Cluster Based Approaches for End-to-End Complete Feedback Collection in Multicast

Mehmet Baysan
Computer Science Department
University of Texas at Dallas
Richardson, Texas 75083

Kamil Sarac
Computer Science Department
University of Texas at Dallas
Richardson, Texas 75083

Abstract

In this paper we study the end-to-end complete feedback collection (ECFC) problem in large scale multicast applications. We consider the case where each receiver is expected to send feedback in a timely manner without causing implosion at the source site. To address the scalability problem and improve timely feedback collection, we introduce the use of clustering algorithms for feedback collection. Our simulation based comparisons show that the clustering based approaches outperform the existing pure (without clustering) multi-round probabilistic and pure (without clustering) delayed feedback collection approaches both in terms of collection delay and message overhead.

1 Introduction

IP multicast [1, 2] provides a scalable and efficient mechanism to support large scale one-to-many networking applications such as Internet radio and Internet TV. Even though *inter-domain* deployment of the multicast service has been slow, multicast is widely deployed and actively used in *intra-domain* environments. Today, a large number of streaming multimedia players, including those by Microsoft and Real Networks, as well as data delivery tools, including solutions from companies like Digital Fountain and Multicast Technologies Inc., make use of multicast. Recently, there have been numerous articles in newspapers and trade magazines about the use of multicast to deliver popular content. For example, the British Broadcasting Company (BBC) announced recently that Olympic events would be delivered to home users via multicast technology.

It has been observed that many-to-one message transfer in the form of receiver feedback is a practical and frequently needed operation in various contexts related to multicast [3, 4]. One of the most important issues in collecting receiver feedback is to provide timely feedback without causing implosion at the source site. One widely studied context for receiver feedback is reliable multicast. In reliable multicast, source depends on receiver feedback to provide reliability via retransmissions. In general, the source needs to receive only one single feedback message to learn about a packet loss event. Hence, most of the work in many-to-one feedback collection have focused on developing mechanisms to reduce the number of receiver feedback going back to the source [5, 6, 7, 8]. Several researchers have also developed mechanisms to estimate the group size so as to use this information to control the load generated by receiver feedback at the source site [9, 10, 11].

In this paper, we consider *end-to-end complete feedback collection (ECFC)* problem, i.e., the problem of providing end-to-end feedback from all the receivers in a multicast group to the source in a timely manner without creating implosion at the source site. We consider a very general case of receiver feedback and do not make any assumptions on the nature of the feedback. Therefore, we expect that the techniques presented in this paper can be used by a large set of applications including counting the number of receivers; collecting monitoring and management related feedback; collecting votes or results of receiver polls from the receivers; etc. In this context, since we consider a generic feedback collection case, we do not use support from the network and mainly focus on end-to-end feedback collection.

Due to its many-to-one nature, ECFC has inherent scalability problems. In our previous work [12], we developed approaches (a probabilistic multi-round and a delayed feedback collection approach) to achieve ECFC for large scale multicast applications. We briefly discuss these approaches in Section 3.

In this paper, we propose the use of clustering algorithms to significantly improve the scalability and timeliness of end-to-end feedback collection for large scale multicast applications. We consider three clus-

tering algorithms two of which (presented in [13] and [14]) were developed for forming clusters in mobile ad hoc networks (MANETs). The third algorithm is an improvement that we propose on the algorithm presented in [14]. The main contribution of this work is to demonstrate the use of clustering algorithms for solving the ECFC problem. Instead of developing new algorithms, we adopt the above mentioned algorithms to study the feasibility of using cluster-based approaches in ECFC context. In addition, we perform simulation based evaluations to compare the performance of these approaches with the previously known multi-round probabilistic and delayed approaches.

The rest of the paper is organized as follows. The next section summarizes the previous work. Section 3 presents the cluster based approaches for ECFC. Section 4 includes our simulation based evaluations of alternative approaches for ECFC. Section 5 provides a brief discussion on additional clustering issues. Finally, Section 6 concludes the paper.

