Hierarchical Reactive Monitoring of Multicast Membership Size

David Breitgand, Danny Dolev, Danny Raz

***Abstract*— Knowing the number of active receivers in a multicast group is very important and useful for many applications. Retaining an exact count of the multicast group size is a very difficult task. Fortunately, estimating the group size rather than finding it exactly may be sufficient for applications like pricing. In particular, it may suffice to know at any given moment that the number of receivers in the group is between some predefined bounds, *thresholds*. This estimation is disseminated to the multicast ISPs worldwide. The ISPs then locally compute the price of transmitting a unit of bandwidth to the group, and present this price to senders in their domain.**

This paper concentrates on the algorithmic aspects of retaining multicast group size estimation for the purposes of accounting and pricing. We suggest novel network-level algorithms that combine a hierarchical control structure with a reactive (event-driven) monitoring mechanism in order to conserve network traffic and cope with the problem of feedback implosion.

We provide a rigorous definition of the problem, theoretically analyze our algorithms, and investigate their relative performance using extensive simulation studies. We show that on real Internet traces that our algorithms reduce the overhead monitoring traffic by a factor of 2-10 depending on the required accuracy and a specific behavior of the group size.

***Index Terms*—network management, IP multicast, monitoring, group size estimation, pricing, accounting.**

I. Introduction

USUALLY IP multicast tree construction is guided by receivers. When the first receiver from a domain joins a specific group, the network provider should bear the control overhead of joining that group's tree and maintaining the new tree branch for as long as there exists at least one interested receiver in its domain.

D. Breitgand is with the School of Engineering and Computer Science, Hebrew University, Jerusalem, Israel. E-mail: davb@cs.huji.ac.il. This research was supported in part by a special grant from Check Point Ltd.

D. Dolev is with the School of Engineering and Computer Science, Hebrew University, Jerusalem, Israel. E-mail: dolev@cs.huji.ac.il.

D. Raz is with the Computer Science Department, Technion, Haifa, Israel. E-mail: danny@cs.technion.ac.il.

In general, network providers may be interested in recovering the overhead costs of enabling multicast in their domains while at the same time encouraging both receivers and transmitters to use the most cost-efficient network service type (*i.e.*, unicast or multicast) for carrying their traffic. On the other hand, the transmitters need to assess the benefits of transmitting into a specific multicast group, *i.e.*, the size of the audience that they can reach.

Recently, a number of proposals [1], [2], [3] suggested to address these issues through membership size-dependent pricing of multicast transmissions. In general, the price of a transmission grows, as the group size increases.

However, it is extremely difficult to find out an exact number of individual receivers in a multicast group at any given moment. The following points summarize some of the more important problems in this context.

- The membership of a group is changing dynamically as receivers join and leave the group. Significant changes of the group size often occur over very short periods of time.
- The membership information is not confined to any particular location, but is distributed over a wide area network.
- Standards do not require the multicast routers to maintain an exact count of the local receivers.
- Centralized solutions for gathering the membership information are infeasible. In particular, if all membership notifications are being sent directly to a single management station, we get the problem known as *feedback implosion*.
- The communication overhead caused by distributed monitoring may be substantial.

For applications like pricing it is sufficient to know at any given moment that the group size is between the two predefined threshold values. The main reason for this is that senders are interested in stable pricing functions that are insensitive to minor fluctuations in the membership size. Accordingly, ISPs are not interested in tracking membership size changes when the cost of accounting exceeds the expected benefit from it.

The price of a multicast transmission into a specific group is calculated locally by each ISP using the globally consistent estimation of this group size. Additional factors such as spatial distribution of users, time of day, *etc.* may be taken into account by the pricing scheme. However, in this paper we focus specifically on the group size estimation. We propose the following generic approach to monitoring of the multicast group size for the sake of pricing. A specific pricing scheme requires knowing the group size up to some predefined constant factor $1 \leq \gamma < \infty$.

Suppose that initially the group size is known to be S_0. As long as the group size S remains such that $\frac{S_0}{\gamma} < S < \gamma * S_0$, there is no need in any additional accounting activity. We say that γ defines *low* and *high* thresholds values. Multicast ISPs need to be notified about the new group size if and only if either low, or high threshold value is violated. This way each ISP knows the group size up to factor γ at all times. Given this estimation each ISP locally computes the price of transmitting a unit of bandwidth to the given group, and presents it to the local senders.

There exists a clear trade-off between the accuracy of the group size approximation, and communication cost paid for it. Thus different pricing schemes may choose different QoS levels of accounting for controlling their overhead, while still maintaining a precision that suits them.

An exact procedure for thresholds calculation depends on the specific demands of the pricing scheme. For instance, the threshold values may reflect the maximal allowed change from the last known size of the group as above. Alternatively, the thresholds may be computed as certain percentiles of the maximal population size.

The problem can be abstracted as follows. We are given N random variables $x_1, x_2, ..., x_N$, where each variable represents an ISP. The value of each variable represents the number of receivers in the multicast domain of the corresponding ISP. The variables are dispersed over a wide-area network and maintained locally by the ISPs. Given the target function $f(x_1, x_2, ..., x_N) = \sum_{i=1}^{N} x_i$, and a predefined set of threshold values we have to devise a monitoring algorithm that detects all *threshhold violation events (TVEs)*. When a TVE is detected, the algorithm is required to notify all ISPs about the new group size estimation in a timely manner, and at low communication and processing costs. It is required that the algorithm reports a new estimate if and only if a TVE has occurred.

When we consider the typical lifespan of a multicast group, we identify three successive periods: membership build-up, saturation, and membership termination. Membership build-up and membership termination phases are characterized by rapid membership changes. However, these two periods are short relatively to the saturation phase that constitutes most of the transmission duration.

Therefore, it is natural to consider an event-driven (*reactive*) monitoring scheme that generates more traffic in the first and last phases of the multicast transmission, and conserves bandwidth in the second stage. Thus, the total communication overhead of event-driven accounting for the multicast group size will be lower compared to a monitoring scheme based on periodic polling.

This makes our problem similar to the *generic monitoring problem* studied by Dilman and Raz [4] in a different context. The event-driven monitoring schemes presented in [4] combine local thresholds violation reporting by the monitored variables to the centralized management station, and global polling performed by the latter if the probability of a TVE is high. For a variety of reasons explained in detail in Section II, the algorithms presented in [4] will not be satisfactory for solving the problem above. Instead, they serve us a starting point for developing better solutions, and provide the yardstick for assessing it.

When we consider multicast, it is very natural to assume that we have a (bounded out-degree) tree structure where the leaves are the end routers that have a local count regarding the number of participants in the group. Since we want to know the total size of the group, our aim is to calculate the sum of the values of the leaves. This can be done in a hierarchical way in the internal leaves of the tree. In this way, every polling step consists of calculating the sum of the values up the tree and disseminating the value from the root to all the leaves if needed.

This hierarchical polling process is a natural way to overcome the feedback implosion problem, and it also saves a factor of $\log N$ messages in each polling interval, where N being the number of ISPs. The next step is to combine it with the reactive monitoring process. This leads us to constructing a hierarchical and reactive monitoring algorithm that allows to maintain the multicast membership size estimation with very low communication overhead.

It is important to notice that making the membership monitoring function to use the tree constructed by multicast routing protocols requires changes to the latter. This approach has been taken by Holbrook and Cheriton in their EXPRESS proposal [3]. In the current paper we are not proposing any changes to the existing routing protocols, but rather suggest the accounting function to be implemented as a separate service, being orthogonal to the routing functionality. Other substantial differences between EXPRESS and our proposed algorithm are summarized in Section II.

In the current paper we specifically concentrate on the algorithmic aspects of monitoring multicast group size. We do not deal with other issues concerning multicast or specific multicast pricing schemes. We define a model that captures the essential parameters of the problem, which allows us to formulate it rigorously. Using this model we first show that event driven monitoring methods are preferable even if the flat control structure is used. Then we turn out to study monitoring algorithms that use hierarchical trees. We show that in addition to addressing the feedback implosion problem, the combination of a hierarchical structure and event driven techniques results in very efficient monitoring algorithms. Next we analytically study the behavior of these algorithms under various probabilistic conditions. Then we conduct an extensive simulation study that demonstrates the amount of reduction in traffic overhead that can be achieved when using our algorithm, and validate the obtained results with the data from the real traces.

