

Efficient Dissemination of Personalized Information Using Content-Based Multicast¹

Rahul Shah, Ravi Jain and Farooq Anjum

Applied Research, Telcordia Technologies, 445 South St, Morristown, NJ 07960

Abstract-- There has been a surge of interest in the delivery of personalized information to users (e.g. personalized stocks or travel information), particularly as mobile users with limited terminal device capabilities increasingly desire updated, targeted information in real time. When the number of information recipients is large and there is sufficient commonality in their interests, as is often the case, IP multicast is an efficient way of delivering the information. However, IP multicast services do not consider the structure and semantics of the information in the multicast process. We propose the use of *Content-Based Multicast (CBM)* where extra content filtering is performed at the interior nodes of the IP multicast tree; this will reduce network bandwidth usage and delivery delay, as well as the computation required at the sources and sinks.

In this paper we evaluate the situations in which CBM is advantageous. The benefits of CBM depend critically upon how well filters are placed at interior nodes of the IP multicast tree, and the costs depend upon those introduced by filters themselves. Further, we consider the benefits of allowing the filters to be mobile so as to respond to user mobility or changes in user interests, and the corresponding costs of filter mobility. We consider two criteria: minimizing total network bandwidth utilization and minimizing mean information delivery delay. For each criterion we develop an optimal filter placement algorithm, as well as a heuristic that executes faster than the optimal algorithm. Finally, we evaluate all the algorithms by means of simulation experiments.

Our results indicate that filters can be effective in substantially reducing bandwidth and delay. We also find filter mobility is worthwhile if there is sufficient locality in the interests of users, or there is marked large-scale user mobility. We conclude with suggestions for further work.

Index terms—IP multicast, Personalized information, mobile computing

1 INTRODUCTION

Future information systems will increasingly need to deliver the results of diverse applications to growing numbers of users. While the information can be unicast to all the users, it is clear that multicasting the information offers a better alternative considering the commonality in the information desired by the various subscribers. Therefore, we believe that multicast will increase in importance at all levels of information and networking systems. A case in point is the delivery of

personalized information to users, particularly mobile users who desire real-time targeted information (e.g. stocks or traffic information.) Multicast is also used in the signaling and management protocols of networks, e.g. for congestion control [Kasera00]. It is also being used at the middleware level, e.g. in event subscription and notification mechanisms for Java and CORBA [OMG97]. And of course, multicast applications like Internet radio, chat, and conferencing are proliferating rapidly.

Along with this increase in the use and importance of multicast, we also observe a need to generalize the notion of multicast to bring its advantages to bear on more sophisticated and specialized uses. Traditional or *basic IP multicast* consists of a set (or *group*) of participants, of which typically one is a source of information and the other are sinks; any information generated by the source is delivered to the rest of the group by setting up a *multicast tree*. Typically the size of a group is more than a few hosts and much less than the total number of hosts in the system, and basic multicast is preferable to unicast (to each recipient) or broadcast (to all hosts) in such situations.

However we observe that basic multicast does not concern itself with the content or structure of the information being delivered. Any structure in the information can only be accommodated either by using the concept of multiple layers in the same multicast group (e.g. as commonly proposed for video applications) or by setting up multiple multicast groups. The former requires the data to be highly structured, and typically the structuring is such that the recipient cannot receive any arbitrary layers it chooses to. As an example of the latter, multicast groups corresponding to different sources (e.g. Internet radio channels or chat servers) can be set up. In that case, once a recipient joins a group he receives all the information sent to the group; the groups can be set up to avoid the need for any further filtering or control by the end-user's application or the user herself. Unfortunately this approach may require a number of groups exponential in the number of data items; in the worst case, $\min(u, 2^m)$ groups for u users and m distinct data items, which can lead to scalability problems. On the other hand using fewer groups leads to unnecessary network traffic over the network and filtering effort at the end points (possibly duplicated at several end-points.)

As an example, consider disseminating real-time information such as traffic for an area, current news on various topics like sports, financial, technical, weather etc. as video clips. Using basic multicast the source has a limited set of options. One option is to transmit all the video clips on a single multicast channel. Another is to divide the video clips into a limited number of groups (each with roughly equal number of

¹ Copyright Telcordia Technologies Inc.

recipients, say) where each recipient typically subscribes to multiple groups, filters out unnecessary information from each group, and collates remaining clips into a personalized result. Thus for a given set of information being disseminated, providing finer-grained filtering increases the number of multicast groups, which increases the burden on the recipient (the application or user) as well as the network to manage group membership. On the other hand, if fewer groups are used, the recipient has to filter out a large amount of unnecessary information and network bandwidth is utilized inefficiently.

The limitations of basic IP multicast become much more severe once the recipients desire more complex filtering and personalization, and especially if the information being delivered is unstructured or has limited meta-data. For instance, one recipient of an Internet radio channel may desire to block out all advertisements, except those for cars, which she is currently in the market for; another likes country music but cannot abide certain artists or songs; a third wants news but no more about the latest celebrity scandal; and so on. Even if the information is well-structured, e.g. consider a NYSE stock quotes example, users can desire complex filtering e.g. "Inform me only if IBM stock changes by more than 10% in price or 5% in volume compared to yesterday". Thus, in this paper we propose to use IP multicast as an underlying mechanism for personalized information delivery but provide solutions to address some of the limitations of this approach.

We propose the paradigm of *Content-Based Multicast (CBM)*. CBM takes into account the structure and semantics of the information being disseminated and attempts to minimize both network utilization as well as the processing at the recipients by intelligently filtering the information as it propagates towards each recipient. The information is disseminated using IP multicast. We consider a solution where the filtering is done by means of *filters*, which can potentially be *mobile*, residing at (a subset of) the intermediate nodes in the network, either in IP or application-level routers or at attached CBM servers. A filter can then apply complex criteria and ensure that information propagates down the tree to a child only if a user at a leaf in that child's subtree desires the information. As users move or change their filtering criteria, or as the information being disseminated changes, filters may move from one interior node to another in response. While filters thus increase multicast efficiency they also introduce additional computation, complexity and delay in the multicast process, and hence the number of filters in the network should be limited.

