Imprecise Multicast Routing for Scalable Information Distribution

Samphel Norden
Applied Research Laboratory
Washington University in St. Louis
St. Louis, MO 63130
USA
samphel@arl.wustl.edu

Marcel Waldvogel
IBM Zurich Research Laboratory
Säumerstrasse 4 / Postfach
8803 Rüschlikon
Switzerland
mwl@zurich.ibm.com

Abstract—**Typically, multicast data distribution uses rendezvous points (PIM, CBT), multicast distribution tree building protocols, and multicast forwarding. Whereas the first two approaches have been extensively studied, scaling multicast forwarding state without increasing forwarding complexity has not been addressed in detail. Having a scalable strategy for aggregation of multicast forwarding state is essential for inter-domain multicast which could have any number of concurrent multicast groups, especially in applications such as event notification and web cache invalidation mechanisms. We first present the essential characteristics of a scalable multicast routing mechanism. We then introduce and analyze, according to these metrics, a scalable aggregation mechanism for multicast-based update and change distribution based on imprecise (too generous) aggregation. Our mechanism is simple to implement, requires no additional information about the groups, and allows important savings in routing table size and routing protocol overhead, at a minimal expense in additional network and end-system traffic.**

I. Introduction

Scalable data distribution is currently a major obstacle for the future growth and prosperity of the Internet. Popular servers and their network connection are constantly under heavy load, with peaks quickly exceeding their abilities. This often results in their owners and operators increasing their hardware pools, both in terms of processing power and network attachment. In other cases, the problem is outsourced to solution providers that have already set up hardware pools and bought network connectivity. A prominent example of such a provider is Akamai. Unfortunately, at some point the investment in resources to handle peak processing will become an obstacle in itself. Thus, other solutions have to be found.

Considerable effort has been put into Web caching and cache synchronization [1, 2]. Still, Web caches, unless forcibly imposed on customers by their Internet Service Provider (ISP), are rarely used. A main reason for this is the high latency involved in determining whether the cached document is still up-to-date. Thus caches often provide no visible benefit for the end user. Changing from the polling-based checking by the cache (“pull”) to a server-driven update (“push”) would not scale well and is therefore not available.

Other potential applications for scalable push technologies include distributed file systems such as the Andrew File System [3], group collaboration and discussion forums, and mechanisms for news distribution to end systems (e.g., Pointcast).

Currently, even if such technologies are sometimes labeled as “push,” they are mostly implemented by a client program polling the server, therefore resulting in exactly the same scalability problems mentioned earlier. This is further intensified by the fact that clients potentially need to poll frequently to become aware of updates before their data becomes stale. Only rarely will the clients be informed via call-back mechanisms, as this requires the server to keep track of the Internet Protocol (IP) addresses of interested clients. Besides the scalability issues, the wide-spread use of dynamic IP addresses assigned by ISPs and the ever-increasing popularity of Network Address Translation boxes (NAT [4]), long-term storage of IP addresses for call-backs becomes troublesome. Multicast comes to mind as a solution to transmit update notifications or even the entire document. Although multicast is currently only deployed in limited ways, it is gaining popularity, is readily available in its native fashion in many research backbones, and is being offered by an increasing number of ISPs.

A closer look, however, reveals a problem: Choosing a single multicast group to transmit all notifications and changes is very convenient at the sender side, but the subscribers to this service will receive a lot of unwanted traffic. Providing a multicast group per document is optimum in terms of receiver data rate, but requires the network (and potentially also the receivers) to keep track of an uncountable number of multicast groups, posing several scalability problems.

The remainder of the paper is organized as follows. Section II outlines general requirements for data and change distribution schemes. Section III discusses work that has been done on aggregating multicast routing information. Section IV describes IMPRESS, our implementation of a scalable multicast aggregation. The simulation environment is introduced in Section V and the simulation results are presented in Section VI. The paper is concluded in Section VII.

II. Data Distribution Requirements

In this section, we present seven requirements which we believe to be necessary and sufficient for fast, scalable, and efficient data distribution techniques. The first six items on this list relate to creating well-behaved protocol, the last three items are necessary to deploy this in an existing world, making it interoperable with existing protocols and routers.