The rest of this paper is organized as follows. Section II provides an overview of the related work. In Section III we rigorously define our model. In Section IV the trade-off between the reactive and non-reactive monitoring is explored. In Section VIII the basic form of our hierarchical reactive monitoring algorithm is described. In Section IX we analyze its performance theoretically. Section X provides simulation methodology and results. Section XI extends the basic hierarchical reactive monitoring algorithm and provides more simulation results. Section XII validates our simulation and theoretical results with the real traces. We conclude in Section XIII.

II. Related Work

The need to know the number of active receivers in a multicast session often arises at the transport and application levels [5], [6], [7]. To this end a number of probabilistic estimation methods have been proposed recently [8], [9], [10], [11], [12]. Although there exists a common core in all probabilistic methods of multicast group size estimation, the details of various algorithms differ considerably. A comprehensive comparative study of the existing probabilistic estimation algorithms is beyond the scope of this paper. Nonetheless a comparison of our proposed algorithms to the existing probabilistic methods is naturally required. To this end in Section IV we define generic probabilistic monitoring algorithms that embodies the functionality common to all probabilistic group size estimation methods. We use these generic algorithms for comparison of communication costs involved. The issue of communication overhead is rarely in the focus of the probabilistic estimation proposals.

The reactive monitoring algorithms presented by Dilman and Raz [4] constitute the starting point for this research. A number of problems preclude a straightforward re-use of these algorithms in the context of multicast group size estimation.

- Centralized global polling of the variables poses the feedback implosion problem as the size of the network grows.
- The cost of communication with the centralized manager does not depend on the actual distance between the monitored node and the manager, and assumed fixed. This may be appropriate for intra-LAN monitoring, but does not fit well into a WAN environment.
- The efficiency of the algorithms depends on the statistical nature of the monitored data. Thus, one should adjust the algorithms to the statistical nature of the multicast group sizes monitoring.
- In case of shared trees, the group size estimation should be propagated to all monitored variables (*i.e.*, all ISPs involved), and not only to a dedicated manager.

Recently, Holbrook and Cheriton proposed a new IP multicast routing protocol, EXPRESS [3], that incorporates the membership size accounting function at the granularity of individual hosts as part of its operation. The native support for this accounting constitutes the primary motivation for EXPRESS. The price of transmission is assumed to be dependent of the size of the group as accounted by the routing and management protocol called ECMP.

The event-driven version of ECMP is close in spirit to our hierarchical reactive monitoring algorithm of Section VIII. There are important differences, though. In ECMP a dedicated multicast tree is built for each active source. If the source wants to find out how many active receivers exist in its reach, it multicast counting request with parameters τ and α. Each multicast router in the tree computes the *error tolerance curve* $e_{max} = \frac{-log(dt/tau)}{\alpha}$, where dt is the time interval since the last report to the father router in the tree, τ is the maximal time that the router is allowed not to report, and α controls the rate of decay of e_{max}. In addition, at each node the current error is computed as $e = max(\frac{cur}{adv}, \frac{adv}{cur})$, where cur being the current number of receivers in this router's subtree known to this router, and adv being the last value reported to the father in the tree.

As long as $e < e_{max}$ holds, the router does not report anything. When the inequality no longer holds, the router should report its current notion of the number of receivers to its father. These values are being summed up along the tree, and eventually the sum reaches the source. Thus the source obtains the estimation of the group size with the precision being defined by e_{max} curve.

One important observation about this algorithm is that it generates accounting traffic even when there are no actual changes in the membership size. This happens because e_{max} decays with passage of time, and eventually reaches 0 after maximum τ time units.

In [3] an initial simulation study of ECMP have been presented. However, no compelling comparative study, or analysis of the performance have been shown.

Since our primary target application is pricing, a short discussion on multicast pricing proposals is called for. Chuang and Sirbu [1] were the first to express the cost-effectiveness of multicast as a simple power-law function of the active membership size. Namely,

$$N^{k-1} = \frac{L_m}{L_u N}. \tag{1}$$

L_m is the total number of links in a multicast tree, L_u is the average length of the unicast route, and N is the number of leaf routers. The exponent k is called *Economy of Scale* (EoS) factor, $0 \leq k \leq 1$. In [1] it was demonstrated through simulations that $k \approx 0.8$ for realistic networks.

This result was later re-validated mathematically by Phillips *et al.* [13]. Recently Chalmers and Almeroth [14] confirmed it using the data collected from MBone, though they suggested that the EoS factor is closer to 0.7.

Equation (1) suggests a straightforward multicast pricing scheme. It is assumed in [1] that the multicast routers can be discovered using distributed monitoring. However no specific accounting mechanism has been suggested. Additional important discussion on accounting and pricing of the multicast traffic can be found in [2], [15], and [16].

Pioneering work investigated the join/leave behavior of the users has been done by Almeroth and Ammar [17]. They established that there are two types of multicast sessions, *long* and *short*. For the periods of maximal activity, the inter-arrival and stay time for receivers participating in the short sessions is well modeled by exponential processes. For the long sessions the inter-arrival time fits well into an exponential distribution, while the stay time is better modeled by Zipf distribution. Similar behavior has been discovered by Sarac and Almeroth a few years later, in 1999 [18]. We use the findings of [17], [18], and [1] in our simulation studies.

III. Model and Problem Definition

Let $G(V, E)$ be some synchronous network. Let $V' \subseteq V$ be a collection of nodes representing the ISPs participating in a multicast group tree. We assume that there exists a reliable unicast channel between each pair of nodes $v_i, v_j \in V'$ in addition to the spanning tree of the multicast group connecting all the nodes in V'.

Each $v \in V'$, maintains variable $x_i \in I^+ \bigcup \{0\}$, $x_i < \infty$. By $x_i(t)$ we denote the value of x_i at time $t \in I^+ \bigcup \{0\}$. The physical meaning of this variable is the number of active receivers in the domain represented by node v.

The system proceeds in rounds where all messages sent at the beginning of a round are received by their destinations by the end of the round. Each round is completed within one time unit. In practice, the actual value of the time unit may be important. This is the maximal round trip time between any two neighbors in the network.

In addition, each node in V' can send or receive no more than a fixed number of messages. All messages that exceed this threshold get lost. All messages are assumed to be of a fixed limited length.

Let $f(t) = f(x_1(t), x_2(t), ..., x_N(t))$ be a *target function* depending only on the values of the variables at time t. At any given time every node in V' is interested in knowing whether the value of the target function exceeds a given value, or falls below another given value. These values are termed *upper*, and *lower thresholds* respectively. In this paper we are interested in a specific target function, $f(t) = \sum_{i=1}^{N=|V'|} x_i(t)$.

Let $\omega(n)$ be some function assuming non-negative integer values, and bounded from above.

Definition 3.1 $f(t) = f(x_1(t), x_2(t), ..., x_N(t))$ is ω-constant function if and only if it is being constant on the semi-open intervals of length ω: $[0, \omega), [\omega, 2\omega),\ etc.$

In fact, it is the same function after discretization on time intervals of length ω. Every node possesses only the value of its local variable. Thus, it takes some time for a node to assess the value of the target function, since it needs to communicate at least with some of the other nodes. In this paper we assume that the membership size is $O(\log^2 N)$-constant. In other words, we do not account for the fluctuations in the target function value if these changes last for less than $O(\log^2 N)$ time units. Under the assumptions made above, we are interested in assessing the value of $f(t) = \sum_{i=1}^{N=|V'|} x_i(t)$ being an ω-constant function.

Definition 3.2 If at time t, $f(t) \leq H$, and at time t_a, $f(t_a) > H$, where t_a being minimal time such that $t_a > t$, we say that a *global upper threshold event* (or a simply *cross-over* event) has occurred at time t_a.

We define a *global lower threshold event*, or *cross-down* similarly. Now we are ready to formulate our monitoring problem. Figure 1 provides the definition.

Given synchronous reliable network $G(V, E)$, a collection of independent random variables $x_1(t), x_2(t), ..., x_N(t) \in I^+ \bigcup\{0\}$, each being maintained by some $v \in V' \subseteq V$, a target function $f(t) = \sum_{i=1}^{n=|V'|} x_i(t)$, being ω-constant function, a set of threshold values $X_1, X_2, ..., X_m$, find an algorithm that guarantees that if and only if at time t_a a global threshold violation event occurs, all nodes in V' are notified about this event by time $t_a + \omega$.

Fig. 1. Problem Definition

Monitoring algorithms differ on the set of variables they measure, the actual timing of measurement, and the means of diffusing the information. We propose to compare different monitoring algorithms in terms of the total communication cost incurred by them over the time of their operation. Let $C(A, \delta)$ denote the total number of messages sent by algorithm A over time δ. *It is important to stress that we count traversing of each hop between any two neighbors in the network as a separate message*.