1.1 Example

We use the IP multicast tree of Figure 1 to illustrate the benefits of CBM. The subscription requests of each recipient (leaf), labeled A-I, for items (numbered 1 to 8 in this example) are shown along the bottom of the tree. These subscriptions propagate up the tree; interior nodes are labeled with the union of the subscriptions from the leaves in their sub tree. The source sends the 8 (equal-sized) items of information

requested by at least one recipient. With basic IP multicast, all 8 items are sent on all 15 links, resulting in 120 units of total traffic. Each recipient then must filter out all undesired items. If filters are placed at every interior node of the tree, the traffic can be reduced to 47 units, and the recipients need not perform any filtering.

However, recognizing that filters themselves represent a cost (since filters imply additional processing at a node), suppose we restrict the number of filters (in this case, arbitrarily to 3), and place them at the shaded interior nodes. Then the total traffic is reduced to 68, and some recipients e.g. A-D, must perform filtering, while others e.g. E-I, need not. Now suppose some users leave, move or join the tree, or their subscriptions change; the total traffic will change. This raises the questions we address in the rest of this paper: (1) how should a limited number of filters be placed to be most effective, and (2) as conditions change, is it worthwhile to move filters in response?

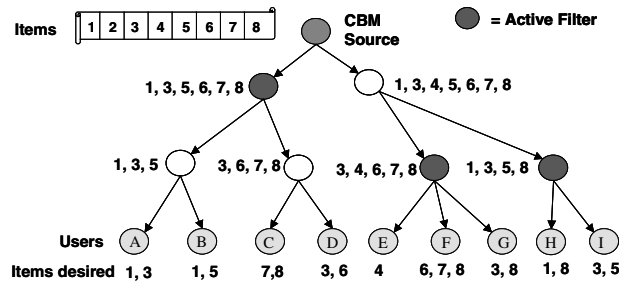


Figure 1: Example of decreased bandwidth utilization with filters (shaded vertices)

Thus, as seen from the example, CBM reduces network bandwidth and recipient computation at the cost of increased computation in the network. CBM is increasingly important as the quantity and diversity of information being disseminated in information systems and networks like the Internet increases, and users suffering from information overload desire personalized information. In addition, CBM is increasingly important as wireless networks (ad-hoc networks, multi-hop packet radio networks, wireless LANs and wide-area cellular networks) proliferate, since wireless bandwidth is scarce and resources (in terms of computation, storage and battery life) at wireless devices are limited.

The benefits of CBM depend critically upon how well filters are placed at interior nodes of the multicast tree, and the costs depend upon those introduced by filters themselves. Further, there is a tradeoff between the benefits of allowing the filters to be mobile so as to respond to user mobility or changes in user interests, and the corresponding costs of filter mobility.

The contributions of this paper can be summarized as follows. We consider two objectives for developing filter placements: minimizing total network bandwidth utilization and minimizing mean information delivery delay. For each objective we develop an algorithm for filter placement on the IP multicast tree, show that it is optimal, and derive its time

complexity. For each objective we also develop a heuristic that executes faster than the optimal algorithm. Finally, we evaluate all the algorithms by means of simulation experiments. Note that in this paper we do not focus on the process of setting up the underlying basic IP multicast tree itself, and assume that an appropriate method has been used for this purpose.

This paper is organized as follows. In sec. 2 we look at the related work. The system model is described in sec. 3, and the two filter placement objectives are explained in sections 4 and 5. In sec. 6 we develop heuristics corresponding to our optimal algorithms. In sec. 7 we describe results from simulation experiments to evaluate the optimal and heuristic algorithms. Finally in sec. 8 we end with a summary and our conclusions.

2 RELATED WORK

Work on personalized information delivery to large numbers of recipients via multicast has focused on two approaches: Application Layer Multicast and IP multicast. We briefly review the literature for each.

Application layer multicast [Chu00, Yoid00, Scatter00], a well-studied problem in the context of content distribution networks, provides multicast functionality at the application layer while assuming only unicast IP service at the network level. Note that CBM as proposed in this paper differs from application layer multicasting in that CBM proposes to enhance IP-multicast functionality by adding filters. The filters themselves might be active at the application level or at the IP level.

Application layer multicast has been applied to publish-subscribe systems where many publishers can send information to many subscribers with the matching being done by brokers in the network. In our context the brokers can be considered as the root and the nodes of the multicast tree. The publishers deliver their information to the root node while subscribers register with the root node of the multicast tree. [Aguiera98] addresses the problem of providing an efficient matching algorithm suitable for a content based subscription system. [Banavar98] addresses the problem of matching the information being multicast with that being desired by leaves. Their scheme assumes that the subscription pattern of users is static. SIENA [Carzaniga98] also proposes to address the concept of personalized notification (event) delivery using application layer multicast. The authors consider notification selection (filtering) explicitly and define a useful formal syntax and semantics for the filtering aspect of event notification service. We in fact assume a similar filtering mechanism in CBM. A related line of work is to use active network approaches (e.g. [Wen01] and references therein) which propose to build IP multicast services based on unicast forwarding and some support from the routers. Thus, similar to application layer multicasting, this approach provides for multicast functionality with unicast support only. We would

like to remark here that this is one way to implement CBM but do not pursue this further in this paper.

CBM can also basically be looked at as an enhancement to the functionality provided by IP multicast. We next look briefly at work related to this aspect. [Kasera00] proposes to use active elements in the interior of the network, as proposed in CBM, but these elements are assumed to be stationary and their objective is to address the reliability and congestion control challenges presented by IP multicast. [Donahoo98] consider enhancements to the usual IP multicast along three directions. At the network layer they try to investigate the relationship between the application performance and the placement of the root of the multicast tree. They propose algorithms to influence the root placement as well as to migrate the root over a period of time. They also consider modifications to applications to take advantage of network layer multicast. Finally they also study reliable multicast transport protocols.