2 Related Work

The related work can be divided into two groups: (1) minimum feedback collection and (2) complete feedback collection. In minimum feedback collection, researchers developed various approaches [5, 6, 7, 8, 15, 16] to reduce the number of simultaneous feedback messages (i.e., feedback implosion level) reaching the feedback collection site. The techniques developed in this direction can be used to improve feedback scalability in reliable multicast applications. The main difference between our work and the above approaches is that in our work we are interested in collecting feedback from all the receivers in the group whereas the studies cited above aim at reducing the number of feedback messages to a minimum level.

In complete feedback collection, the goal is to scalably collect feedback from all/most of the receivers in the group. The work in this group is also divided into two areas. In the first area, support from intermediate network entities, i.e. routers, is used to aggregate receiver feedback [17, 18]. These approaches require modifications to routers which affect their practicality. In the second area, feedback is collected end-to-end without involving any routers in the process [12]. The work on end-to-end complete feedback collection is the most relevant work to our cluster-based feedback collection study and therefore deserves further discussion as presented below.

Existing approaches for scalable end-to-end complete feedback collection include a multi-round probabilistic approach and a delayed approach. In the multi-round probabilistic approach, the source uses multiple rounds to collect feedback information from the receivers. In the first round, the source uses an initial blind guess about the number of receivers in the group and computes a response probability. Then, the source sends a request message to the multicast group asking the receivers to send their feedback with this probability. Depending on the specified response probability, a number of receivers send their responses back to the source via unicast. On receiving the responses, the source modifies its guess about the number of receivers and uses this new guess to adjust its response probability. Next, it sends a new request with this new probability to the receivers via multicast. On receiving the new request message, only the receivers that have not sent their responses in the previous round(s) participate in the process. This way, in each round, the source collects a manageable number of responses and also improves its guess about the number of receivers on the tree. Figure 1 presents the source's estimation of the number of receivers in the multicast group.

$K \leftarrow$ desired num. responses
$X_i \leftarrow$ num. returned responses in a round
$\bar{R}_i \leftarrow$ manager's estimation of num. of receivers
$P_i \leftarrow$ reporting probability

$$\bar{R}_i \leftarrow \frac{X_{i-1}}{P_{i-1}} + \sum_{j=0}^{i-2} x_j$$
$$P_i \leftarrow \frac{K}{R_i - \sum_{j=0}^{i-1} x_j}$$

Figure 1: Manager's estimation of num. receivers in multi-round approach.

In the delayed approach, the source first runs a round to get an estimation about the number of the receivers in the group as described above. Based on this initial guess and the desired implosion level (i.e., maximum number of response messages that the source is willing to receive in a round), the manager computes a maximum delay interval and sends its request with this information. Each receiver, upon receiving the request, first chooses a random delay value out of the given maximum delay interval and sends its feedback at the end of this delay period. The main goal in using a random delay period is to disperse the receiver feedback and reduce the possibility of having an implosion problem at the source site.

One issue with the delayed approach is that the source may not know the number of feedback messages to be received and depends on the outcome of one simple experiment to estimate it. Therefore, the given maximum delay interval calculation may not be

accurate all the time. An overestimation of the value may result in unnecessary delays in collecting the feedback and an underestimation may result in implosion.

In addition to scalability, the above approaches have different behaviors when packets (especially request packets) are lost. In the case of the delayed approach, if the request packet is lost before reaching a fraction of the receivers, this will cause feedback information loss for the source. On the other hand, due to the nature of the probabilistic multi-round approach, the source will have a better chance of collecting the missing information in additional rounds.

3 Cluster-based Approaches for ECFC

In this section, we present two algorithms that were developed to form clusters in MANETs and show how they can be used in receiver clustering for multicast feedback collection. We also present a third algorithm which improves on one of the algorithms.

In the below algorithms, clusters are formed based on proximity (i.e., hop count). We use IP TTL field value to define neighborhood range among the receivers. All receivers within a given hop count of a receiver are considered to be neighbors. Based on the given TTL value, receivers identify their neighbors and form their clusters as presented below.

