An adaptive membership management algorithm for application layer multicast

Rong, Bin; Khalil, Ibrahim; Tari, Zahir

An Adaptive Membership Algorithm for Application Layer Multicast

Bin Rong, Ibrahim Khalil, and Zahir Tari
School of Computer Science and Information Technology
RMIT University, Australia
{brong, ibrahimk, zahirt}@cs.rmit.edu.au

Abstract

Due to deployment difficulty of network layer multicast, application layer multicast is considered to be a good substitute for massive P2P video/audio streaming in large networks. However, in application layer multicast, the participating users join and leave the on-going session at will. Therefore, a scalable and reliable group membership management algorithm is necessary due to the highly dynamic nature of the overlay network, built on top of the Internet. Gossip-based algorithms seem to be a solution. However, most gossip-based membership management algorithms lack flexibility, and are unable to adapt to the ever-changing network dynamics, imposing roughly the same amount of overhead on the network. A new adaptive gossip-based membership management algorithm is proposed to bridge the gap. This algorithm captures the changes of the network and adjusts the parameter settings dynamically, bringing adaptivity and reducing overhead. Simulation results indicate a maximum of 50% *reduction can be achieved in terms of network overhead on core network components, such as backbone links and attached routers, without sacrificing reliability and scalability.*

1 Introduction

Membership management is crucial to group communication protocols, because the participating users need certain identifiers (e.g., IP address) to communicate with each other. In traditional IP multicast, group membership management is performed in a transparent way. Both senders and receivers register with the routers and routers perform membership management. However, in application layer multicast (ALM henceforth), there is no central server and the overlay is built on-the-fly, in a distributed way, rendering a robust and scalable membership management algorithm.

Epidemic or gossip-based algorithms seem to be a good candidate, and a gossip-based membership management algorithm has been published [4]. However, it lacks flexibility and imposes the same amount of overhead on the network regardless the current characteristics of the network. According to Kunwadee [12], most applications in ALM are short lived, and there are 3.3 requests from a single IP address during a session. In such a highly dynamic environment, the major concern is how to capture and communicate these changes among the remaining users in a timely and efficient fashion, and how to balance the network overhead, computational complexity and network performance. To address this problem, a new adaptive gossip-based membership management algorithm is proposed. The contributions of this paper are as follows. Firstly, an adaptive membership management algorithm for ALM. The parameter settings of the gossip algorithm are fine-tuned by dynamic weight setting throughout the session, in terms of the length of the gossip round and the scope of the gossip targets selection. The tuning process is done in such a way that it reflects the changes and the characteristics of the network. Secondly, a simple protocol to construct a reliable multicast tree for application layer multicast. Proactive measures are taken to enhance the reliability. It works by allowing every node to maintain a backup list of nodes to which they may contact in case of failures, and this can minimize the impact of failures.

The rest of the paper is organized as follows. Section 2 provides background information on ALM and gossip-based algorithms. Section 3 describes the details of the protocol. Section 4 describes the experimental setup. Simulation results are also provided. Section 5 overviews some of the existing work and puts our work in context. Finally, we conclude our work in section 6.

2 Background

2.1 Multicast and ALM

Multicast was proposed to overcome the shortcomings of Internet protocol (IP) and provide efficient multipoint delivery [2]. It works by sending one and only one copy of each packet along the so-called "multicast tree", realizing

0-7695-2622-5/06/$20.00 (c) 2006 IEEE

efficient usage of network resources. However, due to development and deployment issues, it has not been widely used. Consequently, a new scheme called *application layer multicast* has been proposed. Chu et al. [6] raised the idea of ALM. In ALM, data packets are replicated at end hosts rather than being replicated at routers inside the IP network. The end hosts form a logical layer atop the IP layer. Nevertheless, the large amount of control overhead used for membership management limits its use only to a small group of users. So, a scalable group membership management algorithm is necessary for the implementation of ALM.