[Opyr00] considers ways of minimizing the number of multicast groups in a basic IP multicast network either by judicious clustering of the users or by minimizing the amount of “waste” information sent to users. Thus this is a problem similar to the problem that we consider but the solutions that they investigate are within the confines of normal IP multicast without any enhancements. [Zhou00] proposes a variation of IP multicast in which the source decides the entities that will receive the multicast data stream. The decision is based on the data contents.

3 SYSTEM MODEL

We assume for simplicity that there is a single source of information connected by an IP network to a set of recipients. The source periodically receives updated information to be disseminated. We assume that an IP multicast tree has been set up using an appropriate protocol; the root is the source and the leaves are the recipients. For CBM we assume that there is a set of software modules, called *filters*, distributed at the interior nodes of this multicast tree. The filters reside at *filter platforms*, which can be IP routers or at servers attached to the local subnet of a router. In sec.3.1 we provide background on how the filters operate and how information is disseminated in the multicast tree. In sec. 3.2 we describe the framework for our algorithms for determining where filters should be placed in the IP multicast tree.

3.1 Subscription and Matching algorithms

Filters can be written in any suitable language that allows execution across filter platforms. The filters may move from one filter platform to another. Thus, for example, filters could be mobile Java applets or programs, which are very widespread and successful on the Internet, or they could be agents as in Concordia [Kob99] or Aglets [Tai99] systems. These filters could be created by the source of information or by a third party considering the content generated by the source. For convenience we assume there is always a filter at the root of the multicast tree.

The information disseminated by the source is described using a hierarchical information schema of major keys, minor keys and subkeys where all are triples of the form (*type, name, value*). As an example, an item of information for an IBM stock quote may be:

Major key: (*string, financial, NYSE*)
Minor key: (*string, symbol, IBM*)
Subkeys: (*float, prior_closing_price, 95*),
 (*float, current_price, 98*),
 (*float, 52_week_high, 110*)

Obviously various optimizations and operations can be performed in this schema [Carzaniga98] that we will ignore since this is not the focus of our work. Further, we assume that this information is available to the recipients through out-of-band channels (such as a predefined web address); we do not consider this further.

Each filter collects requests for personalized information (*subscriptions*) from its children in the tree. A subscription consists of the description of an item along with a *filter criterion* specifying operators on its values. For example, in addition to the data item specified above (IBM stock quote) the subscription request for a particular user *u* could say:

Filter: (*float, current_price, >, 100*)

indicating the recipient is to be notified if the current price exceeds \$100. Filter criteria can be combined with logical operators to make more complex filters.

The filter forwards subscriptions to its parent filter in accordance with a *Subscription Algorithm*. The filter maintains a subscription table of current subscriptions that it knows about and the child in the tree that each subscription originated from. If a new subscription request is subsumed by an existing subscription, the filter updates the table only; otherwise it makes a new entry in the subscription table and forwards the subscription to its parent.

Each filter sifts the information it receives from its parent filter in accordance with a *Matching Algorithm*. The filter compares the information item received with the subscriptions in its subscription table. It then composes a message to each child filter containing only the information (if any) matching the subscriptions originating from that child. For an example of a matching algorithm, see [Aguilera98].

Filters can be in one of four states: *active*, *migrating*, *spawning*, or *passive*. A filter performs the subscription and matching algorithms only in the active state. In the migrating state a filter physically relocates from one interior node to another. In the spawning state a filter either creates a new filter which will migrate to another node and become active, or will activate an existing filter at that node (by sending it an activation message). In the passive state the filter resides at a node and listens for activation messages from other filters but performs no subscription or matching algorithms.

The Subscription and Matching algorithms executed at the filters could be arbitrarily complex; but, as remarked earlier, this is not the focus of this study and hence we will not describe them further. Observe that they are distributed algorithms in the sense that each filter only operates using information in its local storage, and on-line algorithms in the sense that they are executed as each information item or subscription is received.

3.2 Filter placement algorithm framework

A *filter placement* on a multicast tree $M = (V, E)$ with vertex set V and edge set $E \subseteq V \times V$ is a set $P \subseteq V$ where filters are placed at all vertices in P and on no other vertex in V . Let $|V| = n$. The root of M is denoted w and $Tree(v)$ denotes the subtree rooted at vertex $v \in V$. Thus $Tree(w) = M$.

All the filter placement algorithms we discuss could in principle be executed at the source or at some other arbitrary server; for ease of exposition we assume they are executed at the source. Also note that the algorithms are centralized and operate upon the tree M that is in the source's memory.

Users may make or modify their subscriptions at any time, and these subscriptions propagate up the tree to the source in accordance with the Subscription Algorithm. The source collects the requests periodically into batches. At the end of each period the source runs the filter placement algorithm and calculates a new filter placement². It then sends signaling messages to the filters in the tree to activate, passivate, migrate or spawn filters as necessary. The source then multicasts information which is the union of the subscriptions in the current batch.

For simplicity we make the following assumptions in the rest of the paper: (1) each information item is of unit size; (2) the size of an information item is much greater than the size of a signaling message; and (3) the delay to send the filters signaling messages, and for them to respond with the appropriate action, is much less than the time to multicast the information required in a batch. Note that these assumptions are realistic especially when we consider bandwidth-intensive content like audio/video data.

Let $f(v)$ denote the information flow into a vertex v . The step of calculating the amount of information flow f required at each vertex based on the set of subscriptions B is as follows. For each leaf $v \in V$ the source calculates $f(v)$ as the size of the subscription request from that user. The source then repeats this process recursively up the tree. This information can be obtained by calculating successive unions. It can be calculated when the subscriptions are processed, with little extra effort. It

² For the heuristics described later, once filters are placed, all subscriptions need not be sent up to the source. The filters can carry out the necessary calculations and only send summary information to the source. For simplicity we omit description of this improvement.

is straightforward to see that this takes time linear in number of vertices.

