain unchanged, the DST_BF updating operation will not be able to eliminate the false positive forwarding loop. This kind of event is termed as *conservation of bits.*

C. Upper Bound on Loops in DOM

Theorem 2: The upper bound of the probability that a permanent loop occurs in DOM is

$$\left(\sum_{i=1}^{M} \binom{M}{i} \cdot {P_f}^i \cdot (1-P_f)^{M-i} \cdot (1-(1-\frac{k}{m})^i)^X\right)^3 \qquad (2)$$

under the conditions i) in Lemma 1, where

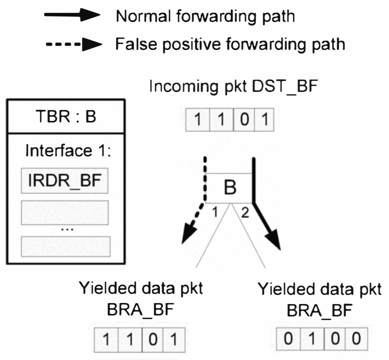


Fig. 4. Conservation of bits.

- M is the maximum number of IRDR_BFs on an interface of a TBR in the network;
- P_f is the false positive rate for bit matching between two Bloom filters, one containing the maximum number of elements in DST_BF and the other 1 element;
- X is the maximum number of 1-bit positions in a DST_BF.
- m is the total number of bit positions in a DST_BF or IRDR_BF;
- k is the number of hash functions to form a DST_BF or IRDR_BF.

Proof: According to Lemma 1, the permanent forwarding loop occurs only if the conservation of bits event illustrated in Fig. 4 happens along a loop topology; because the DST_BF updating operation of DOM can always eliminate forwarding loops in other cases automatically, the probability for conservation of bits event happens along a 3-node loop topology must be the upper bound for a permanent loop occurs.

We first find the probability that the conservation of bits event happens on one node. Specifically, suppose we have a DST_BF with X bits of 1s, and there are M IRDR_BFs on the local interface. We want to find the probability that at least one of the IRDR_BFs along the interface incur false positive forwarding and those falsely matched IRDR_BFs form a BRA_BF that is the same as the original DST_BF. This probability is:

$$\sum_{i=1}^{M} \left(\begin{array}{c} M \\ i \end{array} \right) \cdot {P_f}^i \cdot (1-P_f)^{M-i} \cdot P(e), \qquad (3)$$

where e is the event: given a set of i falsely matched IRDR_BFs, a DST_BF has at least one of the IRDR_BF matches all bit positions in itself, and $P(e)$ is the probability that event e happens.

$P(e)$ can be found through $1 - P(\bar{e})$. According to the principle of the Bloom filter, an IRDR_BF has probability $\frac{k}{m}$ to set a single bit position the same as in the original DST_BF, thus the probability that given i falsely matched IRDR_BFs, none of the IRDR_BF matches *a single 1-bit position* in the DST_BF is $(1-\frac{k}{m})^i$.

For X bit positions in the original DST_BF, the probability that each 1-bit has at least one of the i mismatched IRDR_BFs setting the same 1-bit position is:

$$P(e) = (1-(1-\frac{k}{m})^i)^X. \qquad (4)$$

Consequently, the probability that the conservation of bits event happens on one node is:

$$\sum_{i=1}^{M} \left(\begin{array}{c} M \\ i \end{array} \right) \cdot {P_f}^i \cdot (1-P_f)^{M-i} \cdot (1-(1-\frac{k}{m})^i)^X. \qquad (5)$$

Thus the conservation of bits event over 3-node loop happens with the probability as shown in (2). ∎.

The upper bound on permanent loops in DOM can be very low. Suppose we use the same Bloom filter configuration as in FRM [13], and the network topology adopted in [19], the resulted upper bound is in the order of 10^{-36}.

IV. Complete Loop Elimination in DOM

The root cause of the loop is false-positive forwarding. In this section, we first describe a tree branch pruning scheme based on the RPF forwarding mechanism in DOM, and then we show that the scheme can delete the loop caused by false-positive forwarding.

