Placement of Proxy Servers to Support Server-Based Reliable Multicast

Raja Jothi and Balaji Raghavachari
Department of Computer Science, University of Texas at Dallas
Richardson, TX 75083, USA
{raja, rbk}@utdallas.edu

***Abstract*— We consider the problem of placing k proxies on a n node network to support server-based reliable multicast. Typically, in a multicast connection, the sender transmits information to all the receivers in the multicast group. Failures in transmission causes receivers to send retransmission requests to the sender. Due to high retransmission requests, the sender could become a potential hot-spot in the network. Also, receivers far away from the sender could experience high delays in receiving the retransmitted information. To prevent these and other issues, the notion of placing proxy (or repair) servers across the network was introduced, so that receivers could send their retransmission requests to the nearby proxy servers than to the source. In this context, we consider static placement of proxies on a network in which multicast requests (trees) are generated dynamically. The placement of proxies should be such that the proxies are utilized efficiently by the dynamically generated multicast trees.**

In this paper, we present three heuristics for efficient placement of proxies on any given network. We compare the performance of our heuristics with the best available heuristic KMPC [18] as well as the random proxy placement heuristic. Experiment results show that one of our heuristics, the NASPC, outperforms all the other heuristics with respect to all performance measures considered.

***Keywords*— Reliable Multicast, Proxy Servers, Proxy Placement, Network Management.**

I. Introduction

Companies, Internet Service Providers and other large campus and communications networks often need to multicast messages originating at a central location. Emerging applications such as video conferencing, instant messaging, pay-per-view special events make use of such services. The nodes participating in a multicast are modeled as a tree. Depending on the receiving nodes, the shape of the tree and hence the participating nodes forming the tree also change. The tree is thus a dynamic structure superimposed on a static network comprised of nodes and links between these nodes. At any point in time, multiple multicast trees may be active.

While it is natural to have the message originating at any one node, it is not efficient, in the case of requests for retransmissions (due to errors in original transmissions and or other failures), to involve the same original (source) node, with all the attendant overheads of multiple link negotiations, multiple levels of node traversals and the added problem of overwhelming the source node with too many simultaneous requests. It would be far more efficient if caches of the message are stored in proxies along the way. Intermediate and multiple nodes storing and forwarding copies of messages, upon request, would enable both the distribution and the balancing of the load on the originating source node and speedier responses to the nodes requesting retransmissions. Such nodes are called "repair servers" or "proxy servers". An added benefit is the reduction in the traffic on the network, since the lengths that both requests and retransmissions travel will be minimized in such a scheme.

In this model, as the messages arrive from the source, a proxy server sitting on that multicast tree starts storing those messages. These messages are purged when either all recipients in the subtree corresponding to the server have acknowledged receiving them or when the buffer gets full and a scheduling algorithm deletes an item to make way for future packets. When nodes detect missing packets, they send a retransmit request to the source, and this request is intercepted by its nearest ancestor acting as a proxy server which has a copy of those packets.

Proxy server placement is non-trivial. For, while the trees are generated dynamically, the server placement is made only statically. Proxy servers cannot be created dynamically, tailored to dynamic multicast trees on-demand, because their specific hardware and software requirements are not readily found on all network nodes. Nodes acting as proxies require substantially more storage and processing power, and therefore are substantially more expensive to operate than normal routers. We are thus left with the problem of selecting, a-priori, optimal number of nodes and node sites to host the proxy servers on a network, without prior knowledge of the multicast trees that will arise. An allocation of nodes as proxies is evaluated on a number of metrics, such as the average and worst-case number of hops traversed by a retransmit request before it finds a proxy server (latency), and the number of proxies that are able to serve a given multicast tree (hit ratio).

A. Problem Formulation

Given a network, represented by a graph $G = (V, E)$, where V is the set of nodes (nodes can be thought of as routers) and E is the link set. A proxy server can be placed at any node, and once a proxy is placed at a node, it cannot be relocated to another node. In simple terms, a proxy server can be thought of as a specially designed host co-located with a router, or as a router with proxy server capabilities integrated into it. Multicast trees on this network are established and released dynamically (online), i.e., no prior information about the multicast trees (senders and receivers) are known.