Data Bandwidth Efficiency The bandwidth used to distribute the information to the subscribers should not be significantly increased, if at all.

Routing Message Traffic The size, number, and frequency of routing messages should not increase significantly, making sure that routing message traffic does not block data traffic.

Routing Message Handling Processing signalling traffic is a major task for Internet routers. Increasing the amount of traffic and/or the cost of handling them significantly would cause the route processing subsystem to become overloaded and thus disfunctional.

Router Memory Efficiency The amount of memory required at the router, both for managing routing information as described above and for performing the actual per-packet forwarding operation, should remain small.

Simple Forwarding The addition of such a technique should not cause the packet forwarding mechanism to become significantly more expensive. E.g., switching from a one-dimensional prefix matching to a multi-dimensional matching mechanism should be avoided. Even though the latter have improved significantly recently [5–7], they remain significantly more expensive.

Conserves Address Space The currently available multicast address space should be used efficiently. While the 2^{27} addresses available to multicast applications may seem vast, wasteful handling, such as assigning a multicast group to each of the estimated 200 million web pages, will quickly exhaust them without leaving space for future growth and other services. Also, current multicast address allocation services [8] require the address space to remain sparsely populated.

Non-consecutive Address Space The protocol should also not rely on having large consecutive chunks of address space, as multicast addresses are currently allocated randomly to provide some degree of uniqueness [8].

Downward Compatible The new protocol should be downward compatible to routers that do not understand the new mechanisms. Of course, these legacy routers will not be able to take advantage of some of the performance improvements. Without this compatibility requirement, another layer of overlay networks would be required on top of the existing MBone (Multicast Backbone) overlay.

In this paper, we present a scheme which provides for significant improvements when rated according to the above categories.

The paper is organized as follows. Section III reviews existing work for scalable data distribution. Section IV presents our mechanism. Section V defines the environment used for the simulations, which are discussed and evaluated in Section VI. Section VII concludes this paper.

III. Aggregating Multicast Groups

As of now, only a limited amount of work exists on aggregating multicast routing information. The only direct comparison possible is the leaky aggregation scheme proposed in [9], which groups adjacent multicast addresses into a prefix. This operation is similar to the unicast prefix aggregation in CIDR [10]. Their scheme requires routers to perform an expensive second *longest matching prefix lookup* on the multicast group address, in addition to the standard lookup on the source address. This significantly increases the forwarding cost for every multicast packet. This approach is further constrained by introducing a distinction between *high- and low*-bandwidth groups, a moving target in the Internet world. Aggregation is performed only for low-bandwidth groups. One option would be multicast senders must either be modified to include their data rate in a signaling message[1], possibly requiring modifications in all routers. Instead, [9] proposes expensive data rate measurements to be performed at every router, which increases the computational load during packet forwarding even further. Rather than performing the operation at each router, a complex scheme of staggering the bandwidth estimation across Autonomous Systems (BGMP [11] "domains"), and distributing the resulting information using the inter-domain multicast routing protocol (BGMP) is proposed. This is accompanied by the unwanted side effect of poor responsiveness to new groups and group changes. Every change in the routing protocol requires work proportional to $O(\log n)$, where n is the number of groups known to the router.

[1] Currently, IP multicast senders do not send any signaling messages

Another relevant approach that directly addresses the problem of reducing the number of multicast forwarding entries is described in [12] that uses tunnels to bypass routers that are not directly connected to any multicast group. However, this requires modifications to the routing protocol, along with additional encapsulation and de-encapsulation for the tunneling.

The scheme described in [13] performs aggregation only on the multicast *forwarding* state, which does not affect multicast routing state and protocol overhead. It does so on a lossless basis, and reaches an aggregation factor of four and often higher, depending on the scenario. Owing to its implementation (switching from a global to a per-interface table), the actual memory savings in environments where multiple interfaces are controlled by a single forwarding engine could be less than that, or even negative.