A. Pruning False Tree Branches

In DOM, a RDR stored on an interface of a router (in the form of an IRDR_BF) implies that a forwarding path through the interface exists from the router to the destination domain represented by the RDR, according to the joining process and the RPF techniques adopted in DOM. Note that such a fact is true in both symmetric and asymmetric scenarios [19]. Thus if a false positive forwarding happens in the network, due to mismatching with a certain RDR_i on an interface, the falsely forwarded packet will finally reach the destination domain associated with RDR_i. The destination domain can then identify that the traffic was due to false forwarding if the group was not requested by it, and subsequently sends a pruning message upstream reverse to the forwarding path to prune the false-forwarding branches and stop the mis-delivered packets.

Implementing the branch pruning design is not trivial. How to ensure the direction of the upstream pruning messages along those false-positive branches? Consider the example shown in Fig. 5, where RDR D and E subscribe to SRC S_1 and S_2 along path D-B-A and E-B, respectively. The packet generated by S_1 may be falsely forwarded to E if a false-positive match with IRDR_BF(E) on interface 2 of B happens. When E received the mis-delivered packet associated with S_1, it can not tell that the false forwarding was induced by a joining message to which SRC (note that E may have sent joining messages to many different SRCs). Even if E somehow correctly sets S_1 as the destination of the pruning messages, there are multiple paths from E to S_1. If E let pruning messages to S_1 take E-C rather than E-B as the reverse path, the pruning operation still can not block the mis-delivered traffic actually along B-E. With limited knowledge, the destination SRC of the upstream pruning message could not be properly set, and the upstream

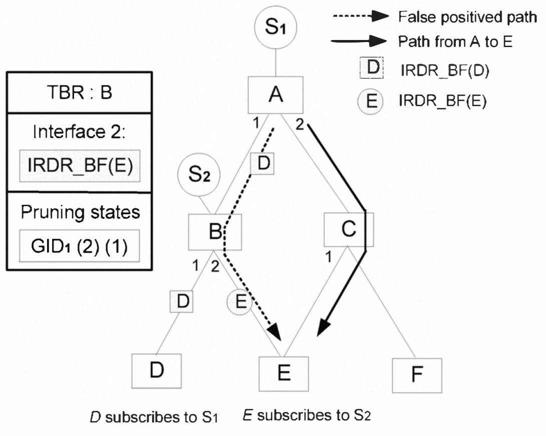


Fig. 5. Pruning false tree branches.

path could not be determined with the impact of asymmetric routing.

We design the pruning message propagation scheme as follows. In the first hop, the destination domain receiving falsely forwarded packets just sends a pruning message upstream through the interface where the false packets come. The pruning message carries the GID associated with falsely forwarded group. When the pruning message reaches an intermediate router, the false packets will be still coming to the router. By checking the incoming interface for packets associated with the tagged GID, the intermediate router can then easily identify the next upstream hop. Note that the destination domain receiving false packets will keep sending pruning message upstream, according to a certain schedule such as sending one pruning message after receiving n (≥ 1) false packets, until the false packets stop coming.

To facilitate the pruning process, each router involved also manages pruning states. When the pruning message arrives at an intermediate router for the first time, the router creates a pruning state in the format of $\mathrm{GID}_i(\mathcal{F})(\mathcal{N})$, where GID_i is the group identifier of the mis-delivered packets, $\mathcal{F}$ is the set of output interfaces that have falsely forwarded packets with GID_i, and $\mathcal{N}$ is the set of output interfaces that are normally forwarding packets with GID_i. If there are subsequent pruning messages associated with GID_i coming from another interface, the corresponding interface will be moved from set $\mathcal{N}$ to the set $\mathcal{F}$. In the example shown in Fig. 5, $\mathcal{N} = 1$ and $\mathcal{F} = 2$, which means that the interface 1 of B is normally forwarding packets of GID_1; because otherwise there should be another pruning message from interface 1. In this way, B could block the traffic with GID_1 towards E and keep forwarding that towards D.