Under these circumstances, we are asked to place k proxy servers at k node locations such that proxy servers are utilized to their fullest extent. Utilization of a proxy server can be interpreted as the efficient usage of the proxy server by as many multicast trees as possible. Naturally, the placement of proxies should be such that the latency (or delay) experienced by a receiver is kept to a minimum. The latency for a receiver to receive a packet after it has issued a negative acknowledgment (request for retransmission), is defined to be the distance (number of hops) between that receiver and the closest proxy server along its path to the sender (source). In this paper, the words source and sender are used interchangeably.

B. Previous Works

Numerous schemes have been proposed in recent years to handle proxy placements on the Internet satisfying various requirements. The fundamental requirement is fast local recovery (retransmissions) of lost packets. In general, local recovery schemes are classified into two types: (i) receiver-based recovery schemes and (ii) server-based recovery schemes [12]. For details on local recovery schemes, we refer the reader to [1], [3], [13], [14], [22] for server-based recovery schemes and [4], [6], [7], [15], [19], [20] for client-based recovery schemes.

Li et al. [16] considered the placement of k proxies on a n node tree network. They presented a dynamic programming approach to find the optimal placement of proxies that minimizes the sum of latencies of all the nodes in the network. In [8], Jia et al. considered the problem of placing k proxies on an n node tree network with m proxies already placed on it. They use a dynamic programming formulation to find an optimal solution that minimizes the maximum latency among others. Kamath et al. [11] considered the proxy placement on Internet logical topology based on the network topology and routing policies. Qiu et al. [21] studied the online problem of placing web server replicas in content distribution networks. They use workload information to make informed placement decisions. Lin and Yang [18] were the first ones to present a set of heuristics for the problem discussed in this paper. Among the 3 heuristics that they presented, the *k-maximum shortest path count* (KMPC) heuristic obtains the best results. KMPC places proxies on the nodes through which the most number of shortest paths pass through.

II. Heuristics For Proxy Placement

For the problem defined in this paper, we propose the following three heuristics for efficient placement of proxy servers on any given network.

A. The k-median heuristic (MEDIAN)

Initially a proxy is placed at a node such that the sum of the distance (number of hops) from every node to this proxy is minimum. The remaining $k-1$ proxies are placed one at a time based on the policy that every time a new proxy is placed on one of the nodes, it is made sure that the sum of the distance from every node to its closest proxy is minimum.

The k-median heuristic above was designed based on the k-median problem [17], which is NP-hard. In the k-median problem, we are asked to find k nodes (medians) in an n-node network, such that the sum of the distances from each node to its nearest median node is minimum.

B. The non-adjacent maximum degree count heuristic (NAMDC)

The nodes are sorted in non-increasing order based on the number of links connected to them. Number of links connected to a node is otherwise known as its degree. The k proxies are placed at the first k non-adjacent nodes of the sorted list.

The intuition behind the NAMDC heuristic is that there is a high probability that most of the paths in a network passes through high degree nodes. The reason behind placing proxies on non-adjacent nodes is to prevent clustering of proxies.

C. The non-adjacent shortest path count heuristic (NASPC)

An all-pair shortest path algorithm is first used to find the shortest paths between any two nodes. Then the number of shortest paths passing through each node is counted. The nodes are sorted in non-increasing order based on the number of shortest paths passing through them. The k proxies are placed at the first k non-adjacent nodes of the sorted list.

In the NASPC heuristic, proxies are placed such that no two proxies are next to each other. The reason behind this is that in a low-degree network, such as the Internet whose average degree is in the range 3 to 4, it is important to spread the proxies than to cluster them together. Clustering the proxies serves no purpose as it will only increase the average latency of the nodes.

To understand the importance of placing the proxies at non-adjacent nodes, consider an instance in which most of the shortest paths are passing through a low-degree

node, say A. Since A is a low-degree node, there is a high probability that majority of the paths passing through A would pass through A's neighboring nodes as well. In which case, if we do not restrict the placement of proxies at non-adjacent nodes, there is a high chance that the algorithm will place proxies at A's neighboring nodes as well, which could useless and detrimental if the network size is fairly large and the number of proxies that are to be placed on the network is relatively very small. Hence, it is very important that the proposed method prevents clustering of proxies.

III. Simulation And Performance Evaluation

We performed simulations to study the performance of our heuristics against the current best heuristic KMPC [18], and the random proxy placement heuristic (in which the k proxies are placed at k random nodes).