IV. Reactive Monitoring versus Polling

In this section we analyze the benefits provided by a reactive monitoring scheme versus a polling-based one in which the values of the monitored variables are being solicited at deterministic time intervals irrespective of the actual conditions in the network.

To separate this issue from all others, we consider a flat structure for information dissemination. More specifically, we consider the often used *many-to-one, one-to-many* structure. In this scheme the nodes use unicast channels to communicate their values to the relay, and the relay uses the multicast tree to disseminate the sum of the values to all ISPs at the edges[1].

In general, in this work we are not concerned with the actual topology of the network, or with the exact topology of the multicast tree constructed by the underlying multicast protocol. To simplify matters, we assume that the height of a spanning tree constructed by the underlying multicast routing protocol is $O(\log N)$. Moreover, we assume that the multicast delivery tree is a full binary tree. In our calculations we defined the cost of a single multicast to be the number of hops in the tree, $c_m \stackrel{\text{def}}{=} 2(N-1)$ (because there is a total of $2(N-1)$ hops in the full binary tree with N leaves). Consequently, we define the cost of unicast, $c_u \stackrel{\text{def}}{=} \log N$.

We define a straightforward monitoring algorithm OB_c as shown in Figure 2.

[1]The alternative, *many-to-many* communication pattern is not being considered as impractical in the WAN setting.

Each predefined time interval every node sends its value to the dedicated relay-node. The relay computes the sum, and compares it against the global thresholds. If it detects a TVE, it multicasts the new size of the group. Otherwise, it sends no messages. If a node does not receive any message from the relay in response before a predefined timeout, it assumes that the size estimation of the group remains unchanged.

Fig. 2. Straightforward Monitoring Algorithm OB_c.

$C(OB_c, \delta) \stackrel{\text{def}}{=} O(\delta * N * c_u + e_\delta * c_m)$, where e_δ being the number of threshold events detected by the relay node during time period δ, and N being the number of the monitored variables.

In [4] four different reactive monitoring algorithms using similar flat control structure were presented, albeit in a different context. The key idea of these algorithms is that instead of inspecting the local values each predefined time interval, the monitored nodes are assigned local thresholds. When a local threshold is violated, this triggers a report toward the dedicated manager that decides whether to poll other nodes.

With minor modifications this idea can be applied to our problem. Let $f(0)$ be the sum of the local values, *i.e.*, the size of a group at time 0. Let L and H be the threshold values *s.t.*, $L \leq f(0) \leq H$. Then every node representing a leaf router is assigned local thresholds $l = \frac{L}{N}$, and $h = \frac{H}{N}$.

- **Simple Value (SV):** every predefined time interval, each node updates its local value. If it detects a local TVE, it sends a trap to the dedicated relay. If at least one trap is received, the relay multicasts an unconditional polling request, collects the reported values, and calculates the sum. If either H, or L, is violated, the relay recalculates the local thresholds, and multicasts them along with the new sum. Otherwise no messages are sent.
- **Simple Rate (SR)**: instead of checking the absolute value, the node inspects the local rate of change. The relay makes an assumption that the change rate is bounded by ρ. The value of ρ is multicast to the nodes. Under this assumption the relay calculates the number of time intervals during which polling can be safely omitted before the global sum has a chance to reach either H, or L. If the relay does not hear from the nodes until this time, it multicasts a polling request to validate the current thresholds. If a node detects that the assumption is locally violated, it sends a trap to the relay. If the latter receives at least one trap, it polls all the nodes and acts upon the received values similarly to SV above.
- **Improved Rate (IR):** Similar to SR, but instead of

checking for the rate change at each step, the local node calculates the average change since the last report it sent. If this average is below the rate threshold, no trap is sent to the relay. This improvement increases the chances to save the cost of communication from the nodes to the relay, which may be very significant.

- **Improved Value (IV):** Similar to SV, but instead of acting upon a single trap, the relay lowers the local thresholds for the nodes a little bit, *i.e.*, instead of l, h, the nodes receive local thresholds $l+\lambda$, and $h-\lambda$. A single trap does not trigger the global polling. Instead, the relay calculates the minimal and maximal possible values of the target function assuming the minimal and maximal values for the nodes that have not reported. If there is a chance for a global TVE, the relay polls the nodes as in SV. The basic idea in this improvement is that although the possibility of an individual trap slightly increases, the possibility of the global poll decreases which, in turn, decreases the total cost of the algorithm.

For all the algorithms above the cost calculation is, essentially, the same. We refer to them as *flat* algorithms, since the control structure they use is, essentially, flat. Let r_δ be the total number of local traps sent by all nodes to the relay over time interval δ. Let e_δ be the number of global events detected by the relay, and k_δ be the number of global polls initiated by the relay. Then $C(A) = O(r_\delta c_u + c_m(k_\delta + e_\delta) + c_u k_\delta)$.

In the following section we explain the simulations methodology that is common to all simulation studies performed in this paper. Section VII presents the comparative simulation study of the flat reactive algorithms versus the polling-based algorithms for different methods of threshold calculation.

V. Simulation Methodology

In this section we explain the synthetic dataset creation methodology, and estimation methodologies that we investigated.

A. Synthetic Dataset

In this work we focus on the short-lived multicast sessions. The reason for this is that shorter sessions are less stable, and, therefore are more challenging. Following [17], [14] we model the inter-arrival and stay times of the multicast receivers as exponentially distributed variates with means μ_{arr}, μ_{stay}. The inter-arrival time is the time between consecutive join operations of the same user. The stay time is the time between a join and the next leave operation of the same user.

Throughout the simulations we assume that users are uniformly distributed over ISP domains. Each user is assigned a fixed domain number throughout a single trace. We assume a fixed ratio of users to domains, namely, 30. Thus, for each fixed number of domains, we know the total number of receivers. For instance, for 256 domains we have the total of $256 * 30 = 7680$ potential receivers.

Each synthetic multicast trace that we create is comprised out of 10 fixed length *sessions*. Each session models a multicast transmission of a specific popularity. As explained earlier, a session lifespan consists of three consecutive stages: membership build-up, membership saturation, membership termination. The popularity of a session refers to the average number of receivers in the membership saturation stage.

To model sessions of different popularity we compute 15% regions of the receivers space. In our example, for 7680 users the threshold values computed this way will be $0, 1152, 2304, ..., 7680$. Each pair of neighboring thresholds defines a different popularity index (rating) of the group.

We draw a group rating for each session out of the uniform distribution. Then we randomly choose a point within the interval defined by the rating and set the expected average number of receivers for this session to be that point. For instance, if we draw 0 as the group rating for a given session, the average number of users for this session will be set as a random number between 0 and 1152 in our example.

Let U_{total} denote the total number of potential receivers. We fix μ_{arr} throughout all sessions, to be $0.1 * d$, where d is the session duration. Then, we find μ_{stay} for a given session in the multicast trace using the following formula

$$U = \frac{\mu_{stay}}{\mu_{arr} + \mu_{stay}} U_{total}. \tag{2}$$

This methodology removes a statistical dependency on the specific parameters of the model. Figure 3 provides a sample trace of the group membership for 512 multicast domains, 15360 potential receivers acting according to the exponential model above, and being uniformly distributed among the domains. There are 10 different multicast sessions per trace. All sessions are of equal length, 1800 time units.

Each session starts to build its membership from 0, and at the end of transmission all users leave the group simultaneously. There are no inter-session idle periods, and the sessions themselves are relatively short. This is, of course, an extremal scenario. In reality the membership not necessarily drops to 0 at the end of each session, and exist larger periods of stability. Our model serves us as a basic framework. The basic scenario can be changed by adding stability periods to the trace that freeze the membership activity. We express these stability periods as a certain percentage of the total trace duration. As suggested in [18] in today's MBone it is common to observe bursts of activity around some well publicized events interleaved with the long inactivity periods.

The cost of a single unicast in this trace, $c_u = 9$ messages, the cost of a single multicast transmission, $c_m = 1022$ messages.