IV. Imprecise Multicast using IMPRESS

In this section, we describe the IMPRESS approach that uses imprecise (aggregated) routing table information in order to minimize the forwarding state while also minimizing the wastage of traffic that is sent on links that terminate in non-group members. First, we setup our network model and describe an application for IMPRESS. We then present two schemes that aggregate forwarding information while minimizing the traffic sent unnecessarily.

A. Network Model

The network topology that we assume is a source-based tree. We assume that all routers in the tree are multicast routers. Realistically, we assume some tunneling between multicast routers if the routers are not directly connected. However, this information is abstracted out, and only the core distribution tree is shown in Figure 1.

The application we consider is multicast for sending cache invalidation messages. Essentially, there are N_d documents at the root or source of the distribution tree. For each document d_i, there are N_{d_i} members that subscribe and need to receive messages to invalidate their caches when the document is modified. Thus, a group member at the leaf nodes of the tree will want to subscribe to notifications for each of the documents it currently holds.

The initial setup phase consists of installing forwarding information at routers when members join a particular group. Each document has an unique ID with a good statistical distribution. If necessary, this can be achieved by passing to ID through a hash function before using it.

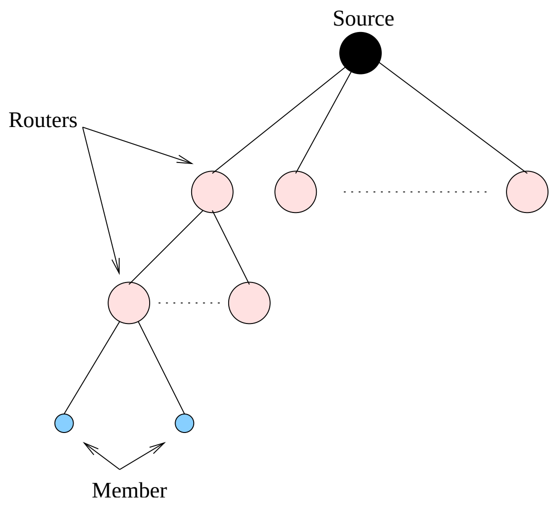

Fig. 1. Multicast Distribution Tree

TABLE I
BASELINE IMPRESS ALGORITHM

```
Baseline IMPRESS
Event A: Init
$BM_i$ = Random($D_i$) // Pick random bits from $Doc_d$
Event B: JOIN($D_i, H_j$) Host $H_j$ subscribes to group $D_i$
Add $D_i$ to $H_j$ doc list
Setup Forwarding State on Routers $R_r$ on path to src
        Init: $R_r$ = PARENT($H_j$)
        INSTALL $BM_i$ at Router $R_r$
        $R_r[BM_i] \rightarrow$ fwd = $H_j$ // Set up forwarding entry
        Set $H_j = R_r$; $R_r$ = PARENT($H_j$)
        REPEAT till $R_r$ = src
Event C: LEAVE($D_i, H_j$)
Remove $D_i$ from $H_j$ doc list
Remove forwarding state at routers on path to src
        Init: $R_r$ = PARENT($H_j$)
        IF $R_r[BM_i] \rightarrow$ fwd = $H_j$ AND MultipleBitsSet()
            DELETE $R_r[BM_i] \rightarrow H_j$ AND STOP
        ELSE
            DELETE $R_r[BM_i] \rightarrow H_j$
            Set $H_j = R_r$; $R_r$ = PARENT($H_j$)
            REPEAT till $R_r$ = src
```

The forwarding state maintained at routers maps the document ID to outgoing links, on which messages pertaining to this document should be forwarded to. The portion of the document ID used for the actual look-up is called the *bit mask* for a particular document. These bit masks are set up for all documents that exist at the source. The bit mask is used to extract a subset of the bits from the document ID.

At the end of the setup phase, all routers have bit masks installed for routing the appropriate documents to the leaf members. In the subsequent sections, we describe how IMPRESS aggregates state information at routers so as to scale with large numbers of concurrent active multicast groups.