A. Network Model

The network model used in this paper is similar to the ones used in [5], [9], [10], [21], [23], and possess some of the characteristics of a real network. We generated random graphs to represent the network model. The non-deterministic nature of random graphs is critical as it ensures that the quality of the network design algorithms are independent of the network configuration. In our network model, the nodes are randomly distributed on a rectangular unit grid. Existence of a link between any two pair of nodes, u and v, is decided based on a link existence probability function $P_e(u, v)$. One of the most popular link probability function, known as the the Waxman model [23], is given by

$$P_e(u, v) = \beta \exp\Big(\frac{-dist(u, v)}{L\alpha}\Big)$$

where $dist(u, v)$ is the distance between nodes u and v and L is the maximum distance between any two nodes in the graph. Parameter $0 < \beta \leq 1$ controls the density of links in the graph. Increase in the β value increases the number of links in the network. Parameter $0 < \alpha \leq 1$ controls the density of relatively short links in the graph. Increase in the value of α will increase the density of short links in relation to the longer links. One major problem with the Waxman model is that as the number of nodes in the network increases, the number of links from each node (degree of the node) also increases. To maintain the characteristics of the graph under scaling, a modification has to be introduced [5], which scales the value of $P_e(u, v)$ by a factor related to the number of nodes, n, in the graph. And to ensure that the mean degree remains relatively constant, another scale factor between two random points must be introduced [9]. The modified link probability function now looks as follows:

$$P_e(u, v) = \frac{\psi\delta}{n}\beta \exp\Big(\frac{-dist(u, v)}{L\alpha}\Big)$$

where δ is the mean degree of a node and n is the number of nodes in the graph. Graphs generated with values of $\alpha = 0.25$ and $\beta = 0.2$ roughly resembles the geographical maps of major nodes in the Internet [5]. To generate graphs with degree $\delta = 3, \alpha = 0.25$ and $\beta = 0.2$, the value for ψ has to be set to approximately 25 [5]. After generating links using the above link probability function, there is a possibility that the resultant graph might not be connected, i.e. there may be more than one connected component. This can be easily detected using a minimum spanning tree algorithm. If there is more than one component, then necessary links are added to combine individual components into one component, which will be the final graph.

B. Multicast Tree Construction Algorithms

Constructing a multicast tree spanning m nodes of a n node network is often modeled as the Steiner tree problem, which is NP-Complete. Since the computation of Steiner trees is intractable, several Steiner tree heuristics have been proposed. For a detailed list of such heuristics and their performance, we refer the reader to [2]. We considered the following two well-known heuristics for the construction of multicast trees.

- **Point-to-Point Shortest Path Heuristic (PPSPH):** Every receiver node in the multicast tree is connected to the source node along the shortest path to the source node.
- **Shortest Path Heuristic (SPH):** SPH starts with an initial multicast tree containing just the source node. It then repeatedly adds the next closest receiver node to the multicast tree by the shortest path between the receiver node and the tree. SPH terminates when all receivers are added to the tree.

Experimental results [2] show that SPH outperforms PPSPH heuristic by 5% on average.

C. Performance Measures

As in [18], for each data point in our simulation chart, we generated 100 graphs. The proxy placement for each graph was evaluated by establishing 50 multicast trees. For each multicast tree, the sender and the receiver nodes were randomly chosen from the nodes in the graph. Since SPH is efficient than the PPSPH heuristic [2], we used SPH to construct source-rooted multicast trees.

We define *latency* of a receiver node to be its distance (number of hops) to the closest proxy server along its path to the source node (sender). In other words, latency is the number of hops for a receiver node to receive a retransmitted packet after it issues a negative acknowledgment (request for retransmission). Since the major reason for deploying proxies is to reduce the latency experienced by a multicasting member (receiver node), we consider latency to be the major performance measure among others. We consider the following three performance measures to evaluate the performance of our heuristics:

- **Average latency:** The average of the latencies of the receiver nodes.
- **Average worst-case latency:** The average of the latencies of the high-latency nodes (a receiver in a multicast tree that experiences high latency is called the high-latency node).
- **Average hit ratio:** The average of the hit ratios of the proxy servers. For a given graph G and a number of multicast trees defined on G, the hit ratio of a proxy server is the percentage of the total number of mulitcast trees that use the proxy server as one of its proxies.