3.1 A Simple Clustering Algorithm

The first algorithm is designed by Ephremides, Wieselthier, and Baker [13] for building clusters in a MANET environment. The algorithm assumes that the collection site, which is usually assumed to be the source sending data to the multicast group, knows identity/address of all group receivers. At the beginning of the algorithm, all the receivers are considered as unassigned receivers. At each round, source chooses one receiver as a cluster head and asks the receiver to build its cluster with a given neighborhood radius, i.e., hop count. At the end of the round, the cluster head informs the source about the identities of its cluster members. Then, the source removes these receivers from the unassigned receivers set; chooses another receiver from the remaining unassigned receivers set as a cluster head; and asks it to form a cluster. The procedure continues until the unassigned receiver set becomes empty.

The main characteristics of this algorithm are that (1) the source is assumed to know identities/addresses of all the receivers in the group and (2) only one cluster can be formed in each round. From ECFC operation point of view, the first characteristic deteriorates the utility of the approach. This is because, in general, multicast hides the identities of the receivers within the group and sources may not necessarily know the identities of all the receivers in the group. In addition, according to the second characteristics, the algorithm may potentially take a long time to form the clusters. If the receiver behavior is relatively dynamic, it requires running the clustering algorithm before each and every feedback collection event and this will potentially introduce significant delay in feedback collection for the application.

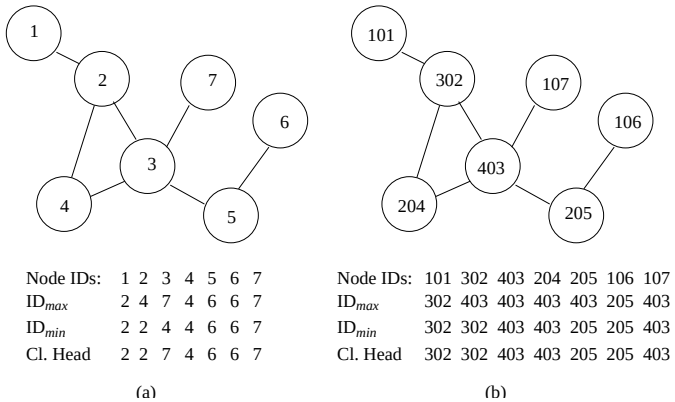
3.2 Two-Round Clustering Algorithm

The second algorithm, by Amis, Prakash, Vuong, and Huynh [14], is a distributed algorithm which can find a clustering in two rounds only. This algorithm requires that each node in the network has a unique ID and in our case we use IP addresses as node IDs. In each round, every node sends a local multicast message to its neighbors by a pre-defined TTL value that defines the neighborhood range. At the end of the second round each node knows if it is a cluster head or the identity of its cluster head. In the first round, each receiver broadcasts its own ID (ID_{self}) to its neighbors. At the end of this round, each receiver stores the largest ID (ID_{max}) it has heard in the first round. In the second round, each receiver broadcasts ID_{max} value to its neighbors. At the end of the second round, each receiver stores the minimum ID (ID_{min}) it has heard from its neighbors during the second round. Then, each receiver independently identifies its cluster head as follows: (1) if ID_{min}=ID_{self}, the node is a cluster-head, otherwise (2) if ID_{max}=ID_{min}, then it chooses this node as its cluster head, otherwise (3) it chooses the node with ID_{max} as its cluster-head.

Figure 2-a presents an example scenario that uses two-round clustering algorithm. The presented graph corresponds to a neighborhood graph among the receivers and the links between the nodes indicate that the nodes are neighbors of each other. For simplicity, we use small integers to represent node IDs. The second row presents the ID_{max} values identified by each node at the end of the first round. Similarly, the third row presents the ID_{min} values identified by each node at the end of the second round. Finally, the last row presents the IDs of the cluster heads chosen by each node in the figure. According to this row, we have a total of four clusters in the network. A more detailed explanation of the algorithm including a proof of correctness is given in [14].