2.2 Group Membership Management

Group membership management protocols are crucial to the success of multicast because they provide applications with the dynamic membership information. There are two types of membership management mechanisms: local group management and global multicast routing. In a traditional network layer multicast scheme, a local group management algorithm enables multicast routers to be aware of the presence of group members within their local networks by letting each participating member register with the router. Hence, it only applies to a single LAN or several LANs. In contrast, the global multicast routing mechanism learns the existence of the members by exchanging membership information among the routers distributed across wide-area networks [10]. The most common local group management mechanism is Internet Group Management Protocol (IGMP) [5]. It periodically updates membership information by using a query/reply model. However, none of these protocols are suitable for ALM, due to either large overhead or central point of failure.

2.3 Gossip-based algorithms

Gossip-based algorithms seem to be a good candidate to offer scalability and reliability at the same time [3]. It could be summarized as follows: each group member keeps a partial membership list and updates it using message gossiping. In each *gossip round*, which is a fixed time interval, each member chooses a fixed number of members, uniformly at random from its partial membership list, to send a copy of its membership list. The major drawback is its lack of flexibility, both in terms of time and space. The algorithm is unable to adapt to the ever-changing network dynamics, imposing roughly the same amount of overhead on the network regardless of the current characteristics of the network.

The proposed algorithm bridges the gap. It dynamically changes the gossip parameter settings (such as the length of the gossip round and the scope of gossip targets selection) in a way such that it reflects the changes in the network. It is adaptive and reduces the overhead by a maximum of 50%, not at the cost of reliability and scalability.

3 Protocol Details

A detailed description of the protocol is given in this section. The proposed algorithm works in control plane, decoupled from data delivery and manages the membership information in a distributed fashion. In order to make it self-contained, some terminology explanations are necessary.

3.1 Terminology and Metric

For simplicity, an ALM session with a single sender is considered here. Each node keeps partial membership information. Similar to the idea of [4], two separate lists are used. *InView* contains the nodes who know it and *OutView* contains the nodes it knows. As an example, in Figure 1, node 4's *InView* contains node 1, 2 and 3; it has node 5, 6 and 7 in its *OutView*.

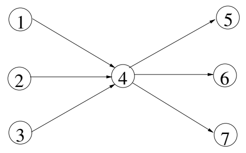

Figure 1. An example of membership list.

Nodes disseminate membership information to their neighbors periodically using message gossiping, and the *gossip round* is adjusted dynamically. In each round, nodes gossip to others in a pseudo-random way by choosing the targets preferentially, according to the metric defined later. Upon receipt of an update message, each node picks some nodes in a pseudo-random way from its *OutView* to propagate the message, just like the way in which some infectious disease spreads. Eventually, most of the nodes will get this message. However, it is not expected that all nodes will have the same view after several rounds of gossiping. The reason is the network is in a highly dynamic state throughout the multicast session. The overhead generated to let all the nodes have very accurate and the most up-to-date view about the entire network is too high. From another point of view, it is inefficient to get the data from other nodes who are very far away. For example, a user in Australia is more willing to get data locally, rather than from another user in Europe. So, the membership information contained in the gossiping message has different meaning for different users, and only part of the population may be interested in a particular piece of information. In order to count for this varying importance, the "goodness" of a node is defined as follows:

$$m = C \times \frac{B_i \times B_j}{D^2} \tag{1}$$

where C is the number of "children" it has, i.e., the number of nodes that are currently receiving data from it; B_i and B_j stand for the residual bandwidth of node i and j respectively, and D is the delay, or simply the network distance. The actual usage of this metric will be discussed later in this section. This metric of "goodness" is borrowed from physics. The idea is very simple: the goodness or desirability of a node is proportional to its available bandwidth, and is reverse to its network distance. That is to say, a "good" node should have high bandwidth and/or low delay. The rationale behind this metric is two-fold. Firstly, it has been proven that the *scale-free* property of the network is in favor of the information dissemination, especially if those nodes with higher degree, or named as hubs, are reached by the gossip message, and the gossip message will propagate through the network at a very high speed [9]. This has also been observed in Gnutella network. The proposed algorithm makes use of this property. By directing the gossip message to the perspective nodes who might be interested in the information, it achieves faster convergence and reduce overhead without sacrificing reliability. Secondly, it has been pointed out that nodes connected by a $56Kbps$ modem are unable to handle more than 20 queries per second, corresponding to a network of about 1000 nodes [1]. If these nodes fail, the network may become fragmented. Therefore, the control overhead has to be distributed unevenly according to nodes' different capacities, reflecting the heterogeneity of the network.