D. Simulation Results

We compare the performance of our heuristics with the current best heuristic KMPC [18] and the random placement of proxies (RANDOM). Unless otherwise mentioned, the number of nodes in each graph is 1000. For comparison purposes, as in [18], we kept the size of the multicast trees to be 50% of the number of nodes in the graph, and number of proxy servers to be 4% of the number of nodes in the graph.

We tested the heuristics under various scenarios: (i) different number of nodes in the graph (ii) different sizes of the multicast tree (iii) different number of proxy servers. Our experimental results show that NASPC performs the best among all the heuristics tested, including the current best KMPC [18]. In what follows, we describe in detail, the results obtained from our simulations.

1) Average latency: Figs. 1, 2 and 3 depict the results for the average latency experienced by a receiver for different number of nodes in the network, different sizes of multicast group and different number of proxies, respectively. All our heuristics perform much better than the random placement heuristic (RANDOM). When compared to the current best heuristic KMPC, NASPC and NAMDC produce smaller latencies. Interestingly, contrary to expectations, MEDIAN obtained relatively higher latencies when compared to KMPC. Results were consistent, regardless of the number of nodes in the network, multicast group size or the number of proxies. NASPC performs the best with an improvement ranging from 2-3% over the current best KMPC. The average latency increases as the network size increase while it decreases as the number of proxies increase. Increase in the multicast group size does not have an effect on the average latency, which is ideally what is needed.

2) Average worst-case latency: Simulation results for average worst-case latency is shown in Figs. 4, 5 and 6. Simulations were performed for varying number of nodes. The effects of change in multicast group size and number of proxies were also examined. All heuristics obtained significantly better results when compared to RANDOM. All the heuristics (MEDIAN, NASPC and NAMDC) proposed in this paper outperformed the current best heuristic, the KMPC, with NASPC obtaining the best results. NASPC obtained improvements ranging from 2-3% over the current best KMPC. Increases in the multicast group size and the network size results in an increase in the average worst-case latency. As expected, the average worst-case latency decreased as the number of proxies was increased.

3) Average hit ratio: The average hit ratio was computed under varying circumstances: different number of nodes in the network, different sizes of multicast group and different number of proxies. RANDOM performs much worse than the other heuristics. Surprisingly, MEDIAN did not perform as well as expected. There was not much difference in the performance of KMPC and NASPC. Both were equally good and outperformed all other heuristics. Increase in the number of proxies results in a slight decrease in the hit ratio as not all multicast trees can cover all the proxies. Increase in the size of the multicast tree results in an increase in the hit ratio, as larger mutlicast trees spans more proxy nodes. Slight decrease in the hit ratio was noticed when the network size was increased.

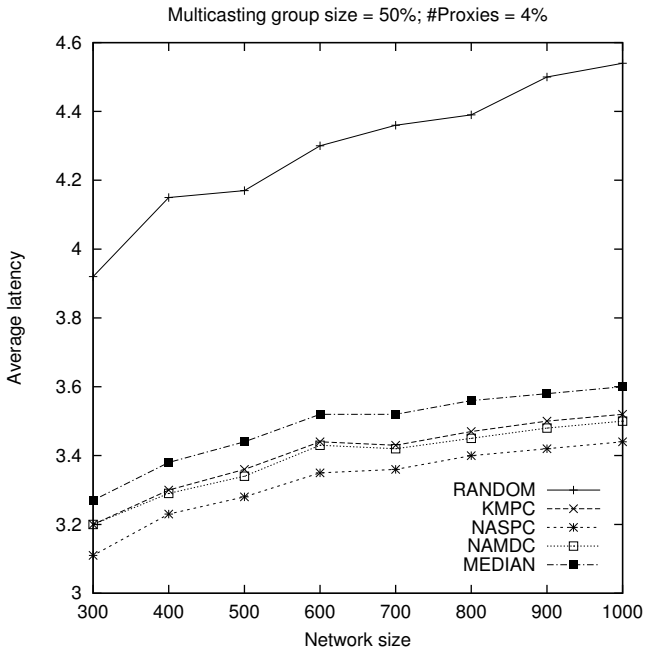

Fig. 1. *Average latency for networks of varying sizes.*

IV. Conclusion

We proposed three heuristics (MEDIAN, NASPC, NAMDC) for the placement of proxy servers to support server-based reliable multicast. Simulations results show that NASPC performs the best among the three heuristics. We considered the average latency, average worst-case latency and hit ratio as the performance measures to test the quality of the proposed heuristics. Experimental results show that NASPC outperforms KMPC (the current best