We consider two objective functions to determine the effectiveness of filter placement. The *total traffic* T is the sum of the bandwidth used in the multicast tree to satisfy a given set of subscriptions. The *mean delay* D is the average delay from the instant the source multicasts the information to the instant that a leaf receives it.

4 MINIMIZING TOTAL TRAFFIC WITH k FILTERS

We develop an algorithm for finding a filter placement on a given IP multicast tree so as to minimize the total bandwidth consumption for a given set of subscriptions assuming required flow values are provided at each vertex. Filters cause overheads in terms of processing and delay since filters are assumed to act at a layer above the network layer. For the problem of minimizing total traffic we model this constraint as a limitation that at most k filters may be placed within the multicast tree, i.e., the resulting filter placement $|P| \leq k$.

Here for simplicity, we describe the algorithm for binary multicast trees; an extension to arbitrary trees is straightforward. For a vertex $v \in V$ let $L(v)$ and $R(v)$ denote the left and right child of v respectively. We also define the *Lowest Tight Ancestor (LTA)* of v as the lowest ancestor of v whose parent has a filter, and denote it $A(v)$. If parent of v has a filter, then $A(v) = v$. We also let $A(w) = w$ where w is the root of the tree.

Let $T(v, i, p)$ denote the minimum total traffic in $Tree(v)$ given that up to i filters can be placed in $Tree(v)$ and the LTA of v is $A(v) = p$. Thus the objective function is to minimize $T(w, k, w)$, where w is the root of M . (Recall that the source multicasts an item of information only if it is required by some user.) The total traffic can be expressed as the following recurrence relations, where for notational convenience we set $L(v) = l$, $R(v) = r$, $A(v) = p$.

If v is a leaf then

$$T(v, i, p) = 0 \quad \text{for all } p, i$$

Otherwise

$$T(v, i, p) = \min\{f(l) + f(r) + \underbrace{\min_{0 \leq j < i: T(l, j, l) + T(r, i - j - 1, r)}_{\text{if } v \text{ has a filter}}, 2f(p) + \underbrace{\min_{0 \leq j \leq i: T(l, j, p) + T(r, i - j, p)}_{\text{otherwise}}\}$$

Theorem 1. The minimum total traffic in the CBM tree rooted at w given that at most k filters can be placed in the tree is given by the recurrence relations for $T(w, k, w)$.

proof. The proof is by induction on the height h of the CBM tree. Clearly for $h = 0$, v must be a leaf so the total traffic is zero, which is minimum. For $h > 0$ consider an arbitrary vertex v (see Figure 2). The traffic entering its LTA is $f(p)$, since the filter at the parent of p will ensure that only the flow

required in $Tree(p)$ is sent to p . Now $f(p)$ will also be the incoming flow to v since there is no filter between v and p to reduce or modify the flow in any way. If a filter is placed at v it will ensure that only the flows strictly required in $Tree(l)$ and $Tree(r)$ are sent to the left and right child respectively where l and r indicate the left and the right vertex respectively; otherwise, both children will receive flow $f(p)$. The decision to place a filter at v is made such as to minimize the total traffic assuming that j ($j = 0, 1, 2 \dots$) filters are placed in the left subtree and the remaining filters are placed in the right subtree. Assuming minimality of values calculated at vertex l and vertex r by induction, we get the minimality of values calculated at v .

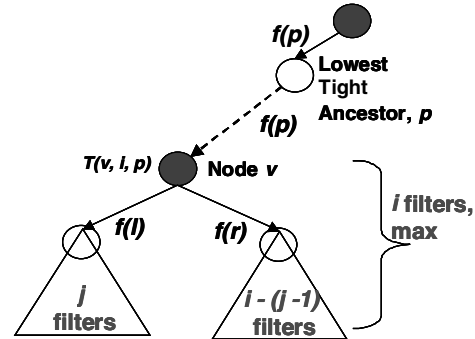


Figure 2: Filter placement to minimize total traffic

Theorem 2. The filter placement to achieve minimum total traffic in the CBM tree given that at most k filters can be placed in the tree can be computed in time $O(kn^2)$.

proof sketch. The optimal placement can be computed using dynamic programming. At each vertex a table of at most $O(kn)$ different values of T is maintained. To compute each value requires at most k comparisons. Since there are n vertices in M for which these tables are to be computed the total time required is $O(k^2n^2)$. However, observe that relatively few vertices in the tree have both left as well as right subtrees of size comparable to k (say at least $k/2$). Hence typically for most vertices we require much less than k comparisons. A careful analysis of this similar to the one for the k -median problem on trees [Tamir96] shows that the time complexity is $O(kn^2)$.

5 MINIMIZING MEAN DELAY

We now consider the benefits of filters in reducing the mean delay of delivering information to individual users. Filtering reduces transmission delay by reducing the amount of information transmitted to nodes downstream. However, the filters themselves introduce a delay, leading to a trade-off as to whether it is worthwhile to place a filter at a node. Note that if need be a cost function that minimizes both delay and bandwidth usage simultaneously can be devised but we choose not to do so for lack of space. Note that such a cost function would be similar to the generalized model of [Tamir96].

For simplicity we assume that (1) the delay on a link is proportional to the length of the message transmitted across the link, ignoring propagation delay and (2) the delay

introduced by a filter is a (typically large) constant, F . Once again, for simplicity we describe the solution for binary trees. The extension to the general case is straightforward and is omitted.