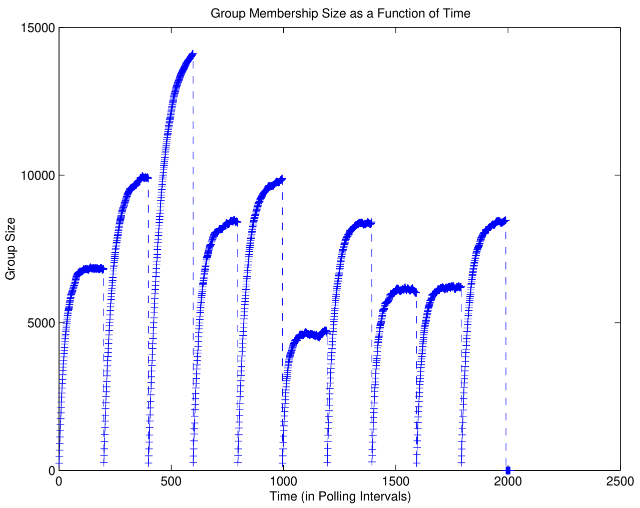

Fig. 3. Sample Membership Size Dynamics

VI. Estimation Methodology

The requirements on estimations come from the application. The following general methodologies are conceivable:

- **Relative to Total (R2T)** : the estimation purpose is to show what percentage of the total maximum population is active. The total space is partitioned into percentiles. These percentiles also define precision of the estimation.
- **Relative to Last (R2L)**: the estimation purpose is to show the change of the group size relatively to the last known size. Precision of the estimation, γ, should be specified separately. At every given moment the estimation provides the group size up to a multiplicative factor γ.

In general, R2T is more applicable to the large groups while R2L is applicable to small and large groups alike. Since R2T is less precise than R2L, it incurs less communication overhead though.

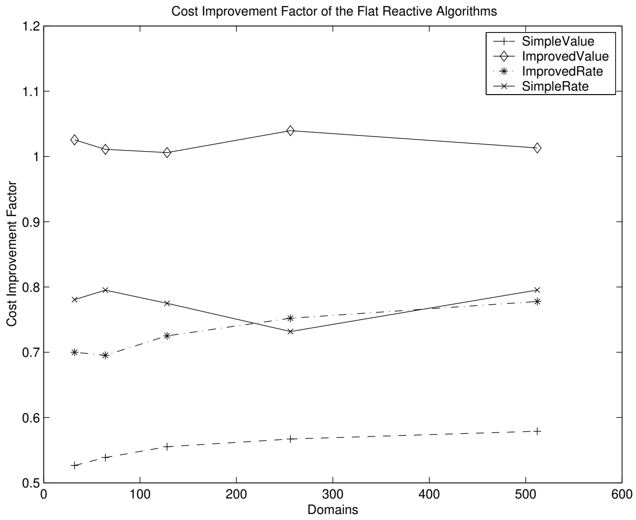

Fig. 4. Cost Improvement Factor (Maximal Load)

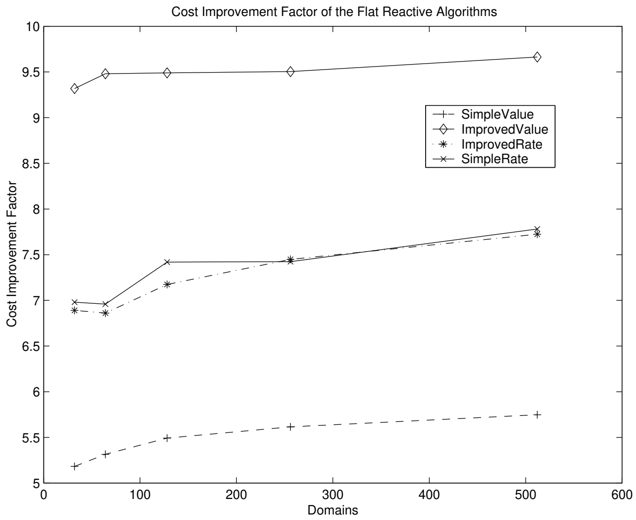

Fig. 5. Cost Improvement Factor (90% Inactivity Time)

VII. Performance of the Flat Reactive Algorithms

Figure 4 presents the average ratio (cost gain factor) between the total message cost of OB_c (the straightforward monitoring algorithm based on the dedicated relay) and the flat reactive monitoring algorithms. The cost gain factor, $C(OB_c)/C(A)$ is shown as a function of the number of simulated domains. The estimation methodology used for obtaining these results was R2T with 15% precision. Notice that due to the synthetic dataset creation methodology, the performance of R2T and and R2L for the same numerical values of precision (in this case 15%) is very close. R2L estimation methodology is explored in greater detail in Section XII where the data comes from the real traces while the total population is unknown.

To eliminate the statistical dependency, 200 different traces have been generated per each set of domains. Each point in the graph represents the averaging of these runs. For all algorithms the standard deviation of the resulting ratios ranges between 0.012, and 0.02. Therefore, the averaging indeed makes sense.

First thing to notice about the presented graph is that the results are rather disappointing. Simple Value and Simple Rate algorithms are about twice more expensive than OB_c. Improved Rate is slightly better, and Improved Value (the most efficient algorithm being presented in [4]) is only marginally better than OB_c.

For Simple Rate and Improved Rate algorithms, the initial choice of the upper bound on the rate of changes, ρ, considerably influences the performance. Similarly to [4] we performed the number of simulations in which this parameter has been varied in order to determine the value rendering the best performance. In Figure 4, only the best performance of the algorithms is shown. The performance maximum has been attained for Simple Rate and Improved rate algorithms for $\rho = 0.0015 * total\_number\_of\_users$. The preliminary simulations used to determine this parameter for attaining the maximal performance are not shown here. Under the maximal load conditions these two algorithms almost do not differ. Similarly, for the Improved Value algorithm the best performance was attained with $\lambda = 0.02 * total\_number\_of\_users$.

So, why these reactive algorithms that have been demonstrated to be so efficient in [4] perform so badly for multicast group size monitoring? The primary reason for this is the different statistical nature of the workload.

Throughout all the traces the probability that at least one variable violates its local threshold has been measured. We found that it is 0.99 on the average. Moreover, on the average $\approx 1/2$ of the variables send their traps to the relay in *each* round. For Simple Value and Simple Rate algorithms even a single trap message triggers the global poll in almost all rounds. However, when we measure the probability of detecting a global threshold event at the relay, we find that it is only 0.02 on the average!

This means that although there may be a lot of continuous activity in the leaf routers with many users joining and leaving the group, these changes do not affect the total size of the membership considerably. Therefore they are not worthy of accounting.

However, in the Simple Value, and Simple Rate algorithms the global polling operation is triggered by even a single trap received by the relay. The global polling means multicasting of the polling request and collecting the unicast replies from the nodes. Since about half of all nodes have already sent their traps in the previous round, this operation makes the algorithm at least twice as expensive as the straightforward convergecast and multicast performed by OB_c. Improved Value algorithm performs better because a single trap does not trigger the global polling operation, although the probability of the local threshold violation is slightly higher. The increase in this probability does not have any significance in this scenario though, because the probability of local trap was almost 1 anyway.

The lesson that we learn from this extreme scenario is that all flat reactive algorithms suffer from the same intrinsic problem. Namely, *the threshold events of no significance are communicated directly to the relay*.

So, does the reactive scheme provide any benefits at all when the flat control structure is used? When we consider the more realistic scenarios when the traces include periods of "no activity" that correspond to either a very popular event *e.g.*, the transmission of the final game of the World Cup, or to the temporary absence of events, we obtain the cost improvement that is considerably greater than 1 even in the flat model.

Figure 5 shows the cost ratio of the same algorithms measured on the traces with 90% stability time being attributed to the reasons above. As one can see (note the change in scale on axis y) the cost ratio is substantially better showing that even the simplest reactive monitoring algorithm, Simple Value, renders the improvement factor of over 5. The Improved Value algorithm performs almost an order of magnitude better than the straightforward monitoring algorithm.

Observing this behavior of the flat reactive monitoring algorithms leads us to developing a different monitoring algorithm that combines the benefits of reactive monitoring with the hierarchical information dissemination scheme that suppresses the local threshold events from being globally propagated. By the same token the hierarchy solves the problem of feedback implosion that have been ignored so far for the sake of discussion. While this problem, indeed, can be ignored for the current MBone whose actual size roughly corresponds to our simulations, the feedback implosion problem will become acute when the multicast will be deployed on a large scale.

VIII. Hierarchical Reactive Monitoring Algorithm

This section presents Hierarchical Reactive Monitoring Algorithm (HRMA) that overcome the problems of feedback implosion and poor performance under the maximal load conditions that are common to all flat algorithms.

The basic idea is as follows. Instead of using the direct unicast communication channels for reporting local values of ISP variables, and the multicast communication channel for propagating the polling requests, we organize the nodes into a logical hierarchy in which the same node may act at different levels. The inter-nodal links are realized as reliable unicast FIFO channels.