This metric will be used in two places as follows. One is in the time domain, the metric is used to dynamically adjust the length of the gossip round. Each node calculates its so called "energy" or "goodness" periodically, according to the defined metric. B_i is its "download bandwidth", and it is the smaller one chosen between the multicast rate and its residual download bandwidth. For example, a node with a download speed of $512Kbps$ joins a multicast session running at $2Mbps$, the "download bandwidth" is therefore $min(512K, 2M) = 512Kbps$. B_j is its residual upload bandwidth. D is the network distance from the sender to this node itself. A node with a higher "energy" can be interpreted as having high bandwidth and/or low delay, and can potentially offer better service to more nodes. It means that potentially more nodes will be interested in the information about this node. Therefore, it should have a shorter time interval between consecutive gossip rounds. At the same time, desynchronizing the length of the gossip rounds among all the participating nodes avoids the "synchronization" problem as well. The other use of the metric is in the space domain, adjusting the scope of the gossip targets selection. Upon receipt of the gossiping message, each node picks up some nodes from its *OutView* to disseminate the message. The metric will act as an index to direct the message to the nodes for which this message might be useful. In Eq.1 C is the "children" account, the more nodes it is serving now, the more responsibility it should take, i.e., it needs to have more information to cope with the potential failure; B_i is the residual upload bandwidth of the neighboring node; B_j is the residual download bandwidth of that node. D is the delay between itself and the neighboring node. It indicates the desirability of the potential path. As stated before, the gossiping message has different meaning for different nodes. For example, when an event is multicasted at $512Kbps$, a node with a download bandwidth of $1Mbps$ may not be interested to know the information about another node with an available upload bandwidth of just $56Kbps$. The idea is to direct the gossip message to those nodes with lower delay, higher bandwidth and higher children count. It has been shown once the highly connected nodes, which are called hubs, have received the message, the message will spread over the entire network at a very high speed [9].

Two assumptions are considered in the protocol design. Firstly, certain information about one or several on-tree nodes will be provided, by some out-of-band mechanism like bootstrap, to the newcomers. Secondly, each node has a rough idea of what is the current multicast rate and the magnitude of the delay between the sender and itself. These assumptions are also used in other protocols, and delay can be measured actively by pinging, or passively by checking the timestamps of the received packets.

3.2 Protocol details

The proposed protocol consists of three building blocks:

1. **Join**: The newcomer sends a **JOIN** message containing its own quality of service (QoS) demands (e.g., bandwidth and/or delay) to the target node (chosen based on the bootstrap assumption). If this joining process succeeds, the new comer will begin to receive data, and a confirm message (**ACK**) will be sent back as well, either piggybacked or sent as an independent packet. All the intermediate nodes will update the entry associated with the contacted node. If this joining process fails, a reply packet will be sent back as well, all the intermediate nodes will check this packet and decide whether to update or not (depending on whether the delay and residual bandwidth have changed or not). If necessary, some data packets will be sent back as well, to reduce the joining latency. Normally, the joining nodes will be given several nodes to contact by the bootstrap algorithm; it selects the best one among the nodes who respond its **JOIN** request.

2. **Gossip**: Each node periodically calculates its current "energy" and adjusts the gossip interval correspondingly. With an initial gossip interval t, the next gossip

Algorithm 1 Gossip Algorithm

```
1: if $it\ is\ time\ to\ gossip$ then
2:     for $every\ node\ n_i \in OutView$ do
3:         $p_i = \frac{E_i}{\sum_i E_i}$
4:         $Send(n_i, p_i)$
5:     end for
6:     $t^{'} = t\frac{\frac{\sum E}{N}}{E}$
7:     adjust the gossip interval
8: end if
```

Algorithm 2 Gossip Forward

```
1: if $receipt\ the\ gossip$ then
2:     update its membership list
3:     for $every\ node\ n_i \in OutView$ do
4:         $p_i = \frac{E_i}{\sum_i E_i}$
5:         $Send(n_i, p_i)$
6:     end for
7: end if
```

interval is calculated using a normalization $t^{'} = t\frac{\frac{\sum E}{N}}{E}$. In which E represents the calculated "energy", and it is normalized over N rounds. When next gossip round comes, it sends a gossip message containing its current QoS attributes to some nodes chosen pseudo-randomly according to the probability as $p = \frac{E}{\sum E}$, in which "energy" is normalized over all the members in the list. Upon receiving the message, the receiving nodes will update its neighboring list if necessary and propagate the message to some nodes chosen based on the metric as well. A hop count is associated with the gossiping message, representing how many nodes have saved and spread this message. On one hand, it is important to make sure this message has been processed by several nodes already. On the other hand, it can be used as a last resort for scope control to reduce the amount of gossiping messages. The algorithm is summarized in Algorithm 1 and 2.