Let $d(v, u, g)$ be the delay for sending traffic g from vertex v to leaf u in the tree M , i.e., it is the sum of the transmission delays for sending g over each link in the unique path from v to u . Then, assuming no intermediate filtering, the total delay for sending traffic g from vertex v to $Tree(v)$ is the sum of $d(v, u, g)$ over all leaves u in $Tree(v)$. In the following we will consider the problem of minimizing total delay, which is simpler in terms of notation; mean delay is minimized also as a result.

Now consider a CBM with a required flow f known for each vertex in the tree. For a vertex v with LTA $A(v) = p$, let $D(v, F, p)$ denote the minimum total delay in $Tree(v)$, given that each filter introduces a delay F . Let $s(v)$ denote the number of leaves in $Tree(v)$, and c the link delay per bit. Then the minimum total delay can be expressed by the following recurrence relation, where $L(v) = l$, $R(v) = r$, $A(v) = p$.

If v is a leaf then

$$D(v, F, p) = 0 \text{ for all } p$$

Otherwise

$$D(v, F, p) = \min \left\{ \begin{array}{l} s(v)F + f(l) s(l) c + f(r) s(r) c + \\ \quad D(l, F, l) + D(r, F, r), \\ \quad \quad \quad \text{if } v \text{ has a filter} \\ f(p) s(v) c + D(l, F, p) + D(r, F, p), \\ \quad \quad \quad \text{otherwise} \end{array} \right\}$$

Theorem 3. The minimum total delay in the CBM tree rooted at w given that filters introduce a delay F is given by the recurrence relations for $D(w, F, w)$.

proof. The proof is by induction on the height of the CBM tree, and is similar to the proof for Theorem 1. We omit details for brevity.

Theorem 4. The filter placement to achieve minimum mean delay in the CBM tree given that filters introduce a delay F can be computed in time $O(n^2)$.

proof. The proof is similar to the proof for Theorem 2.

6 HEURISTICS FOR FILTER PLACEMENT

In the previous two sections we have described two algorithms that determine an optimal filter placement that run in time $O(kn^2)$ and $O(n^2)$ respectively. It is desirable to have filter placements calculated quickly so that the source can respond quickly to changes in user subscriptions. Also, when user subscriptions change, the optimal algorithm may require many of the k filters to move; since filter mobility has a cost, it is desirable to restrict it in some way. In the following we develop a framework for running heuristics, and introduce heuristics that run in linear time for each of our objective functions, but are sub-optimal.

6.1 Heuristic framework

The filter placement heuristics run in essentially the same framework as the optimal algorithms, with the exception that the heuristics take into account not only that filters have costs (which the optimal algorithms model as constraints on the number of filters k , or as a filter delay F), but that moving a filter has a cost. Thus while the optimal algorithms calculate a new filter placement for each batch, where potentially the positions of all filters may change, in the heuristics we allow only one filter to be moved for each batch. (Clearly this constraint could be generalized to allow some $i \leq k$ filter moves, but as we shall see from simulations, the single-move heuristics work sufficiently well.)

Both the traffic-reducing and the delay-reducing heuristics calculate a metric called Importance, $I(v)$ for each v in M . Informally, the Importance of a vertex is an estimate of the value of placing or removing a filter at that vertex in terms of the objective function (reducing traffic or delay). The importance of a vertex depends on the current filter placements in the tree. The Importance is easier and quicker to calculate than the calculations in the optimal algorithm.

For each batch the heuristic can be described as three steps.

1. *Importance flip.* Consider the most important vertex currently without a filter, v , and the least important vertex with a filter, u . (Ties are broken arbitrarily.) If moving the filter from u to v will reduce the total traffic, the heuristic does so.
2. *Parent-child flip.* If the Importance flip would not reduce the traffic, the heuristic considers each vertex with a filter whose parent does not have a filter (or vice versa). If moving the filter to the vertex without the filter would reduce traffic, the heuristic does so.
3. If neither flip would reduce traffic, the heuristic does nothing.

6.2 Heuristic to reduce total traffic

We now describe the specific operation of the traffic-reducing heuristic given the framework described above.

The Importance metric for the minimum traffic case is calculated as follows. Let the *weight* of a subtree $Tree(v)$ of M be denoted $z(v)$. The weight represents the number of edges in $Tree(v)$ affected by the placement of a filter at v . Then, the Importance of a vertex v is defined by the following recurrence relations, where $L(v) = l$, $R(v) = r$, $A(v) = p$.

$$\begin{aligned} I(v) &= (f(p) - f(l)) z(l) + (f(p) - f(r)) z(r) \\ &\text{where for a vertex } x \\ z(x) &= 1, & \text{if } x \text{ has a filter} \\ &= 1 + z(L(x)) + z(R(x)), & \text{otherwise} \end{aligned}$$

The intuition behind this Importance metric is sketched as follows. If a filter is placed at v then excess traffic will not be directed to l . (The same idea holds for r .) The amount of this excess traffic is $f(p) - f(l)$. If l also has a filter, then the savings will only apply to a single link, the link from v to l ;

otherwise, it will apply to all links from v until its next descendant that has a filter.

The intuition behind the Parent-child flip is sketched as follows. The Importance flip may not catch cases where moving a filter from a parent, say v , to a child, say r , is beneficial. Since there is no filter at r , $z(v)$, and hence $I(v)$, is greater than it would be otherwise. Further the filter at v means $f(r)$, and hence $I(r)$, is less than it would have been otherwise. These two effects together may incorrectly inhibit the movement of the filter from v to r .

Theorem 5. The execution time of the traffic-reducing heuristic is $O(n)$.

proof. The Importance for each vertex in M can be calculated using the recurrence relation above, and it is straightforward to show that no vertex is considered twice, so that the execution time for this calculation is $O(n)$. If an Importance flip has to be done, only the most important vertex without a filter and the least important vertex with a filter have to be found, which takes time $O(n)$, i.e., sorting all the vertices by Importance is not required. Once they have been found, determining whether flipping them reduces total traffic, given that all other filters remain unchanged, can also be done in time $O(n)$. The calculations for the Parent-child flip involve looking at each vertex and its parent only, which for each vertex is time $O(1)$, and so over the entire tree can be done in time $O(n)$.