B. Operations on Pruning States

The pruning states can help a TBR to properly determine whether to continue forwarding the pruning messages upstream, stop the forwarding, or remove the obsolete pruning states. Specifically, a TBR will take the following options upon receiving a pruning message associated with GID_i:

- Continue forwarding upstream the pruning message, if $\mathcal{F} \neq \phi$ and $\mathcal{N} = \phi$. This condition indicates that the TBR is an intermediate router along the false-forwarding branch, so it needs to continue forwarding the pruning message upstream towards the root node that generates the false-forwarding traffic.
- Stop forwarding upstream the pruning message, if $\mathcal{F} \neq \phi$ and $\mathcal{N} \neq \phi$ or the TBR is an SDR of a source domain. The first part of the condition, i.e., $\mathcal{F} \neq \phi$ and $\mathcal{N} \neq \phi$, indicates the current TBR is the root node that generates the false positive match, so there is no need to further forward the pruning message upstream. The branches in set $\mathcal{N}$ indicates the normal paths that correctly forward traffic for group GID_i, while branches in $\mathcal{F}$ indicates paths due to false positive match. Such a situation could only be possible that the current router receives correct traffic but false positive matching happens in the forwarding stage, that is, the current TBR is the root for false positive traffic. The second part of the condition is due to the possibility that the pruning message may go up to the SDR in the source domain if the false positive happened due to the mismatch when checking GRP_BF against the local channel list.

It is not difficult to see that the pruning state associated with a GID on an interface should be removed, if the destination domain ending the false-forwarding path now actively requests traffic from this group. Thus, when a destination domain needs to join a new group, it first check whether the group was involved in the false positive situation before. We let the RDR keep a record of the false positive GIDs it has observed. If the new active GID is found in the record, it needs to use a joining message carrying explicit GID and a pruning-removing flag to remove the pruning states on related interfaces. The destination address of such a joining message is set as the SRC address associated with the GID and then follows the DOM joining procedure (note that BGP-view based joining is applied in the asymmetric case [19]). When a TBR/SDR receives a joining message with a pruning-removing flag, if it has a pruning state associated with the indicated GID, it then just remove it. Note that such a pruning removing procedure is efficient, which just remove the pruning states on related hops that might impact the normal forwarding. Considering the false positive probability is small, the joining messages and computing overheads in intermediate TBRs associated with these falsely-delivered groups will not impact the scalability much.

C. Complete Loop Elimination

Theorem 2 The false tree branch pruning scheme can stop the false traffic and eliminate forwarding loops incurred by the false positive in Bloom filter matching under the condition that the domains associated with the SDR and RDRs are stub domains of the multicast group.

Proof: Note that any IRDR_BF on a certain interface was placed by a joining message from a destination domain; reverse to the path of the joining message is a data forwarding path to the destination domain according to the

DOM design for both symmetric and asymmetric cases [19]. Therefore, for given IRDR_BF that is falsely incorporated into the updated DST_BF, there must exist a TBR in the loop, which has a branch leading to the destination domain that generated the join message associated with the IRDR_BF under consideration. So the redundant traffic due to false positive can definitely be detected by that destination domain and incurs subsequent pruning messages. According to the pruning mechanism design, the pruning message will finally reach the root node that originated the false positive traffic and stop the traffic. Thus all the false positive traffic downstream and the forwarding loop if any will be totally eliminated. ■

Consider the example shown in Fig. 2, the false positive matching is initiated at the router R, and a forwarding loop is further formed as shown in the dashed line. In this example, the router R' has a branch inducting the false positive traffic to a destination domain. The destination domain will then identify the situation of false positive and generates the pruning messages. The pruning messages will reach the root node R of the false positive traffic and stop the loop.

V. Performance Evaluation

We use NS2 [22] simulation results combined with numerical analysis to demonstrate the performance of DOM with the proposed loop prevention scheme. The network topology for simulation is given in Fig. 6, which is widely used in the literature as a hypothetical US backbone network [21]. In our model, the source and transit domains are represented as backbone routers or backbone nodes; similarly, regional and organization autonomous systems (ASes) are represented as designated border routers in those domains. The backbone router providing connection service to regional ASes is termed as *access router* or *access node*, and regional ASes are connected by organization ASes, as illustrated in Fig. 6. We simulate multi-source multi-group scenarios. SRCs are attached to SDRs denoted using black nodes with each SRC providing 500 groups. The routing of the network is purposely configured asymmetric. For example, the arrows in the figure indicate the joining paths of nodes 0, 1, 6 and 27 to $SRC1$, while the highlighted tree shows the actual forwarding paths to them. We will focus on the effect of false-positive forwarding on DOM, because the mis-delivered traffic is the root cause of forwarding loop.