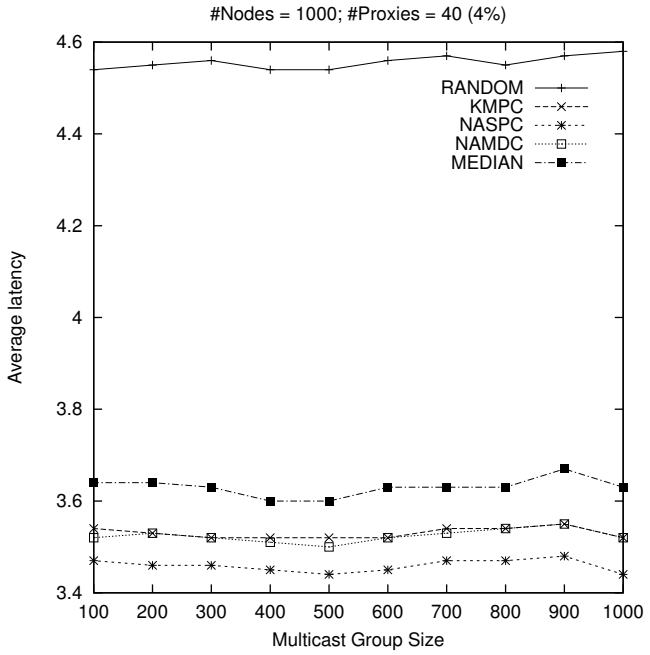

Fig. 2. *Average latency for varying multicast group sizes.*

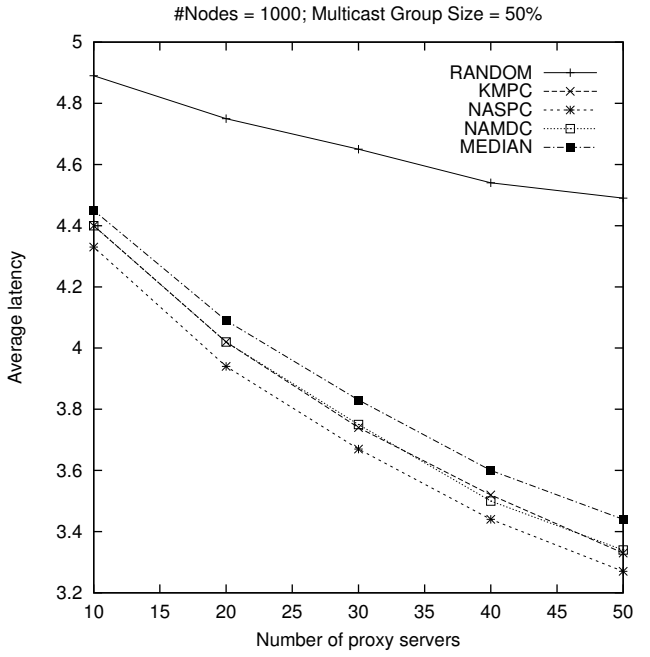

Fig. 3. *Average latency for varying number of proxies.*

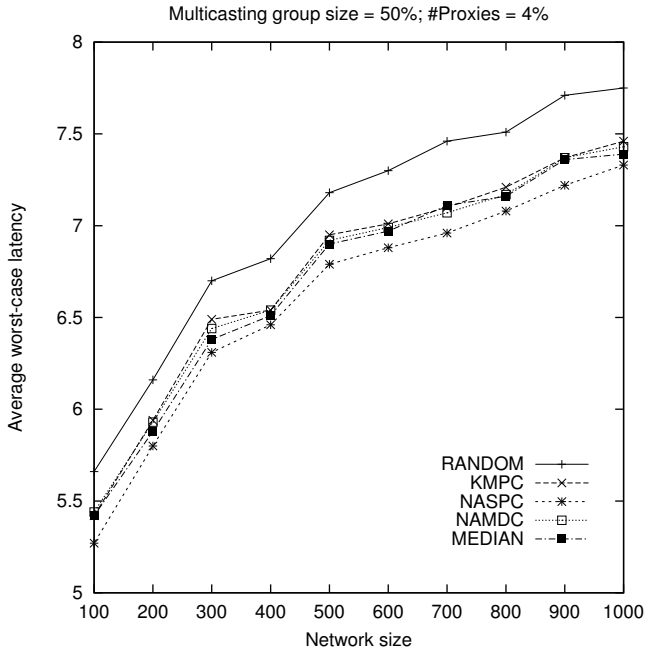

Fig. 4. *Average worst-case latencies for networks of varying sizes.*

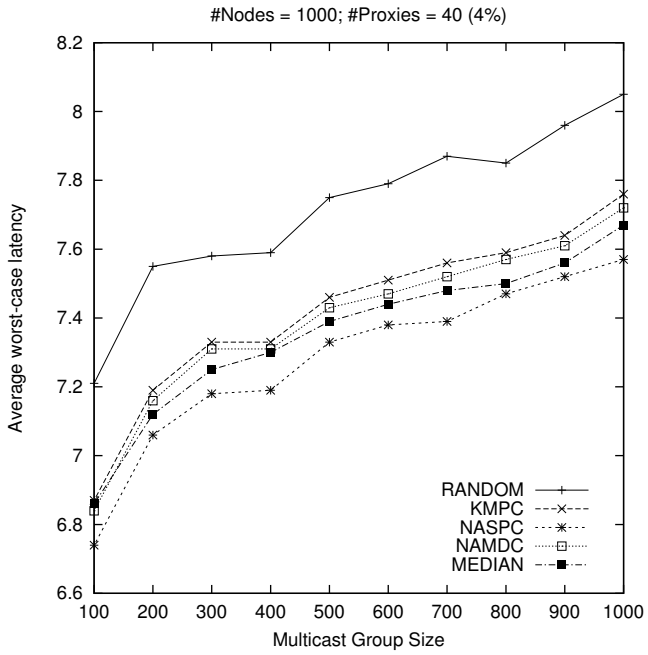

Fig. 5. *Average worst-case latencies for varying multicast group sizes.*

heuristic available for the problem considered in this paper) with respect to all the performance measures considered. To ensure that the heuristics are not input dependent, they were put to test under varying input configurations such as: networks of different sizes, mutlicast groups of different sizes, and different number of proxies. The performance of the heuristics was consistent regardless of the input configurations.

ACKNOWLEDGMENT

This research was supported by the National Science Foundation under grant CCR-9820902. We would like to thank Sathya Peri for bringing this problem to our attention.