B. IMPRESS

We will first describe the baseline version of IMPRESS that attempts to achieve the maximum aggregation possible while minimizing the *wastage* of traffic. The key idea here is to have all routers pick the same bit mask, *i.e.*, they pick the same subset of bits as an index into the forwarding table. The root or source of the multicast group picks the bits that decide the bit mask randomly. The basic algorithm is as shown in Table I.

C. *Discussion*

Each router maintains a mapping table indexed by the bit mask, resulting in a bit map of size k (recall that the degree of the node is k), where the bits indicating which output ports should receive the message received. Each time a host JOINS, one of the bits that points to that particular host H_j is set. Effectively, for a given JOIN operation, assuming a tree topology with height $O(\log n)$, each router performs $O(1)$ memory accesses leading to an overall setup time of $O(\log n)$.

Similarly on a LEAVE, we unset the bit corresponding to that subscriber. However, as can be seen in *Event C*, the LEAVE message is passed up the tree only when the router has no more children interested in these messages, as indicated by the list of active output links for that index becoming empty. The LEAVE operation requires $O(1)$ effort per router, leading to a worst-case time of $O(\log n)$ for the entire tree.

Using the same bit mask at It should be noted that the bit masks are chosen by randomly choosing some bits from the document ID and the same value is used at all routers that lead to a member for that document. This could potentially lead to large number of collisions, and hence lead to a potentially large amount of wasted traffic. Also, there is a bit mask for each document which is not a scalable alternative if the number of documents are large. This baseline algorithm will also be referred to as IMP-A.

D. *Randomized* IMPRESS

Except for very large bit masks, which will waste exponential amounts of router memory, the extraction of a subset of the ID according to the bit mask will map multiple IDs onto a single index, resulting in a *collision*. In order to prevent these collisions from causing many messages to be sent along a subtree with no subscribers, we apply a small variation in the above algorithm, which we call *Randomized* IMPRESS. Essentially, instead of globally choosing the bits when setting up the bit masks, each router at configuration time randomly picks an individual bit mask. Now, when it receives a multicast message with a given document ID, the router looks at the bit positions it had determined at configuration time. If there is match, then the router forwards the message on the corresponding interface. Note that the bit mask is still a per-node variable and remains independent of the document ID. By randomizing the bit masks and having no per-document state, we reduce the collision probability as well as maintaining scalability. The ideal case has $\log_k n$ bit of bit mask for a k-ary multicast tree with n subscribers or leaves. One of our goals is to evaluate the performance gap between the ideal case with $\log_k n$ bit masks and a scenario where we use a smaller number of bits (aggregated state). In the simulations below, we compare the amount of routing state with the “link

TABLE II
SIMULATION PARAMETERS

Parameter	Symbol	Explanation
Bit Mask	BM	Number of bits to represent document
Hosts		Total number of potential members
Density	D	Number of hosts per document
Degree		Degree of the k-ary distribution tree
Link Wastage	LW	# Links used unnecessarily
Link Utilization	LU	# Links used to transmit useful traffic
Relative Performance	RP	LW/LU

wastage," *i.e.,* the amount of data sent on links with no members subscribing to that message.

E. Addressing

To store the document ID in the message, we propose to assign a multicast address range, where the least significant bits of the address are used to store the document ID. The source address would be used to distinguish between multiple groups, an approach borrowed from [14]. This approach is especially suited for use with baseline IMPRESS, as only the few bits required for the pre-masked document ID need to be stored. For longer IDs, the use of an IP option or a shim document-naming layer could be used. Randomized IMPRESS will also be referred to as IMP-B.

V. SIMULATION ENVIRONMENT

We simulated a multicast distribution tree with all documents originating from the source or root of the multicast tree. Each document had a *density*(number of hosts subscribing to the document) that was an exponentially distributed random variable. We simulated two sets of experiments, for 2000 hosts and 2000 documents, and for 10000 hosts and 10000 documents. The distribution tree is actually a k-ary tree with degree k which is fixed to 5 unless otherwise specified. The document ID is assumed to consist of 64 bits. The simulation parameters are given in Table II.