3. **Cope with failure**: By sending and receiving these gossip messages, each node updates its neighboring list. When random failure happens, the affected nodes can select the most suitable parent-to-be from its updated neighboring list, and resume the service quickly without huge service disruption.

The idea is to reduce the membership management related overhead by adjusting the gossip interval and the scope of gossip targets selection dynamically.

4 Simulation Results

Simulation was conducted by using OMNet++, and we also compared our algorithm with SCAMP [4].

In terms of network dynamics, peer-to-peer networks are quite different from the traditional network, since most of the nodes are expected to get data from others, instead of from the data source directly. Hence, the nodes are not really independent of each other. Behavior of some nodes will affect that of others, such as the nodes who are taking the responsibility to forward data. The definition of network dynamics have to be redefined, taking the properties of peer-to-peer network into account. Half-life, which is the time taken by the system to change half of the nodes, has been raised to solve this issue [7].

Because data forwarding job is taken by nodes, so the average hop count and the delay of the data packets were measured. As can be seen from Figure 2 the average hop count and delay increase with half-life time. In other words, they decrease with the network dynamics. This can be explained as when the network are more stable, more nodes will take the responsibility to replicate and relay the data, consequently, more nodes will get data locally rather than from the source directly. On the other hand, when the network is highly dynamic, it is difficult for the nodes to get data locally, and they have no choice, so they are forced to turn to the source directly. It is noticeable that when the average hop counter equals one, i.e., all or most of the nodes will contact the data source directly, the corresponding half-life time is around 60 seconds, that represents a very highly dynamic network, in reality, we would expect a network has more stable behavior.

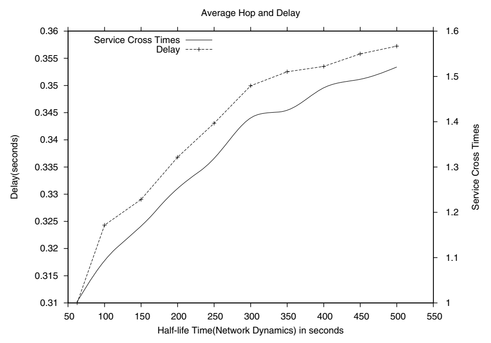

Figure 2. Average hop and delay.

Figure 3 shows the link stress (the number of duplicate packets traversing the same physical link) for data packets. It is necessary to point out that the curve labeled as $tier1$, $tier2$ and $tier3$ routers represents core network, local region network and local access network respectively. The trend is very clear, link stress for local access networks remains relatively constant. This can be interpreted as the

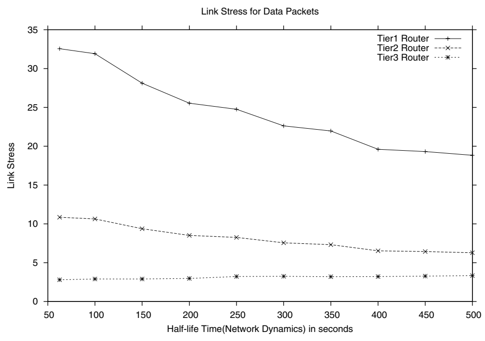

Figure 3. Link stress of data packets.