The optimal hierarchy formation and maintenance needs a special study since one can look for several optimization functions. In this paper, for the sake of simplicity, we assume that the logical hierarchy is a full binary tree.

The leaf nodes (ISPs) report their traps up the hierarchy, and the intermediate nodes act as local relays deciding whether to propagate the trap further toward the root depending on the sum of the values reported by their children. This scheme leads to a logarithmic gain in communication toward the root (that acts as the global relay), and, also avoids the feedback implosion problem. In fact, the communication overhead is even lower because not all links in the hierarchy require actual network transmissions.

Figures 11, 12, and 13 present the pseudo-code of the local algorithms run by the leaf nodes, intermediate nodes, and the root of the hierarchy respectively. Each node in the hierarchy has a notion of its father, and children in the tree. Each node has values l and h that hold the current low and high thresholds for the subtree rooted at this node. For simplisity we assume that the root also has an array of threshold values, $thresh$, as determined by the rating service.

Initially, a convergecast is performed to compute the total number of active receivers at time 0. Depending on this value, the local thresholds for each node are calculated, and multicast down the tree (Figure 13). Initially, the intermediate nodes do not have l and h. They compute these values when they receive traps from their children.

When a node sends a trap to its father in the tree, we say that it starts verifying the threshold condition. This process ends either at an intermediate node, or at the root of the whole tree. A node stops the process if it discovers that neither high, nor low threshold of the tree being rooted at it is violated. In this case, the node does not send any message. The leaf nodes in its subtree will discover that no global event has occurred after a timeout.

If the root discovers that a global threshold event has happened, it sends an *update* message down the hierarchy informing the ISPs about the new group size estimation.

Lemma 1: Verification of any local threshold event is completed after $O(\log^2 N)$ time units, where N being the number of leaves (ISPs).

Proof: The worst case happens when a global threshold event occurs, but only a single leaf node sends a trap to its father because at all other nodes no local thresholds were violated. In this case the father requests a report from its other child, and when this report is received, the father discovers that either lower or upper threshold is violated. This forces the father to send report to its own father *etc.*, until the report reaches the root. The process of checking the total value of a subtree is repeated at most $\log N$ times. Hence, the whole algorithm takes $2\log^2 N$ time units. This completes the proof. ■

Lemma 2: Message cost of *HRMA* per time window of duration $O(\log^2 N)$ is $O(N)$.

Proof: It is sufficient to observe that REPORT-REQUEST message is sent at most once, REPORT is sent at most once, and UPDATE is sent at most once on each edge. Since there are $2(N-1)$ edges in the full binary tree with N leaves, the lemma follows. ■

Theorem 1: Let $\sum_{i=1}^{N} x_i$ be $O(log^2 N)$-constant function. Then *HRMA* is a correct monitoring algorithm. (See Section 1).

Proof: Sufficiency is straightforward since the all ISP nodes get notified about the TVEs only by the root, and the root only informs about the TVEs that indeed occur. To prove the necessary condition, observe that if there is a global TVE, then there is at least one local threshold violation event. Then necessity follows by Lemma 1. ■

IX. HRMA Analysis

In this section we analyze the expected cost gain factor of *HRMA* per round. To simplify the problem we study a restricted case where each monitored variable may assume a binary value. In other words, this is the same monitoring problem when the group size estimation is done at the granularity of ISPs.

For the general case the performance of *HRMA* is studied through simulations. The simulation results are summarized in the next section.

A. Yardsticks

We first need to establish a baseline algorithm against which the cost gain ratio of *HRMA* will be computed. It would be unfair to compare *HRMA*'s performance to that of OB_c since they use different information dissemination structures. Instead, we define a *Hierarchical Straightforward Monitoring Algorithm*, denoted OB_h that uses the hierarchical structure for communication as explained in Figure 6.

Obviously, $C(OB_h, \delta) = 2(N-1)(\frac{\delta}{2\log^2 N} + e_\delta)$, where e_δ being the number of TVEs detected by the root. It is easy to observe that $C(HRMA, \delta)$ may be higher than $C(OB_h, \delta)$. In the worst case the performance of *HRMA* is about twice as bad as that of OB_h. However, the worst

The leaf nodes initiate a convergecast of their local values toward the root of the logical hierarchy every $2\log^2 N$ time units. The root checks whether a global TVE occurs, and if so, it multicasts the new group size estimation using the same hierarchy.

Fig. 6. Hierarchical Straightforward Monitoring Algorithm (*OB*$_h$).

case is not necessarily the one that occurs most often. As our simulations show, *HRMA* is outperforming *OB*$_h$, and also the flat reactive algorithms even when the load is very high, *i.e.*, when the membership is very dynamic in the leaves, while being relatively stable in total.

But does *OB*$_h$ really provide a sensible baseline for assessing communication cost gain of *HRMA*? What about more advanced alternatives?

One such alternative is offered by probabilistic polling techniques. A generic probabilistic estimation procedure communicates directly with the receivers, and not with their ISPs. It proceeds in rounds. In each round a probability of response is multicast by the dedicated node, usually the source. The receivers respond with the advertised probability. Based on the number of responses, an estimation of the total number of receivers is performed. Although there are many more details involved, this *Naive Probabilistic (NaP)* algorithm captures the essence of the methodology. *NaP* avoids the problem of feedback implosion. However, it is easy to see that the communication cost of *NaP* approximately equals that of *OB*$_h$. Therefore, we do not use probabilistic methods for comparison. The comparison of *HRMA* versus *OB*$_h$ applies also to *NaP*.

Another alternative is ECMP of EXPRESS. Nonetheless, it is again easy to observe that the communication cost incurred by ECMP is *at least* that of *OB*$_h$ when the maximal silence period is set to be $O(2\log^2 N)$. Therefore comparing *HRMA* performance to that of *OB*$_h$ is sufficient to draw the conclusions about ECMP versus *HRMA* performance.

B. Analysis

For the sake of simplicity we assume that each leaf node assumes value 1 with an independent probability p, and value 0 with probability $q = 1 - p$.

To further simplify the analysis we assume that all leaf nodes have equal local thresholds, and that the target function is non-decreasing (*i.e.*, the membership only builds up).

Let n_i denote the number of leaves in a full binary subtree of height i. Let n_i^1 denote number of leaves in a binary subtree of height i for which the values of local variables equal 1. Let $0 < \gamma < 1$ be a given value. Then each node in the hierarchy is allocated a local threshold value $\gamma_i \stackrel{\text{def}}{=} \gamma * n_i$. A threshold value γ_i means that the root of the full binary subtree of height i should either issue a REPORT message to its father, or declare a global TVE if it is the root of the whole tree, if and only if $n_i^1 \geq \gamma_i$.

Let $p_i(\gamma_i)$ be the probability that a node at level i in the hierarchy sends a REPORT message to its father at level $i+1$. Under our assumptions

$$p_i(\gamma_i) = 1 - \sum_{k=0}^{\gamma_i * n_i} \binom{n_i}{k} p^k (1-p)^{n_i - k}. \tag{3}$$

For the sake of brevity we will omit γ_i from our notation, and use simply p_i to denote $p_i(\gamma_i)$, and $q_i = 1 - p_i(\gamma_i)$. From observing the *HRMA* operation it is easy to compute an expected cost for an edge at level i (*i.e.*, the expected number of messages per step of the algorithm being sent on this edge). Obviously, this cost, denoted $C(e_i)$, lies between 0 and 2 since at most two messages are sent by *HRMA* over any edge per step. There are two possibilities. Either a REPORT message is sent over e_i by the node being at level $i-1$ (in this case $C(e_i) = 1$), or a REPORT message has been solicited by a node at level $i+1$ (in this case $C(e_i) = 2$). Consequently, $E[C(e_i)]$ is computed as follows:

$$E[C(e_i)] = p_i + 2[q_i * [p_i + q_i * [p_{i+1} + q_i * [p_{i+2} +$$

$$q_i * [p_{i+3} + \ldots * [p_{\log N - 1}]\ldots].$$

Or in a more concise form:

$$E[C(e_i)] = p_i + 2 \sum_{j=i}^{\log N - 1} q_i^{j-i+1} p_j. \tag{4}$$

Since there are $N/2^i$ edges at level $i \in [0, \log N - 1]$, the total expected cost of *HRMA* per step is

$$E[C(\mathit{HRMA})] = \sum_{i=0}^{\log N - 1} \frac{N}{2^i} E[C(e_i)] + 2(N-1) * p_{\log N}. \tag{5}$$