6.3 Heuristic to reduce total delay

The delay-reducing heuristic operates in the same framework as the traffic-reducing heuristic. However, the Importance metric is calculated using the quantity $s(v)$, the number of leaves in $Tree(v)$, as follows. Once again $A(v) = p$.

$$I(v) = (f(p) - f(l)) \underline{z}(l) c + (f(p) - f(r)) \underline{z}(r) c - s(v) F,$$

where for a vertex x

$$\underline{z}(x) = s(x), \quad \text{if } x \text{ has a filter}$$

$$s(x) + \underline{z}(L(x)) + \underline{z}(R(x)), \quad \text{otherwise}$$

The intuition behind this Importance metric is sketched as follows. The quantity $\underline{z}(x)$ here denotes the sum of the lengths of parts of the paths from x to all leaves in $Tree(x)$ for the total delay case. If a filter is placed at v then excess traffic not directed to l is $f(p) - f(l)$, resulting in a reduction in delay of $(f(p) - f(l)) c$. (The same idea holds for r .) If l has a filter this reduced delay is enjoyed by all users in $Tree(l)$. If l has no filter, then the savings will also apply to all links from l until its next descendant that has a filter.

Theorem 6. The execution time of the delay-reducing heuristic is $O(n)$.

proof. Similar to Theorem 5, so omitted.

7 SIMULATION RESULTS

In the following we describe simulation experiments to evaluate the effectiveness of the filter placement algorithms and investigate the regimes where they are worthwhile. There are a large number of possible scenarios that can be simulated.

We have chosen several interesting scenarios that attempt to answer the following basic questions:

1. Can filters be effective in reducing traffic? If so, how does the number of filters used affect the traffic reduction?
2. What is the impact of patterns in user subscription on the effectiveness of filters in reducing CBM traffic?
3. How effective is the traffic-reducing heuristic as compared to the optimal algorithm?
4. Under what circumstances does filter movement make sense for reducing CBM traffic, and how much impact can it have?

We also carried out simulation experiments to answer the same questions for minimizing mean delay. However, we do not include all of them here. The traffic results are shown in subsection 7.1, and the delay results in subsection 7.2. Our experiments show that in most cases the traffic and delay cases are qualitatively similar.

All the simulation experiments were carried out for a complete binary tree of seven levels, i.e., $n = 127$ vertices and 64 leaves (users). We assume that the source has $m = 64$ distinct information items that users can choose from. The probability that a user i , $0 < i < 65$, chooses an item j , $0 < j < 65$, is denoted $P(i, j)$.

7.1 Minimizing total traffic

7.1.1 Effectiveness of filters in reducing total traffic

In this experiment users have uniform subscription probabilities, i.e., $P(i, j) = P$. We vary k from 0 to 63, and measure the total traffic after the optimal algorithm has placed the k filters. The total traffic is measured in terms of messages, where one information item traversing one link counts as one message. The results are shown in Figure 3.

We see that when filters are introduced ($k > 0$) the total traffic can drop sharply. Filters have most effect when subscriptions are sparse (P is low), because as P increases they have less and less work to do; when $P = 1$ all users want all items and CBM is the same as basic multicast. (Note that when subscriptions are very sparse there is a small dip in traffic for $k = 0$ because some of the m items are not requested by any user and the filter at the root node removes them.)

We also see that a small number of filters can have dramatic effects: with only 3 filters and subscription probabilities of less than 20%, traffic can be reduced by about 12-50%. There are diminishing returns as the number of filters is increased. We see that in this experiment with $k = 15$ filters we can obtain roughly 75% of the traffic reduction compared to $k = 63$ over a large range of subscription probabilities.

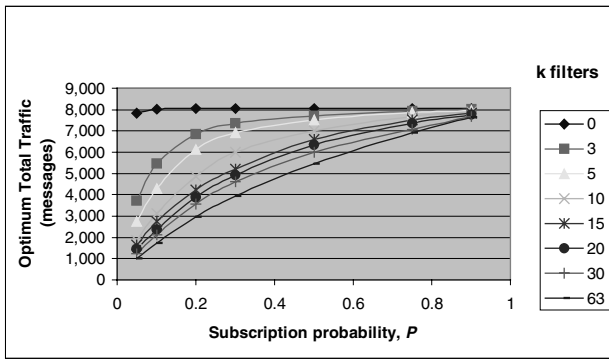


Figure 3: Total traffic with optimal placement of k filters, with uniform subscription probability

7.1.2 Impact of locality in subscriptions

We now consider the very likely case that user subscriptions are in fact not uniformly distributed, but that some kind of locality holds. We model locality in the following way: users that are close in the multicast tree tend to have similar interests, and a user's interests fall off rapidly in the surrounding items of interest. This locality model tends to capture a situation where locality in the multicast tree corresponds roughly to geographic locality (e.g. sibling leaves are physically close together) and users want location-based information (as is often the case with information such as weather, traffic, etc.) Clearly, other models of locality are also possible; we use this as one example.

The model sets $P(i, j) = 1/N$ if $i = j$, and q^r/N otherwise, where N is a normalization factor which ensures average probability is the chosen value a , q is a skew parameter and r is the height of the lowest common ancestor of i and j considering both i and j as leaves. Thus r is a measure of the distance between i and j in the tree. As explained earlier, j in reality represents an information item. The skew $q = 1$ if subscriptions are uniform; the lower the q , the greater the subscription locality. The different curves in Figure 4 correspond to different values of q for the case $a = 0.2$.