3.3 Optimized Clustering Algorithm

The third algorithm is an improvement/optimization on the second one. In this algorithm, we modify the above algorithm to reduce the number of clusters to be formed at the end of the procedure. For this, instead of the IP addresses, we



Node IDs:	1	2	3	4	5	6	7
ID_{max}	2	4	7	4	6	6	7
ID_{min}	2	2	4	4	6	6	7
Cl. Head	2	2	7	4	6	6	7

(a)

Node IDs:	101	302	403	204	205	106	107
ID_{max}	302	403	403	403	403	205	403
ID_{min}	302	302	403	403	205	205	403
Cl. Head	302	302	403	403	205	205	403

(b)

Figure 2: Examples for (a) 2-round clustering and (b) optimized clustering.

use the number of neighbors of each node as its node ID. In the algorithm, nodes with large number of neighbors are selected as cluster heads. The intuition here is that if we choose the receivers with large number of neighbors as cluster heads, since their clusters will include large number of members, at the end, the number of clusters will likely be small. In cases where there are multiple candidates for the cluster head position (i.e., multiple nodes with the same number of neighbors), we choose the one with the largest IP address as the cluster head. Finally, the overhead of this optimization is that it requires one more round to find the number of neighbors for each node in the group. Figure 2-b presents an example scenario that uses the optimized clustering algorithm. In this example we use a slightly different approach for node ID assignment where we multiply number of neighbors with 100 and then add the host IDs from the left figure to generate new host IDs for the nodes. The first round in this approach is used to find out the number of neighbors. Please note that the neighbor relation is dependent on the given TTL scope and finding out the number of neighbors requires one round. The second and the third rounds are similar to the first and the second rounds of the two-round algorithm. Finally, according to the last row, we see that this approach returns three clusters in the network.

At the end of the clustering step, one of the receivers in each cluster will be identified as the cluster head and this receiver will be responsible for collecting and bundling the feedback information within its cluster and sending it to the collection site. Since we do not specify the type of the feedback to be collected, we expect that the receivers are equipped with the necessary knowledge/functionality to bundle the collected feedback information within the cluster.

On receiving feedback request messages, receivers in each cluster will forward their feedback to their respective cluster heads and the cluster heads will then do the necessary processing on the feedback and will send the overall feedback information back to the collection site. This way, we will improve the scalability and timeliness of ECFC operation. In the next section, we combine these clustering approaches with multi-round probabilistic and delayed approaches to achieve further scalability in feedback collection.

4 Evaluations

In this section, we present our simulation based evaluations in two parts. In the first part, we will compare the three presented clustering algorithms with respect to (1) number of clusters they form and (2) clustering time. In the second part, we compare different ECFC algorithms with each other. In this part, we will combine clustering algorithms with multi-round probabilistic and delayed approaches to build several hybrid ECFC algorithms. We will also consider pure multi-round probabilistic and pure delayed approaches in our comparisons. The comparison metrics that we use in this part include (1) message overhead and (2) feedback collection time.

In our simulations we use two different network topologies. The first one is a realistic multicast tree topology collected by Chalmers and Almeroth in their mwalk [19] study. The advantage of this data set is that the tree topology is collected from the Internet and therefore results in realistic simulation scenarios. On the other hand, due to its tree structure, during the clustering step, the shortest paths between neighbors will be computed over the tree topology. In practice, this represents a scenario where we have a multicast routing algorithm that uses a shared distribution tree such as Bi-Directional Protocol Independent Multicast [20] (Bi-Dir PIM) or Core Based Tree [21] (CBT).

Due to the lack of realistic router-level large scale network topology information, the second topology is a synthetic transit-stub network topology that we generated by using the Georgia Tech Internet Topology Modeler (gt-itm) [22] tool. The advantage of using network topologies over tree topologies is that with network topologies the shortest path computation between the neighbors will be based on the network layer connectivity. In practice, this represents a scenario where we are allowed to use network layer broadcast primitives during the formation of the clusters among close by receivers.