A. Memory Overhead under False Positives

We first evaluate the scalability of DOM with comparison to other multicast schemes. We connect each reginal AS node with 16 RDRs, and let all 16 RDRs subscribe to all groups provisioned by both SRC1 and SRC2. Then we study the impact of false positive forwarding on memory overhead of DOM. In the simulation, the false positive forwarding scenario is configured as following: We let all 16 RDRs subscribe to SRC2 and only 2 of them to SRC1. This setting could establish many IRDR_BFs along links in the backbone network, which increases the probability that the SRC1 data packets are falsely forwarded to SRC2 subscribers. To make the scenario more

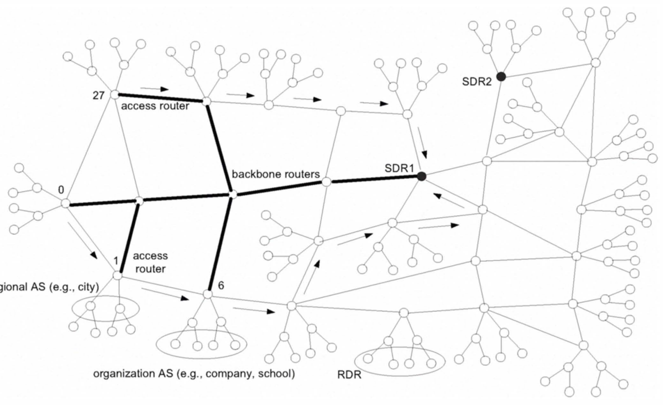


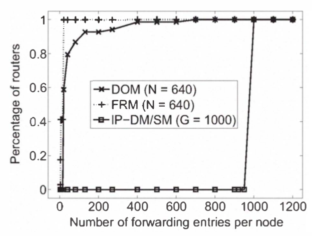
Fig. 6. Simulation topology.

extreme, we configure the false positive rate for Bloom filter matching to around 20% according to equation (1) in Section III-A. The memory overhead is measured by counting *the number of forwarding entries at backbone routers and regional AS border routers that are involved in the multicast.*

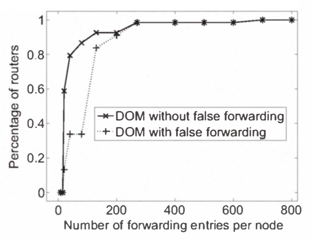
The cumulative distribution function (CDF) of the number of forwarding entries per node for the multicast schemes under study are illustrated in Fig. 7 (a). The forwarding state of DOM with false-positive forwarding is made up of the IRDR_BF; thus the number of forwarding entries per node is the number of RDRs whose joining paths pass through this node. The maximum number of states at a node is equal to the maximum number of subscribing RDRs; therefore, all nodes have entries no greater than 640 in Fig. 7. The forwarding state of FRM [13] is in fact its AS neighbor edges and hence the number of forwarding entries per node is the AS degree of this node. There is no node which has more than 20 forwarding entries for FRM. IP multicast Dense Mode (IP-DM) and Sparse Mode (IP-SM) have fixed number of forwarding entries per node, because the number depends on the number of groups active ($G = 1000$) in the network.

Fig. 7 (a) shows that DOM can significantly reduce the number of forwarding states stored at each node compared with IP multicast. This is because DOM states are destination-specific, instead of group-specific. Consequently, the number of forwarding states per node for DOM is independent of the number of groups being supported by the node. DOM requires comparatively more forwarding states than FRM does; however, the states maintained can greatly benefit the bandwidth efficiency, which is to be discussed in Section V-B.

The impact of false-positive forwarding on DOM is illustrated in Fig. 7 (b). For the DOM with false-positive forwarding, the states $\mathrm{GID}_i(\mathcal{F})(\mathcal{N})$ for pruning false tree branches should also be counted. Fig. 7 (b) shows that the pruning states has limited influence on the scalability of DOM. This is because the pruning states will be finally installed on the root node along the false forwarding path. In this experiment, 14 of the RDRs under a reginal AS are candidates for receiving unrequested data from SRC1, but there still 2 RDRs are normal subscribers. This means that the pruning states are primarily installed at each regional AS node. The



(a) Scalability of DOM



(b) DOM with and without false positive forwarding

Fig. 7. CDF of the number of forwarding entries per node.

pruning states has not been propagated into the core network, as the membership status of subscribers is not changed. We create this setting because this is an adverse case in the distribution of the pruning states. If the scalability of DOM is not serious impacted in this case, the performance for false tree branch pruning will be better in other scenarios. Imagine that if the pruning states are moving towards the backbone network, some of them will merge at some upstream node according to the pruning states operation. Although the pruning states are related to the number of falsely supported groups, the number of resulted states are still much less than that in IP multicast. The scalability of DOM remains with the false tree branch pruning operations.