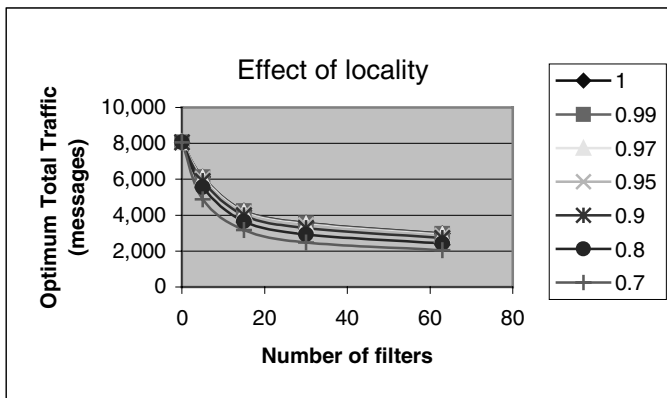


Figure 4: Effect of subscription locality q on total traffic with optimal filter placement

As before, in all cases, as the number of filters increases the total traffic drops, with diminishing benefits. Also, with

increasing locality filters are more beneficial, regardless of the number of filters. For $k = 3$ locality can decrease traffic further by about 15%, while at $k = 63$ almost 30% further reduction can be obtained.

7.1.3 Effectiveness of traffic-reducing heuristic

We next considered the effectiveness of the heuristic and compare it with the optimal algorithm, as follows. The optimal algorithm is run at the start of the experiment, and generates a placement, called *init*. For 49 subsequent time units (i.e., 50 batches total), user subscriptions are allowed to vary, and for each batch the heuristic and the optimal algorithm are compared. The heuristic is constrained to move only one filter at each time unit, while the optimal algorithm may move any number of k filters. The experiment is carried out with users having uniform subscriptions with $P = 0.3$, and $k = 15$ filters. At each time unit, 20% of the users (probabilistically) change their subscriptions.

In this experiment we observed that for uniform subscriptions the initial placement performs very well, and in fact neither the optimal nor the heuristic algorithm do much better, i.e., the total traffic is almost the same for *init*, the optimal algorithm and the heuristic. (Graphs corresponding to this are not shown for lack of space; we briefly describe the results instead.) The reason for this is straightforward: with uniform subscription probabilities, once filters have been placed what really matters is the quantity of information desired by each user; this is especially the case with relatively large subscription probability and relatively large number of filters. Thus, this suggests that once an optimal initial placement has been obtained we do not need precise filter placement to obtain good results provided the subscription rates change gradually. (At the same time we have seen that we need to use the optimal algorithm at least once and that totally random filter placement is very inefficient.) We also experimented with cases where the subscription rates change more rapidly. For such cases, we observed that with time the initial placement becomes more and more ineffective, while the optimal and heuristic algorithms performed very well. We also noticed that the heuristic performed almost as well as the optimal algorithm.

7.1.4 Effect of filter movement

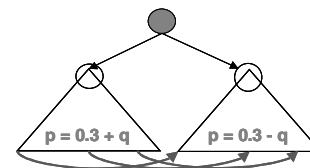


Figure 5: Commuter user mobility model

We now consider a situation where filter movement makes an impact on total traffic.

Commuter user mobility model. Assume at time $t = 0$, users in the left subtree have subscriptions with uniform probability $P = 0.3 + q$, while those in the right have $0.3 - q$ (See **Figure 5**). For $t = i > 0$, however, the probability of user i in the left

subtree is swapped with that of user i in the right. In this manner, probabilistically speaking, we model the movement of users from the left to the right (e.g. from work to home) as a change in subscription probabilities.

With this model, filter mobility becomes very important. Figure 6 shows the reduction in total traffic when filters are mobile as against static, as a percentage, for different values of skew q . When filters are static, they do not move from the nodes where they are placed initially. Note that in both cases, when filters are static and when they are allowed to move, the initial placement is decided based on the optimal algorithm. For our example tree, at time $t = 32$ users in the left subtree will have $P = 0.3 - q$ and those in the right have $P = 0.3 + q$, i.e., all users have moved. Denoting traffic at time $t = 32$ when the given number of filters are static as T_s and traffic at time $t = 32$ when filters are allowed to move as T_m , we plot $1 - T_m/T_s$ vs. the number of filters, k .

When $q = 0$ there is no skew and initial position of filters is good enough; there is little change in total traffic with filter mobility since subscriptions are uniform and stay so. However, as skew is increased, indicating more user mobility, the mobility of filters becomes more effective. As the number of filters is increased to 16, the figure shows that the percentage reduction due to filter mobility decreases. This is because for small values of k , the initial placement of the filters is mostly in the left subtree. Hence, when the users move to the right subtree, allowing the filters to move to the right subtree reduces the traffic substantially so that $T_m \ll T_s$. On the other hand with a medium number of filters ($k = 16$ for the example tree), the filters end up initially arranged at the middle level of the complete tree; this ends up being a very desirable arrangement for the special case $k = 16$ due to which T_m is close to T_s . As the number of filters is increased further, filters initially are placed not only at the middle level but also scattered in the left subtree. When users migrate to the right subtree, allowing filters to move once again provides benefits, so that $T_m < T_s$. Note that the traffic with large number of filters will be less than the traffic over the IP multicast tree when only a few filters are allowed but this is not what is shown in this figure.

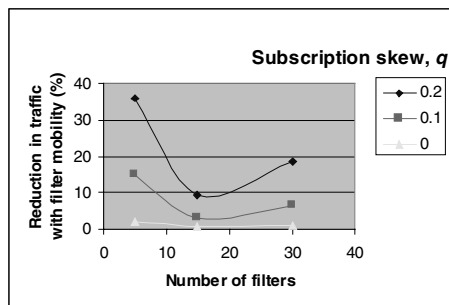


Figure 6: Effect of user mobility and filter mobility on total traffic for a given number of filters

7.2 Minimizing mean delay

We carry out the same experiment as in sec. 7.1, but for minimizing mean delay. Thus users have uniform subscription probabilities. To model the cost of filters, we vary the delay F introduced by a filter from 100 to 2000 time units. The results shown in Figure 7 are qualitatively similar to those for minimizing total traffic, and indicate that filters can be very effective. We have also observed that for low subscription probability and low filter delay many more filters are placed in the tree as compared to when the filter delay is high enough. Now as subscription probability increases the number of filters increases until about a subscription probability of 0.3 following which the number of filters starts decreasing. We do not show this and other results.