The total expected cost of *OB*$_h$ per step is

$$E[C(\mathit{OB}_h)] = 2(N-1) * (1 + p_{\log N}). \tag{6}$$

The second term in Equation (5), and Equation (6) accounts for the multicast of a global TVE notification. Since this cost is the same for both algorithms, it can be omitted from the calculation. Consequently, the expected per step cost gain factor of *HRMA* over *OB*$_h$ (and therefore also over ECMP, and *NaP*) is given by the following equation.

$$E[C(\mathit{OB}_h)/C(\mathit{HRMA})] = \frac{\sum_{i=0}^{\log N - 1} \frac{N}{2^i} E[C(e_i)]}{2(N-1)}. \tag{7}$$

If we let $p = 0.5$ and $\gamma = 0.5$, then $p_i = 0.5$ for all $i \in [1, \log N]$ regardless of N, and the computation becomes easy. According to Equation 4:

$$E[C(e_i)] = \frac{1}{2} + 2 \sum_{j=1}^{\log N-i-1} \frac{1}{2}^j * \frac{1}{2} =$$

$$\frac{1}{2} + \sum_{j=1}^{\log N-i-1} \frac{1}{2}^j = \frac{1}{2} + \sum_{j=0}^{\infty} \frac{1}{2}^j - 1 - \sum_{j=\log n-i}^{\infty} \frac{1}{2}^j.$$

By neglecting the last term we obtain that $E[C(e_i)] \approx 1.5$. Thus, $E[C(OB_h)/C(HRMA)] \approx \frac{2(N-1)}{3(N-1)} = 0.6667$.

In the general case, it is difficult to compute Equation (7) analytically. Therefore we present the numerical results in Figure 7. This figure shows the behavior of the cost gain factor as a function of γ and p for $N = 512$. When p approaches 1, the cost gain approaches 1 irrespective of the threshold allocation. When p approaches 0, the cost gain factor goes to ∞, as *HRMA* pays 0 in communication ($p = 0$ is not shown in order not to distort the visibility of the graph).

Figure 8 shows the isoline plot for $E[C((OB_h)/C(HRMA)]$ when $N = 512$. One can see that as threshold grows and the probability of having at least one receiver in an ISP domain decreases, the cost gain factor of *HRMA* increases. However, as probability of having at least one receiver increases, *HRMA* becomes more costly than OB_h. Given an apriory knowledge of the probabilities (e.g., through long term measurements that identify typical user behavior patterns, or through on-line probability estimation procedure), an administrator can decide which monitoring algorithm cheaper to deploy to meet the given precision requirements.

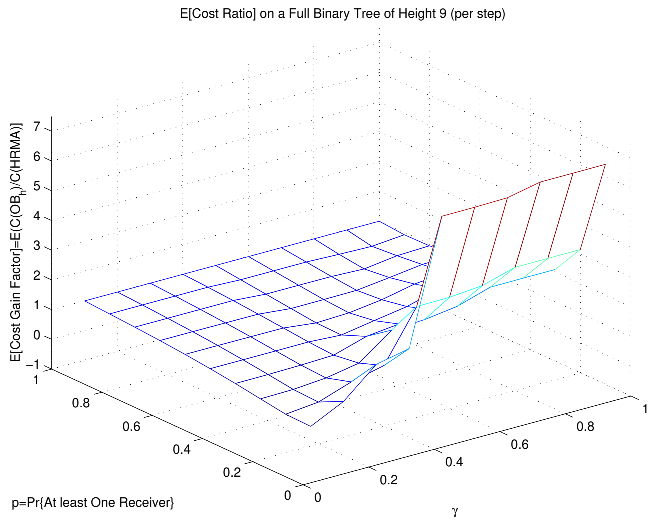

Fig. 7. Expected Cost Gain Factor of *HRMA* for N=512.

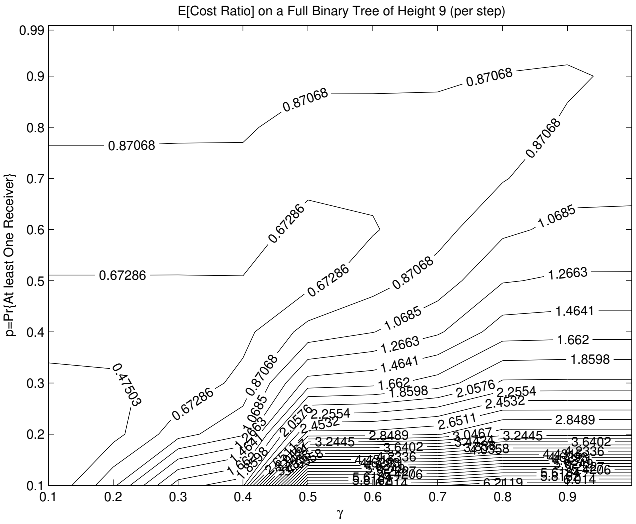

Fig. 8. Isoline Plot for Figure 7

X. Performance of *HRMA* (Simulation Study)

In order to evaluate performance of *HRMA* in the general case we use the same synthetic dataset as in Section V.

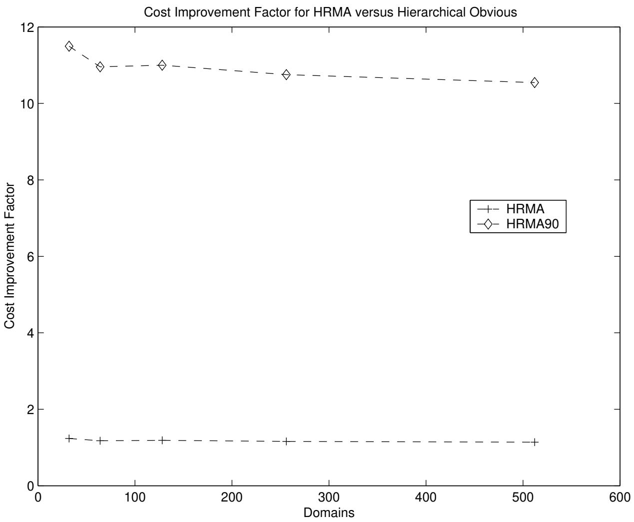

Fig. 9. *HRMA* vs. OB_h

Figure 9 shows performance of *HRMA* versus OB_h under the maximal load conditions (*HRMA* curve), and for 90% inactivity time trace (*HRMA*90 curve). As one can see, the cost gain factor behaves similarly to the flat model. Under the maximal load conditions (high probability of violating a local threshold), *HRMA* produces a small message cost gain ranging from 14% to 20%. To understand why this gain is relatively small recall from Section VII that per step, almost $0.5 * N$ leaf nodes trap to their fathers in the hierarchy. Since users are distributed uniformly, we can expect $0.25 * N$ nodes in the second level to send the *report-request* messages to one of their

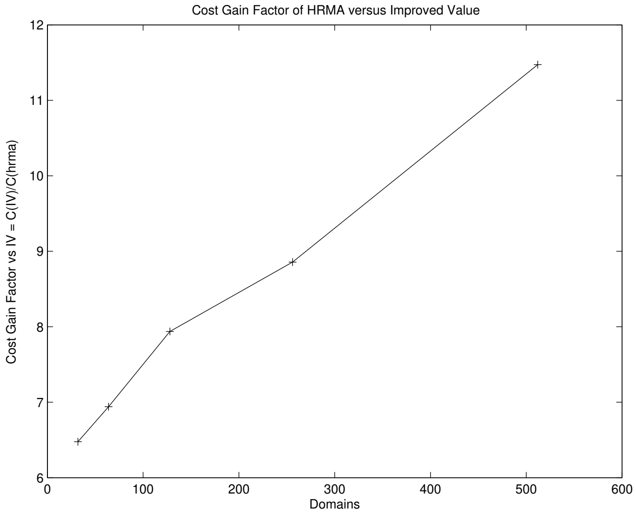

Fig. 10. *HRMA* vs. Improved Value

if (UPDATE from $father$ containing values α_1, α_2)
    $l \leftarrow x_i * \alpha_1$; $h \leftarrow x_i * \alpha_2$;
every $2\log^2 N$ rounds update the local value, x_i;
    if ($x_i \leq l$ or $x_i \geq h$)
        send REPORT to $father$;
    else
        sleep for $2\log^2 N$ rounds;
if received a REPORT-REQUEST from $father$
    respond with REPORT containing x_i, l, and h;

Fig. 11. HRMA (Leaf Node)

children. This makes the cost of a single step $O(N)$. The relative cost gain of *HRMA* versus Improved Value (the cheapest flat monitoring algorithm) for maximal load trace is shown in Figure 10.