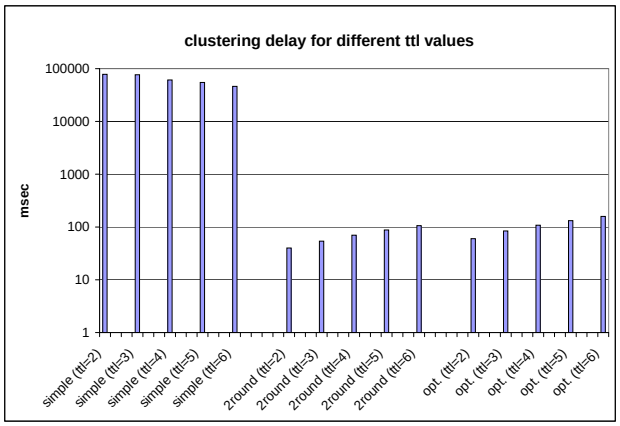


Figure 4: Clustering delay.

4.1 Comparison of Clustering Techniques

In this section, we compare the clustering techniques based on the number of clusters that they produce and the clustering delay. Desirable properties of clustering algorithms are that they produce small number of clusters and they take a short time to finish their clustering.

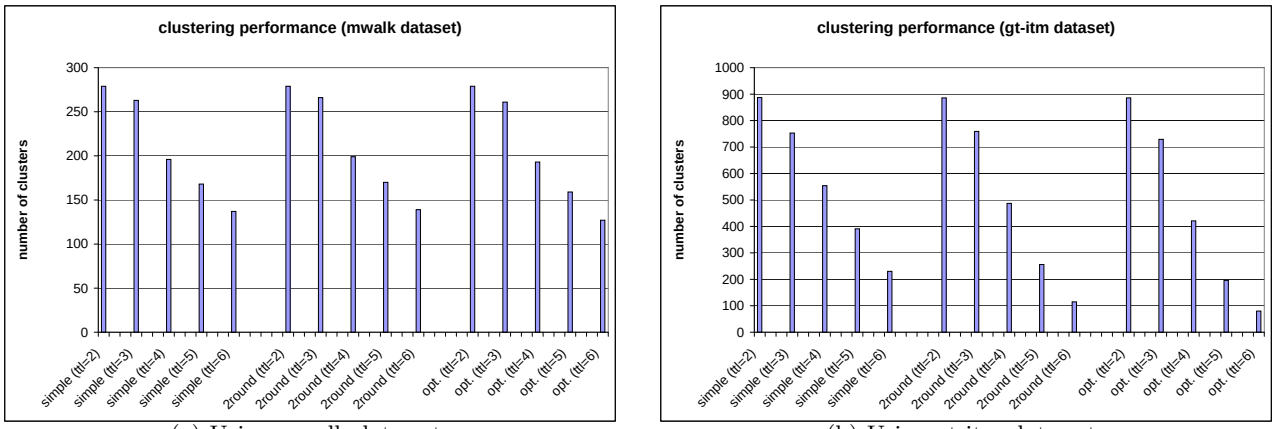
Figure 3 compares the clustering algorithms based on the number of clusters that they produce using neighbor radiuses ranging from TTL=2 to TTL=6. In Figure 3-a, we use a 2871 node (of which 1935 corresponds to the receivers) tree topology. In Figure 3-b, we use a transit-stub network topology with 10,000 nodes (with 9,000 receivers and 1,000 internal nodes). According to the figures, as the neighbor radius increases, the number of clusters decrease. In addition, according to Figure 3-a, there is not much difference among the alternative approaches in terms of the number of clusters formed at the end of the process. This result is mainly because of the underlying network topology used in the simulations. That is, due to the tree nature of the underlying topology, shortest paths between the neighboring receivers always go through the tree topology and this pretty much limits the alternative ways of forming the clusters. On the other hand, the results in Figure 3-b presents some visible differences. Here, the shortest paths between neighbors are computed based on the underlying network proximity. Therefore, as we increase the TTL value from 2 to 6, the performance difference among the different clustering algorithms becomes more visible, the optimal clustering algorithm performing increasingly better than the others.