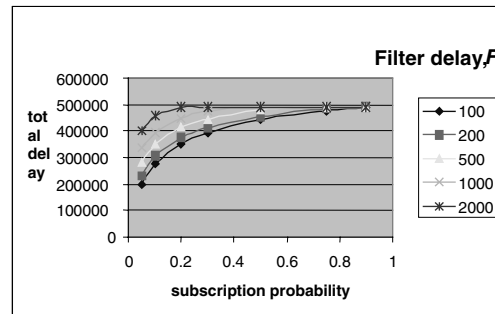


Figure 7: Effect of filters on total (hence mean) delay, for uniform subscriptions and varying filter delays

8 CONCLUSIONS

In this work we have considered the effect of using mobile filters for achieving personalized delivery using the paradigm of content-based multicast. Personalized information delivery has become an important and lucrative application-level service. CBM is also applicable to other important facilities, e.g. event subscription and notification. However, the effectiveness of CBM depends upon the placement of the filters, and their movement in response to changes in user subscriptions and motion.

We have developed optimal algorithms for placement of filters to minimize total traffic as well as to minimize the mean delay experienced by users in receiving information. Both algorithms run in time proportional to the square of the number of vertices in the multicast tree. We have then developed simple heuristics that run in linear time but which are sub-optimal.

Through simulation experiments we have demonstrated that, for the situations studied, *filters can be very effective in reducing total traffic as well as mean delay, and that a relative small number of filters can be very effective*. We also show that the *effectiveness of filters increases with locality in users' subscription patterns*. The simple heuristics work well, especially when user subscriptions are distributed uniformly and are relatively stable, but in extreme cases can cause poor performance. Finally, *filter mobility may not be beneficial in*

general; it is effective if there are large-scale changes in user subscriptions or in user locations.

Proposed work for the future includes further investigations into the costs associated with the instantiation and migration of filters as well as the delay caused due to processing by filters. An implementation using Aglets technology [Tai99] is also in progress.

Acknowledgments. Thanks are due to S. Rajagopalan of Telcordia for many interesting and useful discussions.

References

- [Aguilera98] Aguilera et al, "Matching Events in a Content-based Subscription System", <http://www.research.ibm.com/gryphon>
- [Banavar98] Banavar et al., "An efficient multicast protocol for content-based publish-subscribe systems", Technical report, IBM 1998.
- [Carzaniga98] Carzaniga et al, "Design of Scalable Event Notification Service: Interface and Architecture", Tech Report CU-CS-863-98, University of Colorado, Dept of Computer Science, 1998.
- [Chu00] Y. Chu, S. Rao and H. Zhang, "A case for end-systems only multicast," ACM Sigmetrics, June 2000.
- [Donahoo98] M. Donahoo, "Application-based Enhancement to Network-Layer Multicast." Ph.D thesis, Georgia Inst of Technology, 1998.
- [Kasera00] Kasera et al, "Scalable Fair Reliable Multicast Using Active Services", IEEE Network Magazine, Jan/Feb 2000
- [Kob99] Koblick R., "Concordia", Comm ACM, 42(3): 96-97, Mar 1999.
- [OMG97] Object Management Group, "CORBAServices: Common Object Services Specification", www.omg.org, December 1997.
- [Opyrchal00] Opyrchal et al. "Exploiting IP Multicast in Content-Based Publish-Subscribe Systems", Middleware 2000, LNCS 1795, pp. 185-207, 2000.
- [Scatter00] Y. Chawathe, S. McCanne and E.A. Brewer, "An architecture for internet content distribution as an infrastructure service", February 2000, unpublished work.
- [Tai99] Tai H. and K. Kosaka, "The Aglets project", Comm ACM, 42(3): 100-101, Mar 1999.
- [Tamir96] A.Tamir. "An $O(pn^2)$ algorithm for p-median and related problems on tree graphs", Operations Research Letters, 19:59-94, 1996.
- [Wen01] Wen et al. "Building Multicast Services from Unicast Forwarding and Ephemeral State", IEEE OpenArch 2001.
- [Yoid00] P. Francis, "Yoid: Extending the Internet multicast architecture", <http://www.yallcast.com/>, Sept 1999.
- [Zhou00] Zhou H. and Suresh Singh, "Content based multicast (CBM) in ad hoc networks", Proc. First Annual Workshop on Mobile and Adhoc Networking and Computing 2000, pp 51-60

Symbol	Denotes
M	Multicast tree
V	Vertex set
E	Edge set
n	V
P	subset of V such that all nodes in P contain filters
Tree(v)	A tree rooted at v
s(v)	number of leaves in Tree(v)
w	root of the multicast tree, M
f(v)	information flow into a vertex v
T	Total traffic
D	mean delay
L(v)	Left child of vertex v also denoted as l
R(v)	Right child of vertex v, also denoted as r
A(v)	Lowest tight Ancestor of vertex v
T(v,i,p)	minimum total traffic in Tree(v), given upto i filters in the tree and A(v)=p
h	height of the tree
F	delay introduced by a filter
d(v,u,g)	delay for sending traffic g from vertex v to leaf u in the tree M
D(v,F,p)	minimum total delay in the tree given that each filter introduces a delay F and A(v)=p
c	link delay per bit
I(v)	importance metric for node v
z(v)	weight of a subtree Tree(v), which is the number of edges in Tree(v) affected by the presence of a filter at v
$\underline{z}(x)$	sum of lengths of parts of the paths from x to all leaves in Tree(x)
P(i,j)	probability that a user i chooses an item j
q	skew parameter