As one can see the relative cost gain of *HRMA* grows with the number of domains. *HRMA* does not save communication on multicasting of the notifications about the global TVEs. It saves communication by suppressing the non-significant local threshold violation events though. This is *HRMA*'s main feature. In our traces, *HRMA* pays $\approx 1.5 * N$ messages per round. This is due to the fact that some edges are traversed twice: one message to solicit the value, and another one to report the value itself. In the same circumstances, Improved Value algorithm pays $\approx 2 * N \log$ messages per round. Thus, the expected improvement is around $1.33 * \log N$ and is indeed expected to grow with the number of domains, N. This is very important observation since it suggests that *HRMA* is scalable.

if (REPORT msgs m_l, m_r from $left$ and $right$)
    $l_{left} \leftarrow m_l.l$; $h_{left} \leftarrow m_l.h$;
    $l_{right} \leftarrow m_r.l$; $h_{right} \leftarrow m_r.h$;
    if ($m_l.x + m_r.x \geq h$ or $m_l.x + m_r.x \leq l$)
        send REPORT message with values $m_l.x + m_r.x$ to $father$;
if (received REPORT message m_l from $left$)
    $l_{left} \leftarrow m_l.l$; $h_{left} \leftarrow m_r.h$;
    if ($m_l.x + right.h \geq h$ or $m_l.x + right.l \leq l$)
        send REPORT-REQUEST message to $right$;
if (received REPORT message m_r from $right$)
    $l_{right} \leftarrow m_l.l$; $h_{right} \leftarrow m_r.h$;
    if ($m_r.x + left.h \geq h$ or $m._rx + left.l \leq l$)
        send REPORT-REQUEST message to $left$;
if (received REPORT-REQUEST message from $father$)
    $h \leftarrow h_{left} + h_{right}$; //request missing values from children
    $l \leftarrow l_{left} + l_{right}$; //if needed
    respond with REPORT containing $left.x + right.x, h, l$
if (received UPDATE message from $father$, with α_1, α_2)
    forward this message to both children;

Fig. 12. HRMA (Intermediate Node)

At time 0:
  Let $L = thresh[i] < f(0) < thresh[i+1] = H$;
  $\alpha_1 \leftarrow H/f(0)$; $\alpha_2 \leftarrow f(0)/L$;

if (REPORT(s) m_l, m_r from $left$, or $right$, or both)
    act as an intermediate node;
    if ($m_l.x + m_r.x \geq h$ or $m_l.x + m_r.x \leq l$) //global event
    find new L and H;
    calculate new α_1, and α_2;
        send UPDATE with new L, H, α_1, and α_2 to $left$ and $right$;

Fig. 13. HRMA (the Root Node)

XI. Adaptive HRMA

Can we further improve the performance of *HRMA*? One may observe, that in its current form *HRMA* is not adaptive. Namely, if there is no global TVE, but there exist local threshold events, then at some level in the hierarchy, some intermediate node stops the alarm verification process. However, the algorithm learns no lesson from this. Therefore, if the situation remains similar also for the next step, the unnecessary verification process will repeat itself since the local thresholds of the leaves did not change. This is pure overhead. If we treat false alarms similarly to actual TVEs events, this overhead seemingly can be reduced.

In particular, there is an apparent trade-off between leaving the local thresholds in the subtree intact in case of a false alarm in this subtree, and changing them. The latter requires multicasting in the scope of this subtree, and therefore increases the communication cost of the algorithm at this round. However, if it succeeds in tuning the thresholds in such a way that no false alarms would follow, the total communication cost of the algorithm will be reduced over time.

To this end we define two simple modifications to the basic *HRMA*.

- **Oblivious Adaptive HRMA:** When a false alarm is detected at time t by the root of some subtree (an intermediate node i), this node computes $\alpha_1 = f_i(t)/l$, and $\alpha_2 = h/f_i(t)$ where $f_i(t)$ being the sum of the leaf variables of the subtree rooted at i, and l, h being the lower and higher local thresholds of this node respectively. Then, i multicasts the UPDATE message containing α_1, α_2, and indicating that there was no global threshold event. The nodes in the subtree recalculate their local thresholds using these values as they would upon reception of this message from the global root. However they do not change their current estimation of the group size.
- **Conscious Adaptive HRMA:** Similar to Oblivious. The main difference is that the root of the subtree assess the chances of a successful thresholds tuning by comparing the distance of the current value of the sum for its subtree with the median of the interval defined by its l, and h threshold values. The chances of suppressing false alarms are highest when the current value of the subtree is close to the median, and decrease as it gets close to either l, or h. Accordingly, Conscious Adaptive *HRMA* sends UPDATE message with the higher probability in the former case, and lowers the probability of sending an UPDATE message in the latter case.

Figure 14 demonstrates the adaptivity trade-off. For a fixed number of domains, we vary the trace duration. As one would expect, for shorter sessions the adaptivity heuristics are inferior to the basic form of *HRMA*. However, as sessions become longer, they become advantageous over the basic *HRMA*. Oblivious Adaptive *HRMA* is always inferior to the the Conscious and and Basic variants when sessions are short. This is very natural since Oblivious Adaptive *HRMA* needs more time to compensate for the extra cost it spent on updating the thresholds.

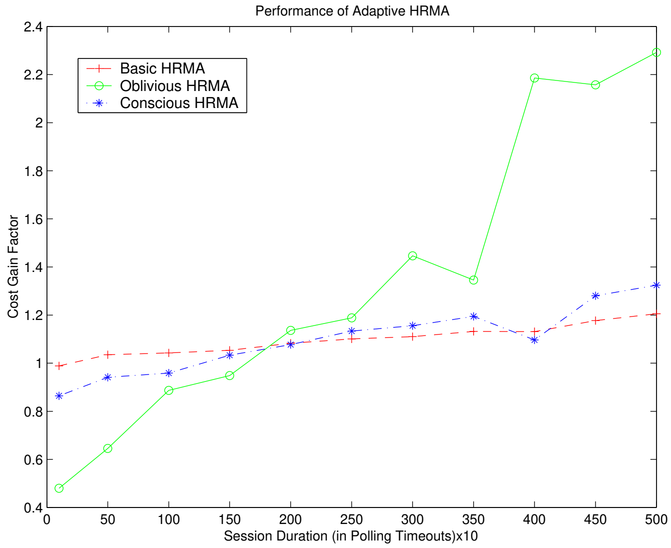

Fig. 14. Performance of Adaptive Heuristics

XII. Validation with Real Traces

In this section we evaluate *HRMA* vs Ob_h using the real membership traces collected using *mlisten* tool [17] in 1996. The purpose of this study is to inspect the behavior of *HRMA* when statistical assumptions on the user behavior used so far are not necessarily preserved. Mlisten generates a separate record for every host receiver. Each record indicates the receiver's IP address, the class D address of the group, port, join time, and stay time in seconds. For each class D address, we treated the network portion of the IP address as an individual multicast ISP. Then for every such ISP we have calculated the number of host receivers at any given moment by counting the different host parts of the IP addresses of receivers. In total there were about $44,000$ "ISP" domains identified this way, and about 200 different multicast addresses. As indicated by [18] the measurements performed in 1999 suggest that MBone did not grow substantially larger since then. In fact, it even shrank down.

Most multicast groups that were observed by us were quite small ranging from a few users to a few tens of users at any given moment, and have not exhibited particularly high activity. See [18] for discussion on the reasons for this.

However, this makes this data even more interesting. So far our algorithms have been evaluated on the synthetic data that simulated very large groups (thousands of highly dynamic users) spanning hundreds of domains. This may be way too far from the current state of the Internet where the multicast groups are much smaller and less active. In a sense, this is the opposite of our maximal load scenarios being inspected so far. The question that we asked was whether the hierarchical reactive monitoring will be useful also for estimating the size of very small groups in such a way that it would be useful for pricing applications.

To this end we revisit the technique for global thresholds calculations. The percentages of the total user population are meaningless for small groups. Far more important is to detect changes in the group from its current size with a predefined precision γ. Instead of having a full scale of global thresholds we define only a single pair of thresholds $H \stackrel{\mathrm{def}}{=} \gamma * f(0)$, $L \stackrel{\mathrm{def}}{=} f(0)/\gamma$, where $f(0)$ is the total size of the group as measured at time 0. The local thresholds l, h are computed as before. If at some time instance t, either of the global thresholds is violated, *HRMA* re-calculates L, H using $f(t)$. This way we always know the size of the group with precision γ as being set by the pricing application.