From Table II, we use two metrics to evaluate our scheme. The link wastage counts the number of links on which messages were sent without members to receive the messages. The link utilization is closely related and counts the number of links that were used to transmit useful traffic. Note that this is not necessarily equal to the difference between the total number of links and the wastage as there could be links that did not carry any messages throughout the simulation. We also define the term *Relative Performance* to be the ratio of LW/LU. The lower the performance ratio, the better the real performance.

We also show the total link utilization $(LU+LW)$ to indicate the total *message traffic*. The relative performance that indicates the *data bandwidth efficiency* is also shown.

VI. SIMULATION RESULTS

The results presented in this section are grouped in three parts. The first part focuses on comparing Baseline and Randomized versions of IMPRESS. The second part concentrates on varying simulation parameters for Randomized IMPRESS. The third part explores the impact of weighting the LU and LW with the density. The most important metrics are those of *utilization* and

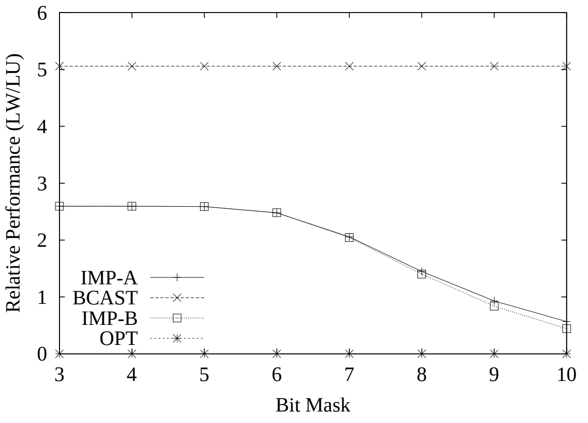

Fig. 2. Comparing IMP-A & IMP-B: RP vs. BM (200 Hosts/Document)

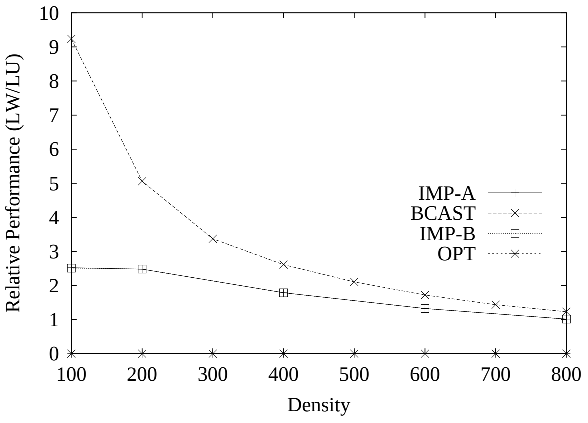

Fig. 3. Comparing IMP-A & IMP-B: RP vs. Density (BitMask=6)

wastage. To put them into perspective, they are compared with two other, simple scenarios: BCAST shows the behavior when no filtering is in place and all data is sent to every node, resulting in the biggest wastage possible, whereas *opt* is the optimal case, where every document is only sent on links which have subscribers below. In other words, BCAST is IMPRESS with $BM = 0$, and OPT has $BM \geq \log_2 D$.

A. IMPRESS *Variants*

In this section, we focus on the different variants of IMPRESS, the baseline and the randomized approaches.

From Figure 4, we see that the total routing message traffic decreases with increasing bit mask for IMP-A and IMP-B as compared to the BCAST scheme. At a bit mask of 6 bits, where we are using roughly 50% of the address space bits[2], or 3% of the router memory required without aggregation, the total traffic is close to BCAST. However, slight increases in address length quickly lead to improved utilization.

Figure 2 shows the relative performance (LW/LU) versus the bit mask. As the size of the bit mask increases, performance improves, because there is less aggregation. The IMPRESS schemes require only two to three times as much traffic than the optimal

[2]The number of bits required to address 2000 documents deterministically is 11. Please note, that saving x bits of the address results in a memory reduction by a factor of 2^x.