B. Bandwidth Overhead under False Positives

We examine the bandwidth overhead incurred by data forwarding in this section. The DOM forwarding incurs bandwidth overhead due to three reasons: i) each packet needs to carry a DST_BF as destinations information; ii) when the number of destination domains is too large, multiple DST_BFs have to be used to cover all the destinations, which will generate redundant traffic; iii) false-positive forwarding traffic. For a fixed-length Bloom filter, the more elements are encoded, the higher the false positive rate can be. In DOM and FRM, when the number of receivers/branches exceeds the capacity of a single in-packet Bloom filter, multiple packets are sent to cover all destinations, which are counted as redundant traffic. Since covering the same number of destinations normally requires more branches, DOM can generate less redundant traffic than FRM does if they keep the same false positive rate. The false-positive forwarding in DOM will be handled by the proposed false tree branch pruning scheme.

In this experiment, we still maintain the false positive forwarding configuration, and focus on the multicast tree rooted at SRC1. A clip of MPEG-4 video stream is delivered from SRC1 to a number of subscribing RDRs ranging from 16 to 80. We will study the performance of DOM with the impact of false-positive forwarding, compared to other schemes without the impact, in order to verify the effectiveness of the false tree pruning operations proposed in the paper. The simulation results show that the false positive forwarding does not affect the advantage of DOM in bandwidth utilization over FRM, with the proposed false tree pruning operations. The bandwidth

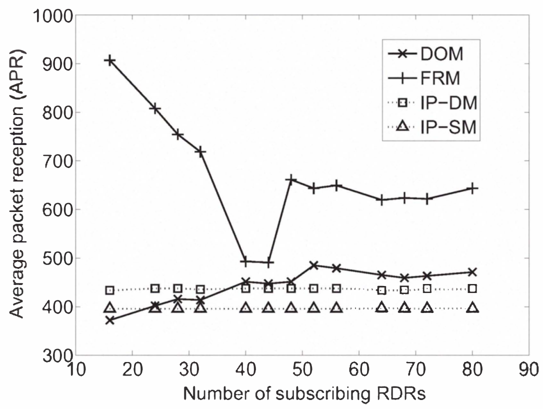


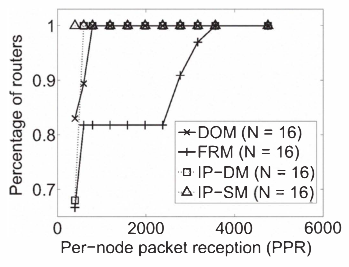
Fig. 8. Average packet reception (APR).

consumption of DOM are primarily caused by the packet-carried DST_BF and the redundantly-transmitted traffic. The specific results and explanations are shown below.

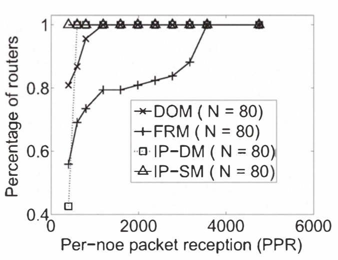
Average Packet Reception (APR) – The average packet reception (APR) is defined as *the average number of packets received by each non-RDR router in the multicast tree in 10 seconds*. Fig. 8 shows the APR versus N (the number of RDRs involved in multicasting) with different multicast schemes compared. We observe that DOM achieves the performance close to IP multicast, and even outperforms IP multicast when N is very small, as the IP multicast has the tree maintenance overhead. DOM has a better performance than FRM, especially when the multicast subtree along one interface of the SDR has many branches. This is because, for FRM, more packets have to be generated to carry subtree branches to cover all destination RDRs.