Out of ≈ 200 different traces that we created using the methodology above, we found only 11 traces with considerable activity. All these traces were comprised out of 256 ISPs. We analyzed only the first $300,000$ seconds (≈ 80 hours) of these traces. Figure 15 illustrates performance of *HRMA* variants for various values of γ. One thing to notice is that on the average Conscious Adaptive *HRMA* obtained exactly the same results as the Basic *HRMA*. This means that the distribution of values in the tree was always such that the values of the subtrees were far from the median of the intervals defined by their local thresholds. We do not have an explanation for this. Also, it would be difficult to come to a decisive conclusion without inspecting more traces. However, the traces we used represent a certain type of real life traffic, and *HRMA* performance on these traces is very satisfactory. The cost gain factor of *HRMA* versus other known techniques represented by OB_h ranges from 5 to 10 on the average. As one would expect, the variance is high because the sample is too small. The variance is lower when threshold values are lower which generates more threshold events irrespective of the nature of the trace. It gets higher when thresholds grow. The variance diminishes as threshold grow even more since large thresholds prevent any local threshold violation events from occurring, and the cost gain factor eventually goes to infinity.

XIII. Conclusions and Future Work

We presented simple and generic algorithmic framework for multicast group size monitoring. This framework, being realized as generic management service independent from the routing, is useful for usage-based pricing of multicast traffic. We demonstrated through analysis and simulations that hierarchical reactive monitoring is a powerful technique. The communication costs it incurs are low. It reduces management bandwidth requirements by factor of 2 to 10, on the average, compared to other existing techniques without degrading the precision of estimation. These results have been validated with the real datasets available to us at the moment.

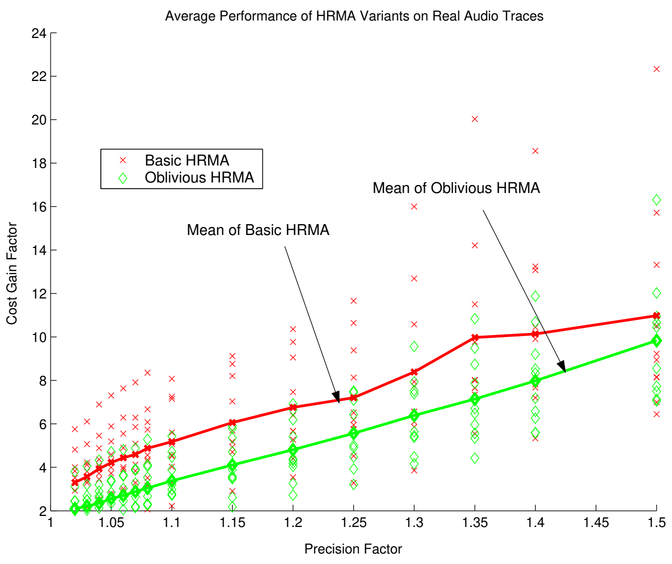

Fig. 15. Performance of HRMAn on the Real Traces

Future directions include studying of the newer real-life datasets, experimenting with different distributions of the receivers over domains and different join/leave statistical models. Further development of the algorithms is necessary to accommodate various additional accounting requirements of the pricing schemes, such as *e.g.*, geographical constraints.

Acknowledgments

We would like to thank Kevin Almeroth, and Robert Chalmers for sharing their MBone traces with us.

References

[1] J. Chuang and M. Sirbu, "Pricing Multicast Communication: a Cost Based Approach," in *INET'98*, Geneva, Switzerland, July 1998.

[2] T. Henderson and S.N. Bhatti, "Protocol independent multicast pricing," in *10th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV'00)*, The University of North Carolina, USA, June 2000.

[3] Hugh W. Holbrook and David R. Cheriton, "Multicast channels: Express support for large-scale single-source applications," in *ACM SIGCOMM'99*, Harvard, MA, USA, Sept. 1999.

[4] Mark Dilman and Danny Raz, "Efficient reactive monitoring," *IEEE Journal on Selected Areas in Communications (JSAC), special issue on recent advances in network management*, Apr. 2001.

[5] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: a transport protocol for real-time applications," Jan. 1996, RFC 1889.

[6] S. Floyd, V. Jacobson, S. McCanne, C. Liu, and L. Zhang, "A reliable multicast framework for ligh-weight sessions and application level framing," *IEEE/ACM Transactions on Networking*, vol. 5, no. 6, pp. 784–803, Dec. 1997.

[7] Dan Rubenstein, Jim Kurose, and Don Towsley, "A study of proactive hybrid fec/arq and scalable feedback techniques for reliable rea-time multicast," *Computer Communications Journal*, vol. 24, no. 5–6, pp. 563–574, Mar. 2001.

[8] J-C. Bolot, T. Turletti, and I. Wakeman, "Scalable feedback control for multicast video distribution in the internet," in *ACM SIGCOMM'94*, London, UK, Sept. 1994, pp. 58–67.

[9] C. Liu and J. Nonnenmacher, "Broadcast audience estimation," in *IEEE INFOCOM'00*, Tel-Aviv, Israel, Mar. 2000, vol. 2, pp. 952–960.

[10] Jorg Nonnenmacher and Ernst W. Biersack, "Optimal multicast feedback," in *IEEE INFOCOM'98*, San-Francisco, California, USA, Mar. 1998, pp. 964–971.

[11] T. Friedman and D. Towsley, "Multicast session membership size estimation," in *IEEE INFOCOM'99*, New York City, NY, USA, Mar. 1999, vol. 2, pp. 965–972.

[12] Sara Alouf, Eitan Altman, and Philippe Nain, "Optimal on-line estimation of the size of a dynamic multicast group," in *INFOCOM'02*, New Yorc City, New York, USA, June 2002.

[13] G. Phillips, S. Shenker, and H. Tangmunarunkit, "Comments on the Chuang-Sirbu Scaling Law," in *ACM SIGCOMM'99*, Cambridge, Massachsetts, USA, Aug. 1999.

[14] Robert C. Chalmers and Kevin C. Almeroth, "Modelling the Branching Characteristcs and Efficiency Gains in Global Multicast Trees," in *INFOCOM 2001*, Anchorage, Alaska, USA, Apr. 2001.

[15] S. Herzog, S. Shenker, and D. Estrin, "Sharing the cost of multicast trees: An axiomatic analysis," in *SIGCOMM'95*, Cambridge, MA, USA, Aug. 1995, pp. 315–327.

[16] B. Briscoe, "The direction of value flow in connection-less networks," in *NGC'99*, Pisa, Italy, Nov. 1999.

[17] K. Almeroth and M. Ammar, "Multicast group behavior in the internet's multicast backbone (mbone)," *IEEE Communications*, June 1997.

[18] K. Sarac and Kevin C. Almeroth, "A Long-Term Analysis of Growth and Usage Patterns in the Multicast Backbone (MBone)," in *INFOCOM'00*, Tel-Aviv, Israel, Mar. 2000.

[19] M. Handley, "SDR: Session Directory Tool," `ftp://cs.ucl.ac.uk/mice/sdr/`, Nov. 1995.

[20] Robert C. Chalmers and Kevin C. Almeroth, "Developing a Multicast Metric," in *GLOBECOM 2000*, San-Francisco, CA, USA, Dec. 2000.

[21] D. Estrin, D. Farinacci, A. Helmy, D. Thaler, S. Deering, M. Handley, V. Jacobson, C. Liu, P. Sharma, and L. Wei, "Protocol independent multicast sparse mode (PIM-SM): Protocol specification," June 1997, RFC2117.

[22] W. Fenner, "Internet group management protocol, version 2," Nov. 1997, RFC2236.

[23] S. Deering, "Host extensions for ip multicasting, appendix i: Internet group management protocol," Aug. 1989, RFC1112.

[24] Brad Cain, Steve Deering, Bill Fenner, Isidor Kouvelas, and Ajit Thyagarajan, "Internet group management protocol, version 3," `http://search.ietf.org/internet-drafts/draft-ietf-idmr-igmp-v3-11.txt`, Internet Draft.

[25] J. Jiao, S. Naqvi, D. Raz, and B. Sugla, "Toward efficient monitoring," *IEEE Journal on Selected Areas in Communications (JSAC), special issue on recent advances in network management and operations*, vol. 5, no. 18, pp. 723–732, May 2000.