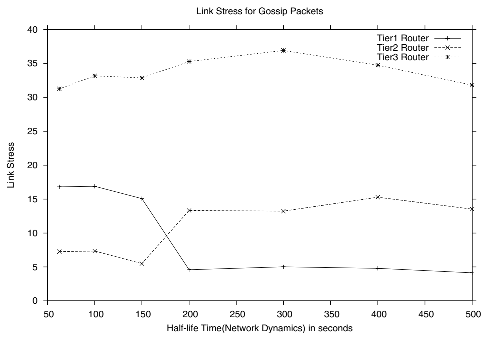

Figure 4. Link stress of gossip overhead.

stress is in proportion to the number of nodes participating the session, so it is relatively constant. Link stress of local region network decreases slightly when the network becomes more stable, compared with the link stress of core networks, which has decreased approximately 44%. The explanation for this phenomenon is that when the network becomes more stable, more nodes will get data locally instead of from the data source, thus some of the data packets will bypass the core network.

Figure 4 displays the link stress of the gossip messages. We investigate the link stress of core, local region and local access networks respectively. The link stress of local access networks remains roughly the same except for some small fluctuations, and this can be viewed as the fluctuation of the number of participating nodes. It is interesting to see there is a crossover point corresponding to a half-life time of around 180 seconds. At first, when the network is highly dynamic, the link stress for local region networks is twice that of the local access network. This indicates that when the network is not very stable, in order to exchange information, the gossip messages have to propagate through the local region network. When the network becomes more stable, the link stress of local access network outgrows that of local region core network after the crossover point. This can be interpreted as the nodes only changing information locally in a more stable network. That means our algorithm is adaptive, and it can confine the gossip messages overhead to local regions, reducing the load of the core networks, and scalability is achieved.

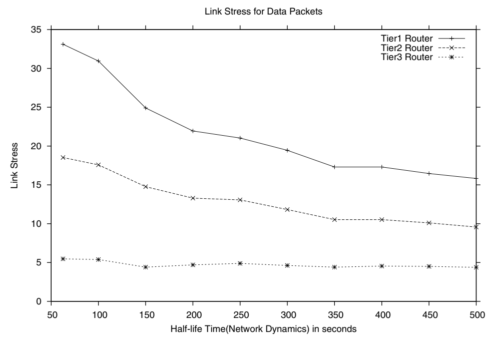

Figure 5. SCAMP: Link stress of data packets.

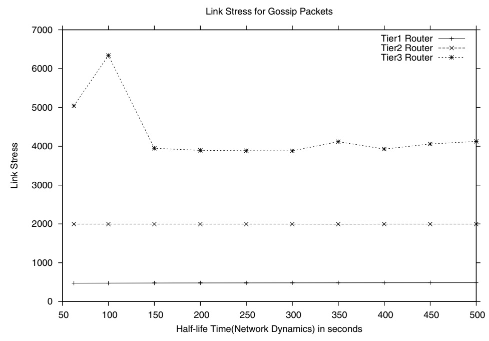

Figure 6. SCAMP: Link stress of gossip overhead.

When it comes to the link stress for data packets, Figure 5 shows the same pattern with that of our algorithm, except for the slight differences in $tier2$ and $tier3$ routers. This can be explained as our algorithm can make use of the hierarchical structure and discover the local resource more efficiently. Hence the nodes can get data locally instead of from the nodes which are far away, preserving the backbone bandwidth. However, there is a big difference in the link stress for gossiping message overhead. Compare Figure 4 with Figure 6, it can be concluded that our algorithm can use the network resource more efficiently. When the network becomes more stable, all the nodes can capture this change and exchange information locally. However, in the case of SCAMP, this change can not be sensed by the nodes, and consequently, the communication overhead remains roughly constant, in proportion to the number of nodes presenting in the network .

5 Related Work

The emergence of P2P applications stimulated many decentralized protocols targeting P2P routing and object locating [13]. They build the multicast tree or the overlay mesh decentralized. But they all relied on a P2P routing overlay to work properly, and messages were even flooded within the mesh [11]. Reliability was achieved reactively by retransmission or tree repairing, rendering high overhead and poor reliability. On the other end of the design spectrum, gossip-based algorithms combine simplicity and reliability[3]. In these algorithms, each node forwards the messages it received to other randomly chosen nodes, and the decision is made independently. These randomized or pseudo-randomized methods offer redundancy proactively, ensuring reliability even in the face of random network failures. Nevertheless, the simplicity comes at the price of more traffic on the network and these algorithms relied on a non-scalable group membership algorithm.