Compared with IP-DM, DOM does not use the "broadcast-and-prune" [10] method to maintain the tree structure. In the sparse scenario with small N, the larger value of APR for IP-DM is due to packet transmissions during the "broadcast-and-prune" operation. The performance of IP-SM is closely related to the selection of rendezvous point (RP). In the simulation, we select the backbone router at the geometry center of the topology as the RP for IP-SM scheme. The data packets are first unicast to the RP and then disseminated to RDRs from the RP, while DOM can deliver packets to RDR directly. This is why DOM can perform even better than IP multicast when $N = 16$.

The reason for the better performance of DOM over FRM is that DOM needs to encode less elements in the Bloom filter than FRM does. For example, consider the case that node SDR1 disseminates data to access routers $\{0, 1, 6, 27\}$; the multicast tree is highlighted in Fig. 6 with thick lines. For illustration purpose, consider that the in-packet Bloom filter can only encode 4 elements. In such a scenario, one filter can encode all the 4 destinations under DOM. Under FRM, the 8-branch tree needs to be encoded, which exceeds the capacity of one Bloom filter. Therefore, 4 Bloom filters over 4 packets have to be used, each containing the tree branches to one of the destinations. Three of such four packets are counted as



(a) N = 16



(b) N = 80

Fig. 9. CDF of per-node packet reception(PPR).

redundant traffic compared to DOM. In addition, we observe that the APR of FRM is decreasing when N is between 16 and 50. This is because the RDRs in these cases are concentrated on subtrees sourced from different neighbor edges of the SDRs. The FRM in-packet Bloom filter happened to be able to contain all branches of each subtree and the number of nodes receiving exact 2 packets increases. However, when N keeps increasing, the number of branches in the subtrees go beyond the capacity of the Bloom filter again, and the number of redundant packets increases. Thus the FRM curve presents the shape of a funnel.

Per-node Packet Reception (PPR) Distribution – The per-node packet reception is *the number of packets received at a given node when multicasting from the SRC to all receivers for 10 seconds*. Fig. 9(a) and (b) plot the CDF of the PPR for different multicast mechanisms when $N = 16$ and $N = 80$, respectively. In DOM, about 80% of the nodes receive less than 400 packets in both scenarios. There is no node receiving more than 600 packets in the case of $N = 16$ and about 1200 packets in the case of $N = 80$. The redundant traffic in DOM is incurred by splitting the destination set into smaller sub-sets to fit into the DST_BF. The falsely forwarded traffic in DOM is effectively blocked. In FRM, only about 66% and 55% of the nodes receive less than 400 packets in the two scenarios, respectively. The largest number of packets a node can receive is up to 3500. In IP-DM and SM, almost every node receives exact less than 400 packets, except for a few that receive some redundant packets in the tree maintenance or the source-to-RP unicast. The number of nodes receiving no redundant packets under DOM is close to that under IP multicast and is larger than that under FRM.

VI. Conclusions and Future Work

In this paper, we have tried to resolve the forwarding loop issue of the Bloom filter based multicast protocols in the context of a destination-oriented multicast (DOM) scheme. After the development of a theoretical analysis of the loop issue in DOM context, we have revealed that the DOM design natively supports automatical elimination of permanent forwarding loops in all cases except one termed as conservation of bits. Based on this conclusion, a probability upper bound on loop occurrence in DOM has been presented. Furthermore, we have proposed an accurate tree branch pruning scheme, which equips the DOM the capability to completely and efficiently remove the false-positive forwarding loop. We have presented simulation results over a practical topology to demonstrate the performance of our design, with comparison to a representative Bloom filter based multicast scheme FRM and traditional IP multicast. Multicast over wireless networks is an important area [23], [24]. In the future work, we will study how to enhance our loop mitigation technique to wireless networks.

References

[1] I. Stoica, T.S.E. Ng and H.Zhang, "REUNITE: a recursive unicast approach to multicast," in *Proc. IEEE INFOCOM*, 2000, pp.1644-1653.

[2] L. Costa, S. Fdida and O. Duarte , "Incremental service deployment using the hop-by-hop multicast routing protocol," *IEEE/ACM Trans. Networking*, vol. 14, no. 3, pp.543–556, Jun. 2007.

[3] T. W. Cho, M. Rabinovich, K.K. Ramakrishnan, D. Srivastava, Y. Zhang, "Enabling content dissemination using efficient and scalable multicast," in *Proc. IEEE INFOCOM*, 2009, pp.1980-1988.