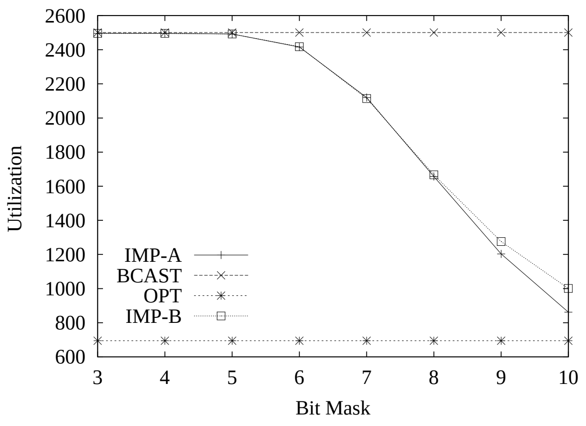

Fig. 4. Comparing IMP-A & IMP-B: Utilization vs. BM (200 Hosts/Document)

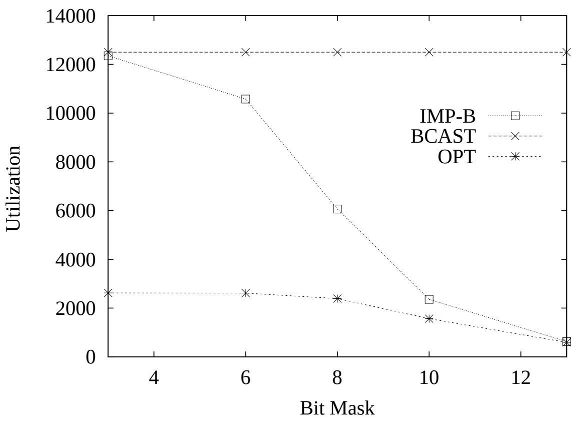

Fig. 6. Increased Receiver Set: Total Usage vs. BitMask (100 Hosts/Document)

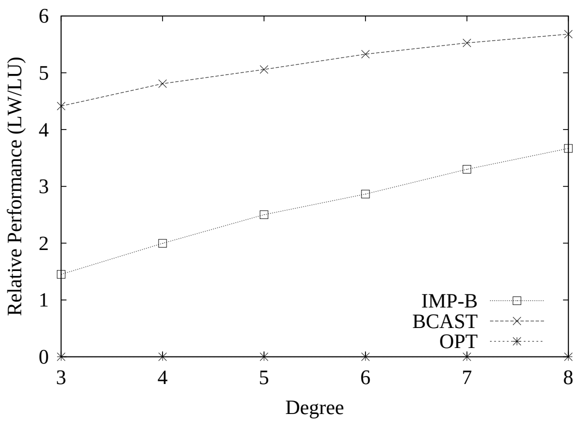

Fig. 5. Performance of IMP-B: RP vs. Degree (200 Hosts/Document,BM=6)

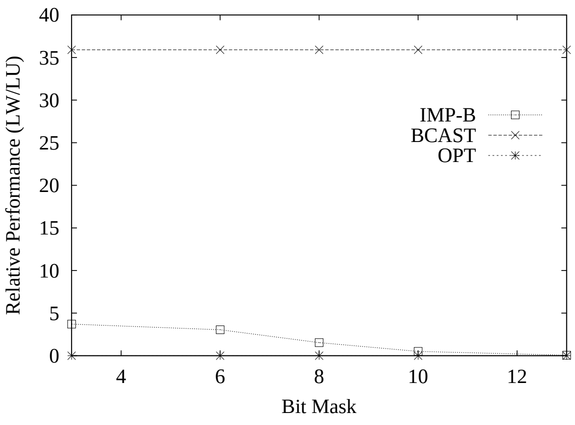

Fig. 7. Increased Receiver Set: RP vs. BitMask (100 Hosts/Document)

case, while reducing memory usage ten- to hundred-fold.

At very high densities, when almost all hosts subscribe to a given document, the BCAST scheme actually improves so that it behaves similar to the IMPRESS, as is shown in Figure 3. In summary, the relative benefit of the IMPRESS scheme over BCAST occurs at around 70% usage of the address space, and is more prominent for lower densities.

Both IMP-A and IMP-B show similar, sometimes identical, performance. This shows that even though randomization seems to have an advantage, the much simpler JOIN/LEAVE protocol and bookkeeping of IMP-A can be used without great sacrifices.