Another important issue in the comparison of clustering algorithms is the delay incurred during the formation of the clusters. Figure 4 presents the delay characteristics of the alternative approaches using a logarithmic scale. In general, the clustering delay increases as we increase the TTL value. That is, as TTL increases, the time required to reach a neighbor also increases. This in turn increases the duration of rounds in clustering and hence the total clustering delay. For the simple clustering case, results are the opposite. This is because the number of rounds needed to finish clustering is not constant. Instead, it is proportional to the number of clusters. As the neighbor radius increases, the number of clusters and therefore the number of rounds decrease. As a result, considering the fact that the simple clustering requires knowledge of the receivers by the source and considering the simulation results on the number of resulting clusters and the clustering delay, we see that the two-round clustering approach and the optimized clustering approach perform better than the simple clustering approach.

4.2 Comparison of Feedback Collection Approaches

In this section, we present our comparisons of alternative feedback collection approaches. In the comparisons, we use several hybrid approaches by combining clustering with probabilistic or delayed approaches for different TTL values. As an example, in Figure 5-a, simpleprob.(TTL=2) refers to an approach which uses a probabilistic approach among a number of cluster heads that are identified by using simple clustering algorithm with a neighbor radius of TTL=2. Note that in all clustering approaches, cluster members send their feedback to their cluster heads and cluster heads then forward these feedback messages to the source based on the mechanism used on top of clustering (i.e., multi-round probabilistic or delayed approach). We also include the existing *pure* (i.e., without any clustering of the receivers) multi-round probabilistic and *pure* delayed approaches.

Message Overhead. First we examine the message overhead of alternative approaches. In the simulations, we use the above mentioned tree topology and we set the implosion level at the source site to 10 messages per round. We consider mainly two types of messages: local messages and global messages. Local messages are the messages exchanged between cluster heads and their cluster members. Since the TTL scope of the local messages is small (2 to 6), in our comparisons we ignore the overhead of the local messages. Global messages, on the other hand, are exchanged between the source and cluster heads (or individual



(a) Using mwalk data set. (b) Using gt-itm data set.

Figure 3: Performance of clustering algorithms.

receivers in the case of pure multi-round probabilistic and pure delayed approaches). Figure 5-a presents the message overhead (global message overhead) of different approaches in a logarithmic scale. Due to the limited use of global multicast messages in the delayed feedback collection, hybrid approaches that use a delayed feedback collection component perform better than the ones that use multi-round probabilistic feedback collection component. In addition, *pure* multi-round probabilistic and *pure* delayed approaches perform worse than their cluster-based counterparts.

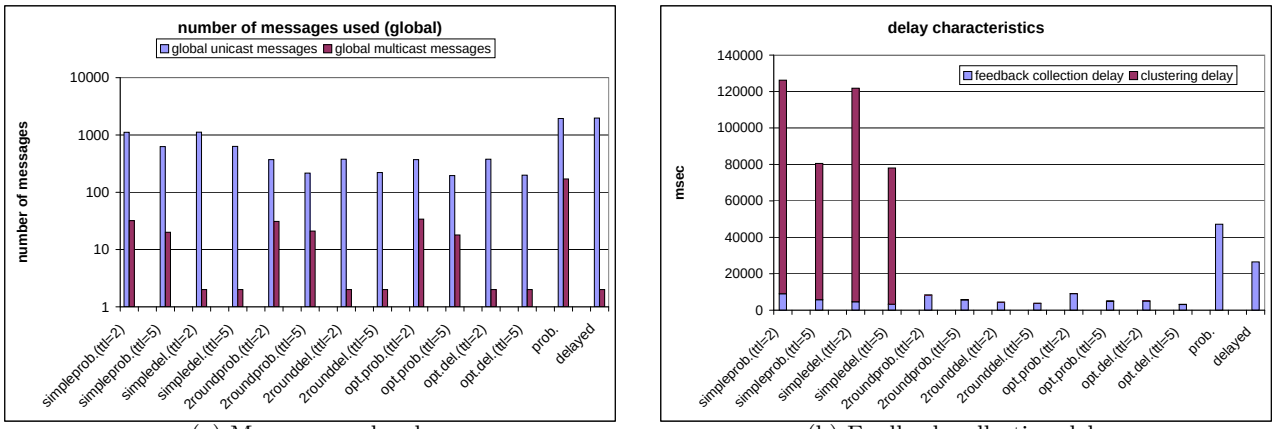
Feedback Collection Delay. In this part, we consider two potential usage scenarios for clustering based approaches: (1) pre-clustering where a clustering is used for multiple feedback process and (2) inline clustering where the source runs clustering at each time it wants to collect feedback from the receivers. The advantage of pre-clustering is that it reduces the feedback collection delay. On the other hand, depending on the receiver dynamics, this approach may affect the implosion characteristics of feedback collection rounds. Figure 5-b presents the performance of different approaches in a logarithmic scale. Here, we explicitly display the clustering delay and feedback collection delay so as to show the performance in pre-clustering and inline clustering approaches together. According to the figure, for the case of two-round and optimal clustering cases, the clustering takes a very short time and therefore are not visible in the figure. These results show that the minimum feedback collection delay case is achieved when delayed method with optimized clustering approach is used. In addition, in most cases the probabilistic approaches take about twice longer to finish. Finally, the figure shows that the clustering algorithms can reduce the delay by more than five times compared to the *pure* multi-round probabilistic and *pure* delayed approaches.