[4] R. Boivie *et. al.*, "Explicit multicast (Xcast) basic specification," Internet draft, Mar. 2001.

[5] S.Deering and D.Cheriton, "The PIM architecture for wide area multicasting," *ACM Trans. Computer Systems*, vol. 8, no. 2, pp. 85–110, May. 1990.

[6] A. Adams, J. Nicholas and W. Siadak *et. al.*, "Protocol independent multicast-dense mode (PIM-DM): protocol specification (Revised)," IETF RFC 3973, Jun. 1998.

[7] D. Estrin *et. al.*, "Protocol independent multicast-sparse mode (PIM-SM): protocol specification," IETF RFC 2362, Jun. 1998.

[8] Y. Rekhter *et. al.*, "A Border Gateway Protocol 4 (BGP-4)," IETF RFC 1771, Mar. 1995.

[9] B. Cain, S. Deering, I. Kouvelas, B. Fenner, A. Thyagarajan, "Internet group management protocol, version 3," IETF RFC 3376, Oct. 2002.

[10] K.C. Almeroth , "The evolution of multicast: from the MBone to interdomain multicast to Internet2 deployment," *IEEE Network*, vol. 14, no. 1, pp.10–20, Jan.-Feb. 2000.

[11] S. Fahmy and M. Kwon , "Characterizing overlay multicast networks and their costs," *IEEE/ACM Trans. Networking*, vol. 15, no. 2, pp.373–386, April 2007.

[12] Y. Chu, S. Rao, S. Seshan, and H. Zhang, "Enabling conferencing applications on the Internet using an overlay multicast architecture," in *Proc. ACM SIGCOMM*, 2001, pp. 55–67.

[13] S. Ratnasamy, A. Ermolinskiy and S. Shenker " Revisiting IP multicast, " in *Proc. ACM SIGCOMM*, 2006, pp. 15–26.

[14] P. Jokela, A. Zahemszky, C. E. Rothenberg, S. Arianfar, and P. Nikander, "LIPSIN: Line Speed Publish/Subscribe Inter-Networking," in *Proc. ACM SIGCOMM*, 2009, pp. 195–205.

[15] A. Broder and M. Mitzenmacher, "Network applications of Bloom Filters: a survey," *Internet Mathematics*, vol. 1, no. 4, pp.485-509, May. 2004.

[16] M. Särelä, C. E. Rothenberg, T. Aura, A. Zahemszky, P. Nikander and J. Ott, "Forwarding Anomalies in Bloom Filter-based Multicast," in *Proc. IEEE INFOCOM*, 2011, pp. 2399–2407.

[17] X. Tian, Y. Cheng, K. Ren, and B. Liu, "Multicast with an application-oriented networking (AON) approach," in *Proc. IEEE ICC*, 2008, pp.5646–5651.

[18] X. Tian, Y. Cheng, and B. Liu, "Design of a Scalable Multicast Scheme with an Application-Network Cross-Layer Approach," *IEEE Trans. Multimedia*, vol. 11, no. 6, pp.1160–1169, Oct. 2009.

[19] X. Tian, Y. Cheng, and X. Shen, "DOM: A scalable multicast protocol for next-generation Internet," *IEEE Network*, vol. 24, no.4, pp.45–51, July 2010.

[20] X. Tian, Y. Cheng, and B. Liu, "A Fast-Join Mechanism for Inter-domain Multicasting," in *Proc. IEEE GLOBECOM*, 2010, pp. 1–5.

[21] R. Doverspike, G. Li, K. Oikonomou, K.K. Ramakrishnan and D. Wang, "IP backbone design for multimedia distribution: architecture and performance," in *Proc. IEEE INFOCOM*, 2007, pp. 1523–1531.

[22] The Network Simulator - ns-2, 2011, http://www.isi.edu/nsnam/ns

[23] X. Li, "Multicast capacity of wireless ad hoc networks," *IEEE/ACM Trans. Networking*, vol. 17, no. 3, pp. 950–961, Jun. 2008.

[24] X. Wang, L. Fu and C. Hu, "Multicast Performance with Hierarchical Cooperation," *IEEE/ACM Trans. Networking*, vol. 20, no. 3, 2011.