In terms of the data distribution requirements that were introduced in Section I, the router memory can be used efficiently, as only a single indexed lookup is needed per packet and no complex data structures are required, especially when compared to other mechanisms such as the one described in [9]. The mechanisms that we use also conserve address space where the bit mask results showed that reasonable performance was possible even with only small amounts of memory, when compared to the OPT case.

B. *Analysis of Randomized* IMPRESS

In this section, we focus our attention the randomized IMPRESS scheme, IMP-B. We study the impact of a number of simulation parameters such as the degree of the distribution tree, as well as of adopting a larger topology by increasing the number of documents and hosts to 10000 each.

Figure 5 shows the impact of varying the degree of the k-ary distribution tree on randomized IMPRESS. As the degree increases, the path length from the source to the leaves/hosts shrinks. However, this increases the size of the output link bitmap at each router, requiring more memory. However, IMP-B provides a factor of 3 improvement over BCAST at a bit mask of 3 bits, and a factor of 2 improvement at higher bit masks.

C. *Impact of the Number of Receivers*

Figures 6 and 7 show the results for the larger case of 10,000 documents and hosts. The interesting result here is that the behavior for the smaller topology scales proportionally for this larger case as well. Additionally (Figure 6), we see that in such large topologies, the improvements of IMPRESS over BCAST are much more prominent, with IMP-B performing very close to OPT. The relative performance ratio for IMP-B is more than an order of magnitude better than that for BCAST, even when only half the address bits are used (see Figure 7).

D. *Weighted Randomized* IMPRESS

In this section, we discuss the effect of weighting Randomized IMPRESS (both LU and LW) with the document density as explained earlier in this section. There are two important effects that we seek to capture in these graphs. The first is the effect of some documents having larger density also having more impact on the wastage and utilization since every wasted link now contributes more if weighted by the density. Likewise, every utilized link will contribute more to the overall utilization if the density of that document is more. Thus, we could uniformly weight the utilization and the wastage by the density in order to capture this effect. The other effect that we seek to capture is the clustering effect as to when density only affects the utilization and not the wastage. This is realized when there are several clusters of members such that the wasted links are confined to some contiguous area and are in some sense unaffected by the weighting using the density leading to a non-uniform weighting scheme.

To recapitulate, we define relative performance in this context as:

$$RP_{eqwt} = \frac{\Sigma LU[i] \times D_i}{\Sigma LW[i] \times D_i}$$

where D_i is the density of document i, LW[i] and LU[i] are the link wastage and utilization for document i respectively. We call this the uniform weighting scheme in the figures.

Another approach to weighting the parameters is to weight the LU alone and normalize it as follows:

$$LU_{wt} = \frac{\Sigma(LU[i] \times D_i)}{N_{docs} \times \hat{D}}$$

$$RP_{uneqwt} = \frac{LU_{wt}}{\Sigma LW[i]}$$

where D_i is the density for document i, $\hat{D}$ is the mean density for all documents, and N_{docs} is the total number of documents. We call this the non-uniform weighting scheme in the figures.

The third scheme we compare is the case when:

$$RP_{base} = \frac{\Sigma LU[i]}{\Sigma LW[i]}$$

when there is no weighting.

We study the relative performance (RP) with variation in the bit mask of the distribution tree as shown in Figure VI-D. We see that RP_{eqwt} is better than RP_{uneqwt} which in turn is better than the baseline RP_{base} version.

VII. Conclusions

In this paper, we have described the problem of multicast forwarding entry state aggregation. We presented IMPRESS, an extremely simple aggregation approach, that is very easy to implement. Despite its simplicity, it strongly reduces the network state that needs to be maintained at routers. Unlike many other, more complex proposals, it does this without increasing protocol processing burden at the routers; it even simplifies the lookup compared to other systems. Experiments were performed to obtain the optimum operating point that also minimizes the *wastage* on links due to traffic being sent to non-group members. The simulation results show that we can restrict the wastage to acceptable levels while maximizing the "lossy" aggregation. As part of future work, we will be using a real MBone map topology for our experiments. We will also work on removing the current restriction of an underlying virtual tree topology in order to make IMPRESS into a scalable multicast routing protocol in its own right.