Directional gossip [8] was proposed to reduce the communication overhead by deliberately choosing nodes with low connectivity to communicate with. However, it differs with our algorithm in the sense that our algorithm works in control plane and handles membership management only, i.e., it is decoupled from data delivery. The size of a normal membership management message is in the order of several hundred kilo bytes, compared with a size of several hundred mega bytes of data packets. Our algorithm achieves a significant reduction in communication overhead.

Several gossip-based membership management algorithms have been proposed, and the most similar one is SCAMP [4]. Scalability is achieved by letting each node have only partial view of the network, and nodes periodically communicate with the nodes chosen at random from this partial, but it lacks flexibility and adaptability. It operates using fixed parameter settings, without taking the changes of the network into account. Our contribution is a simple, adaptive gossip-based membership management algorithm, taking the network heterogeneity (bandwidth and network delay) into account. It enhances the ALM reliability without sacrificing the scalability.

6 Conclusion

An adaptive gossip-based application layer multicast membership management algorithm is proposed. It adapts to changes in the network, confining the communication overhead mainly within local regions. Therefore, it can use the network resource more efficiently. Furthermore, it builds the tree according to the link capacity of the node, i.e., let the node with lower capacity attach to the node with higher capacity, maximizing the capacity of the application layer multicast or peer-to-peer network.

Reliability comes with the redundance of the randomized algorithm, and nodes may have some proactive measures for failures. Scalability relies on choosing gossip targets preferentially; a faster converge and less overhead will be achieved. To conclude, this adaptive gossip-based membership management algorithm is simple and efficient.

Acknowledgement

This project is funded by the ARC linkage grant under contract No. LP0347217. Many thanks to Mr Graeme White to read the draft version of this paper.

References

[1] Clip2. Gnutella: To the bandwidth barrier and beyond., 2000.

[2] S. E. Deering and D. R. Cheriton. Multicast routing in datagram internetworks and extended lans. *ACM Transactions on Computer Systems (TOCS)*, 8(2):85–110, May 1990.

[3] A. Demers, D. Greene, C. Houser, W. Irish, and J. Larson. Epidemic algorithms for replicated database maintenance. *ACM SIGOPS Operating Systems Review*, 22(1):8–32, 1998.

[4] A. Ganesh, A.-M. Kermarrec, and L. Massouli. Peer-to-peer membership management for gossip-based protocols. *IEEE Transactions on Computers*, 52(2), 2003.

[5] B. Haberman and J. Martin. Igmpv3 and multicast routing protocol interaction. In *IETF Internet Draft*, July 2001.

[6] Y. hua Chu, S. Rao, S. Seshan, and H. Zhang. A case for end system multicast. *IEEE Journal on Selected Areas in Communication (JSAC), Special Issue on Networking Support for Multicast*, 20(8):1456–1471, Oct. 2002.

[7] D. Liben-Nowell, H. Balakrishnan, and D. Karger. Analysis of the evolution of peer-to-peer systems. In *ACM Conf. on Principles of Distributed Computing (PODC)*, Monterey, CA, July 2002.

[8] M.-J. Lin and K. Marzullo. Directional gossip: Gossip in a wide area network. In *Proceedings of the Third European Dependable Computing Conference on Dependable Computing*, pages 364–379, 1999.

[9] R. P.-S. M. Barthelemy, A. Barrat and A. Vespignani. Velocity and hierarchical spread of epidemic outbreaks in scale-free networks. *Physical Review Letters*, 92:1–4, 2004.

[10] D. F.-V. J. C. L. S. Deering, D. Estrin and L. Wei. An architecture for wide-area multicast-routing. In *ACM SIGCOMM*, pages 126–135, Oct. 1994.

[11] S.Ratnasamy, P.Francis, M.Handley, R.Karp, and S.Shenker. A scalable content-addressable network. In *Proc. of ACM Sigcomm*, Aug. 2001.

[12] K. Sripanidkulchai, A. Ganjam, B. Maggs, and H. Zhang. The feasibility of supporting large-scale live streaming applications with dynamic application end-points. In *Proceedings of ACM SIGCOMM*, Portland,OR,USA, Aug. 2004.

[13] B. Zhang, S. Jamin, and L. Zhang. Host multicast: A framework for delivering multicast to end users. In *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies*, June 2002.