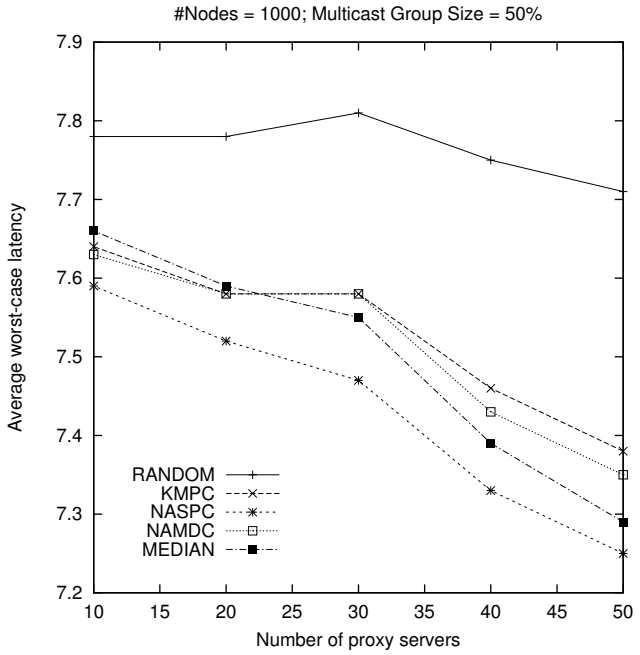

Fig. 6. *Average worst-case latencies for varying number of proxies.*

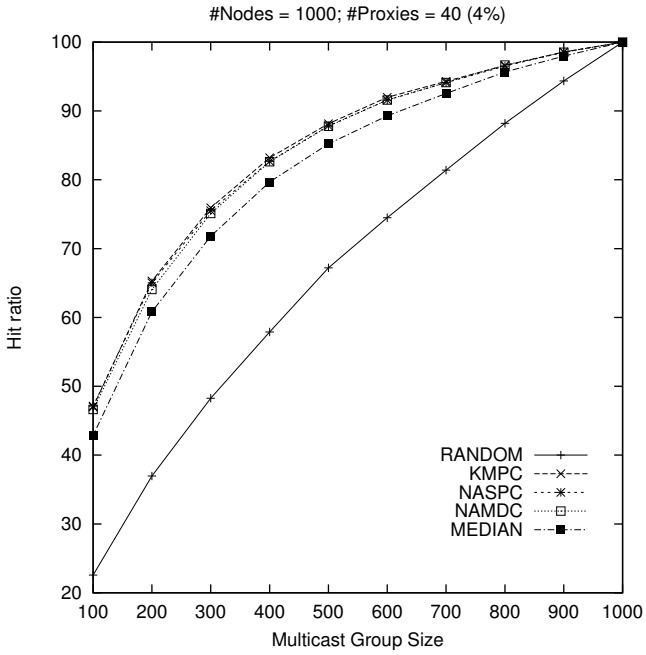

Fig. 8. *Hit ratios for varying multicast group sizes.*

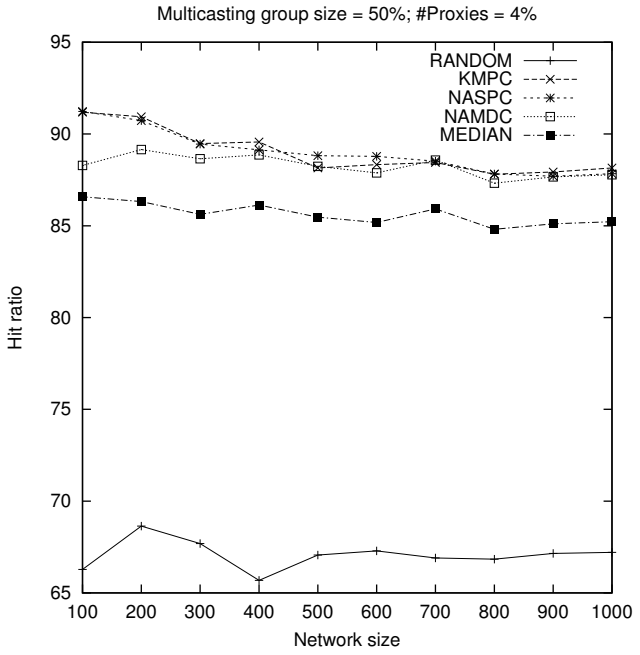

Fig. 7. *Hit ratios for networks of varying sizes.*

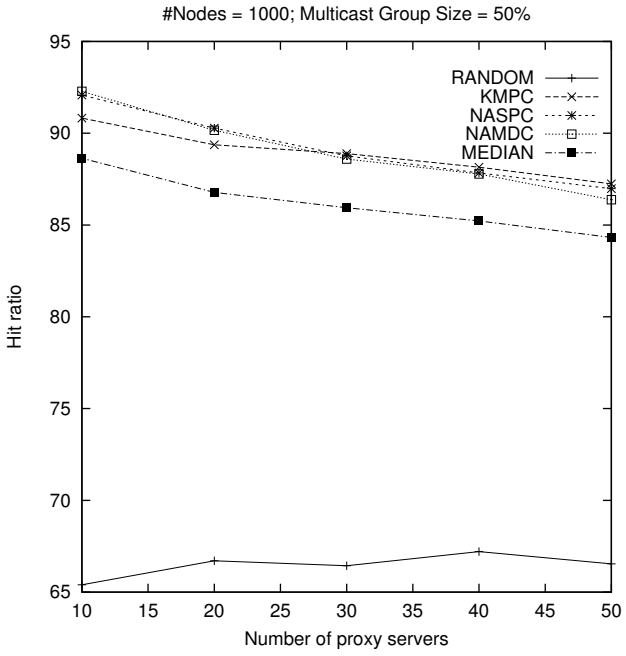

Fig. 9. *Hit ratios for varying number of proxies.*

REFERENCES

[1] A. Azcorra and M. Calderón, "A network-based approach to reliable multicast protocols," *Proc. Protocols for Multimedia Sys. (PROMS)*, pp. 393-404, 1995.