5 Discussion

In practice, after forming clusters within an initial neighborhood range (i.e., TTL hop count), the source learns the number of clusters as well as the total number of receivers in the group. Then, depending on this information, the source can change the TTL value to increase or decrease the number of clusters according to desired implosion rate at the source site.

Another issue in improving the scalability of feedback collection is to use a hierarchical clustering. After the construction of the first level clusters, each cluster head can participate in a second level cluster formation. Since each cluster head in the first level is also a receiver, the same clustering algorithms can be used with larger TTL values to increase the neighborhood range. In hierarchical clustering case, each cluster member sends its feedback to its cluster head at each layer of the clustering and the cluster heads at the highest layer send the collected feedback to the multicast source. Compared to single layer clustering, hierarchical clustering based feedback collection reduces the implosion level at the multicast source site significantly. This improvement is gained by paying an overhead in the form of an increase in feedback collection delay.

Finally, our clustering algorithms depend on the TTL based hop counts in forming the clusters. The number of members in different clusters may be dif-



(a) Message overhead.

(b) Feedback collection delay.

Figure 5: Message overhead and feedback collection delay.

ferent from each other. For load balancing purposes, it may be desirable to form clusters with similar number of members. One approach to achieve this may be to use different TTL based hop counts to define neighborhood ranges for different parts of the multicast tree. In dense regions, we may use smaller TTL values to reduce the number of cluster members within a cluster. In sparse regions, we may use larger TTL values to have large enough clusters. Clustering algorithms that are used in this work assume fixed TTL hop count values. Currently, we are working on modifying/extending these algorithms to support clustering with multiple TTL hop count values for different parts of the multicast trees.

6 Conclusions

In this paper, we focused the end-to-end complete feedback collection (ECFC) problem in large scale multicast applications. The main contribution of this paper is to introduce the use of clustering algorithms for ECFC. Instead of developing new clustering algorithms, we adopted the ones used in mobile ad hoc network environments and shown that the use of clustering can significantly help reduce the delay and the overhead in end-to-end feedback collection. According to our simulation based comparisons ECFC mechanisms that use two-round or optimized clustering algorithms achieve significant improvements in reducing the overhead and delay in collecting receiver feedback. From a practicality point of view, the availability of Bi-Directional PIM [20] (Bi-Dir PIM) makes it possible to use these clustering algorithms and form the clusters. As a result, this study presents a practical solution to end-to-end complete feedback collection operation which is a frequently needed operation in various types of multicast applications.

References

[1] S. Deering and D. Cheriton, "Multicast routing in datagram internetworks and extended LANs," *ACM Transactions on Computer Systems*, pp. 85–111, May 1990.

[2] K. Almeroth and M. Ammar, "Multicast group behavior in the Internet's multicast backbone (MBone)," *IEEE Communications*, vol. 35, no. 6, pp. 224–229, June 1997.