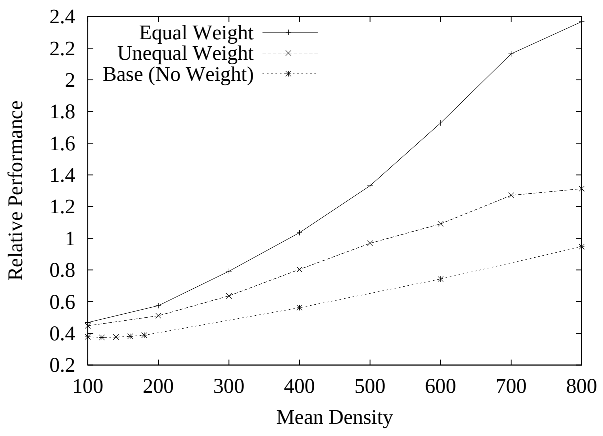

Fig. 8. Impact of Weighting for RP vs. D (BM = 6)

References

[1] Duane Wessels and K. Claffy, "Internet Cache Protocol (ICP), version 2," Internet RFC 2186, Sept. 1997.

[2] Li Fan, Pei Cao, Jussara Almeida, and Andrei Broder, "Summary cache: A scalable wide-area web cache sharing protocol," in *Proceedings of ACM SIGCOMM '98*, Sept. 1998.

[3] Edward R. Zayas, "AFS-3 programmer's reference: File server/cache manager interface," Tech. Rep. FS-00-D162, Transarc Corporation, Pittsburgh, PA, USA, Aug. 1991.

[4] K. Egevang and Paul Francis, "The IP network address translator," Internet RFC 1631, May 1994.

[5] V. Srinivasan, George Varghese, Subhash Suri, and Marcel Waldvogel, "Fast and scalable layer four switching," in *Proceedings of ACM SIGCOMM '98*, Sept. 1998, pp. 191–202.

[6] T. V. Lakshman and Dimitrios Stiliadis, "High speed policy-based packet forwarding using efficient multi-dimensional range matching," in *Proceedings of ACM SIGCOMM '98*, Sept. 1998, pp. 203–214.

[7] Milind M. Buddhikot, Subhash Suri, and Marcel Waldvogel, "Space decomposition techniques for fast Layer-4 switching," in *Proceedings of the IFIP Sixth International Workshop on Protocols for High Speed Networks (PfHSN '99)*, Salem, MA, USA, Aug. 1999, pp. 25–41.

[8] Mark Handley, "Session directories and Internet multicast address allocation," in *Proceedings of ACM SIGCOMM '98*, Sept. 1998.

[9] P. I. Radoslavov, D. Estrin, and R. Govindan, "Exploiting the bandwidth-memory tradeoff in multicast state aggregation," Tech. Rep. 99-697, Department of Computer Science, University of Southern California, Los Angeles, CA, USA, July 1999.

[10] V. Fuller, T. Li, J. Yu, and K. Varadhan, "Classless Inter-Domain Routing (CIDR): an address assignment and aggregation strategy," Internet RFC 1519, Sept. 1993.

[11] Satish Kumar, Pavlin Radoslavov, David Thaler, Cengiz Alaettinoğlu, Deborah Estrin, and Mark Handley, "The MASC/BGMP architecture for inter-domain multicast routing," in *Proceedings of ACM SIGCOMM '98*, Sept. 1998, pp. 93–104.

[12] J. Tian and G. Neufeld, "Forwarding state reduction for sparse mode multicast communication," in *Proceedings of IEEE Infocom '98*, Mar. 1998.

[13] David Thaler and Mark Handley, "On the aggregatability of multicast forwarding state," in *Proceedings of IEEE Infocom 2000*, Mar. 2000.

[14] Hugh W. Holbrook and David R. Cheriton, "IP multicast channels: Express support for large-scale single-source applications," in *Proceedings of SIGCOMM'99*, August 1999.