[2] F. Bauer and A. Varma, "Degree-constrained multicasting in point-to-point networks," *IEEE INFOCOM*,pp. 369-376, 1995.

[3] M. Calderón, M. Sedano, A. Azcorra, and C. Alonso, "Active network support for multicast applications," *IEEE Network*, 1998.

[4] A.M. Costello and S. McCanne, "Search party: Using randomcast for reliable multicast with local recovery," *Proc. IEEE INFOCOM*, 1999.

[5] J.M.S. Doar, "Multicasting in the asynchronous transfer mode environment," *Ph.D. Dissertation*, Tech. report no.298, Univ. of Cambridge, 1993.

[6] S. Floyd, V. Jacobson, C.G. Liu, s.McCanne, and L.Zhang, "A reliable multicast framework for light-weight sessions and application

level framing," *IEEE/ACM Trans. on Networking*, 1996.
[7] H.W. Holbrook, S.K. Singhal, and D.R. Cheriton, "Log-based receiver-reliable mutlicast for distributed interactive simulation," *Proc. ACM SIGCOMM*, 1995.
[8] X. Jia, D. Li, X. Hu, and D.Z. Du, "Optimal placement of web proxies for replicated web servers in the Internet," *The Computer Journal*, Vol. 44, No. 5, pp. 329-339, 2001.
[9] J. Kadirire, "Minimizing packet copies in multicasting routing by exploiting geographic spread," *ACM SIGCOMM Computer Communication Review*, Vol. 24, pp. 47-63, 1994.
[10] J. Kadirire and G. Knight, "Comparison of dynamic multicast routing algorithms for wide-area packet switched (asynchronous transfer mode) networks," *Proc. IEEE INFOCOM*, pp. 212-219, 1995.
[11] K.M. Kamath, H.S. Bassali, R.B. Hosamani, and L. Gao, "Policy-aware algorithms for proxy placement in the Internet," *Proc. Conf. on Internet Perf. and Control of Network Sys. (ITCOM)*, 2001.
[12] S.K. Kasera, J. Kurose, and D. Towsley, "A comparison of server-based and receiver-based local recovery approaches for scalable reliable multicast," *Proc. IEEE INFOCOM*, 1998.
[13] K.W. Lee, S. Ha, and V. Bharghavan, "IRMA: A reliable multicast architecture for the Internet," *Proc. IEEE INFOCOM*, 1999.
[14] L.W.H. Lehman, S.J. Garland, and D.L. Tennenhouse, "Active reliable multicast," *Proc. IEEE INFOCOM*, 1998.
[15] B.Levine and J.J. Garcá-Luna Aceves, "Improving Internet multicast with routing tables," *Proc. IEEE ICPN*, pp. 241-250, 1997.
[16] B. Li, M.J. Golin, G.F. Italiano, X. Deng, and K. Sohraby, "On the optimal placement of Web proxies in the Internet," *Proc. IEEE INFOCOM*, 1999.
[17] J-H. Lin, and J.S. Vitter, "-approximations with minimum packing constraint violation," *Proc. 24th ACM Symp. on Theory of Comp.*, pp. 771-782, 1992.
[18] H. Lin and K. Yang, "Placement of repair servers to support server-based reliable multicast," *Proc. IEEE ICC*, pp. 1802-1806, 2001.
[19] S. Paul, K.K. Sabnani, J.C.H. Lin, and S. Bhattacharyya, "Reliable multicast transport protocol (RMTP)," *IEEE JSAC*, Vol. 15, No. 3, pp. 407-421, 1997.
[20] C. Papadopoulos, G. Parulkar, and G. Varghese, "An error control scheme for large-scale multicast applications," *Proc. IEEE INFOCOM*, 1998.
[21] L. Qiu, V.N. Padbanabhan, and G.M. Voelker, "On the placement of web server replicas," *Proc. IEEE INFOCOM*, 2001.
[22] D. Rubenstaein, S. Kasera, D. Towsley, and J. Kurose, "Improving reliable multicast using active parity encoding services (APES)," *Proc. IEEE INFOCOM*, 1999.
[23] B.M. Waxman, "Routing of multipoint connections," *IEEE JSAC*, Vol. 6, No. 9, pp.1617-1622, 1988.