[3] H. Schulzrinne, S. Casner, R. Frederick, and Jacobson V., "RTP: A transport protocol for real-time applications," Internet Engineering Task Force (IETF), RFC 1889, January 1996.

[4] K. Obraczka, "Multicast transport mechanisms: A survey and taxonomy," *IEEE Communications*, vol. 36, no. 1, January 1998.

[5] J-C. Bolot, T. Turletti, and I. Wakeman, "Scalable feedback control for multicast video distribution in the internet," in *ACM Sigcomm*, London, ENGLAND, September 1994, pp. 58–67.

[6] J. Nonnenmacher and E.W. Biersack, "Scalable feedback for large groups," *IEEE/ACM Transactions on Networking*, vol. 7, no. 3, pp. 375–386, June 1999.

[7] T. Friedman and D. Towsley, "Multicast session membership size estimation," in *IEEE INFOCOM*, New York, NY, USA, March 1999.

[8] C. Liu and J. Nonnenmacher, "Broadcast audience estimation," in *Proceedings of IEEE INFOCOM*, Tel Aviv, Israel, March 2000.

[9] S. Alouf, E. Altman, C. Barakat, and P. Nain, "Optimal estimation of multicast membership," *IEEE Transactions on Signal Processing, Special Issue on Networking*, vol. 51, no. 8, pp. 2165–2176, August 2003.

[10] S. Alouf, E. Altman, C. Barakat, and P. Nain, "Estimating membership in a multicast session," in *SIGMETRICS*, San Diego, CA, USA, June 2003.

[11] D. Dolev, O. Mokryn, and Y. Shavitt, "On multicast trees: Structure and size estimation," in *IEEE INFOCOM*, San Francisco, CA, USA, March 2003.

[12] K. Sarac and K. Almeroth, "A distributed approach for monitoring multicast service availability," *Journal of Network and System Management - Special Issue on Distributed Management*, vol. 12, no. 3, pp. 327–348, September 2004.

[13] A. Ephremides, J.E. Wieselthier, and D.J. Baker, "A design concept for reliable mobile radio networks with frequency hopping signaling," *Proceedings of the IEEE*, vol. 75, no. 1, pp. 56–73, January 1987.

[14] A.D. Amis, R. Prakash, T. Vuong, and D.T. Huynh, "Max-min d-cluster formation in wireless ad hoc networks," in *Proceedings of IEEE INFOCOM*, Tel Aviv, Israel, March 2000.

[15] D. DeLucia and K. Obraczka, "Multicast feedback suppression using representatives," in *INFOCOM*, Kobe, JAPAN, April 1997.

[16] M. Donahoo and S. Ainapure, "Scalable multicast representative member selection," in *IEEE INFOCOM*, Anchorage, AK, USA, March 2001.

[17] J. Walz and B. Levine, "A hierarchical multicast monitoring scheme," in *International Workshop on Networked Group Communication (NGC)*, Palo Alto, CA, USA, November 2000.

[18] F. Filali, H. Asaeda, and W. Dabbous, "Counting the number of members in multicast communication," in *Proceedings of NGC*, Boston, MA, USA, October 2002.

[19] R. Chalmers and K. Almeroth, "On the topology of multicast trees," *IEEE/ACM Transactions on Networking*, vol. 11, no. 1, pp. 153–165, February 2003.

[20] M. Handley, I. Kouvelas, T. Speakman, and L. Vicisano, "Bi-directional protocol independent multicast (bidir-pim)," Internet Engineering Task Force (IETF), draft-ietf-pim-bidir-08.txt, October 2005, work in progress.

[21] T. Ballardie, P. Francis, and J. Crowcroft, "Core based trees (CBT): An architecture for scalable multicast routing," in *ACM Sigcomm*, San Francisco, CA, USA, September 1993, pp. 85–95.

[22] E. Zegura, K. Calvert, and S. Bhattacharjee, "How to model an internetwork," in *INFOCOM*, San Francisco, CA, USA, March 1996.