RIPD – Receiver Initiated Parent Driven Multicast for Grid Environment

R. Abilash Joseph, M. Malleswaran, A. Radhakrishnan

Abstract— Grid applications often need to distribute substantial amount of data from root node to many other machines. This typical communication pattern is termed as multicast. Most of the multicast methods need to maintain an optimized tree structure, based on external network monitoring data. The dependence on network data impacts the adaptability of tree based multicast methods to the dynamically changing heterogeneous nature of the network. In this paper a cluster based multicast algorithm RIPD (Receiver-Initiated Parent-Driven) has been proposed.

Index Terms— Cluster Computing, High Throughput, Multicast, Receiver Initiated, RIPD

1 Introduction

TIMELY data replication is one of the most critical issues in a data-intensive grid computing environment. The main advantage of grid computing is its ability to serve high-performance applications by integrating multiple computers from various geographical locations including single computers to large cluster of computers. This makes the grid environment a highly heterogeneous and dynamically changing environment. When a high-performance computing is to be done in a grid environment, a substantial amount of data should be transferred from the source to all other computers participating in the computation [1, 2].

This communication pattern is *multicast*: initially the root node or the root cluster has the data that needs to be processed by the grid, at the end of the multicast all the nodes in the environment has a copy of the data. Multicast is used to distribute large input data for parallel applications before and or during run [2, 3]. A significant amount of processing time and bandwidth usage is observed while distributing data to the nodes in the grid environment [4]. The need for using grid environment to analyze large data and do high-performance computing is becoming increasingly popular, and needs efficient multicasting.

2 Different Message Formats in Multicast

In a multicast operation the data to be replicated is initially available only with the root node, and at the end of the multicast operation a copy of the original data is available in all the nodes in the environment. The data is transferred through spanning tree (single tree or multiple trees) and random meshes. The message transferred through the mesh or tree contains the whole data or a part of the whole data depending on the size of data.

- *R. Abilash Joseph is currently pursuing ME in Embedded Systems Technologies in Anna University of Technology-Tirunelveli, India. E-mail: abijosh@yahoo.com*
- *M. Malleswaran is currently working as an Assistant Professor in Dept of Electronics and communication in Anna University of Technology-Tirunelveli, India. E-mail:malleshaut@gmail.com*
- *A. Radhakrishnan is currently working as the head of the Department of Computer applications department in Anna University of Technology-Tirunelveli, India. Email: r_k_ngl@yahoo.co.in*

Many different methods are employed for efficient multicasting. The major two types of multicast operation are *sender-initiated* and *receiver-initiated* method. For small sized data, multicast is done by sending the data as single message and the nodes receiving the data forwards it in a *spanning tree*. To multicast a large data, the data is split in to *'n'* number of pieces and is sent as a stream. But the multicast depends on the size of messages sent also, which are as described below as STA, STP and MTP.

2.1 STA (Single Tree, Atomic)

This method is employed when the data to multicast is small in size. This data is forwarded to nodes through a spanning tree, and the nodes receiving the data forwards it across the network.

2.2 STP (Single Tree, Pipelined)

This method is employed when the data to multicast is large in size. For such operation pipelining technique is employed which optimizes the overall performance of the system.

Pipelining is achieved in application level by splitting the whole message to 'n' number of pieces by the source node. The indexed pieces are sent in a pipelined fashion randomly, which is re-assembled by the receiver.

2.3 MTP (Multiple Tree, Pipelined)

The message is sliced in STP, and thus there is a freedom of sending the message through different paths by using multiple spanning trees rather sending them through a single spanning tree. This eventually increases the throughput of the network. In a multiple spanning tree method, a spanning tree can be used to forward a distinct fraction of the whole message.

2.4 Complexities in Implementing STA, STP and MTP

STA is implemented by constructing a spanning tree over the network; the objective is to find a tree that minimizes the overall completion time (*makespan*). However, finding the optimum *spanning tree* is a tedious problem and increases exponentially with the increase of nodes in the network; it turns out to be a *NP-complete* problem for a basic telephone model [6]. In STP and MTP, the message to be sent is large which demands the need for the best *spanning tree*.

The best *spanning tree* is the one which maximizes the average number of pieces sent by the sender every time-unit (*throughput*) of the network. However, maximizing the *throughput* can be viewed as a relaxed problem of minimizing the makespan. The problem of throughput maximization is a *NP-hard* problem [6, 7].

3 METHODS OF MULTICASTING

In multicasting, the root node transmits data to all other nodes in a network. Thus at the end of a multicast operation, there is a copy of the original data in all the nodes. Optimizing a multicast operation means, to minimize the makespan. In our work, multicasting is done for replicating large data sets used in high-performance computing problems, thus optimizing the multicast operation here means; to maximize the throughput. Different methods for maximizing the throughput have been proposed, the methods has been summarized in the following sections as two broad categories. First sender-initiated methods have been summarized, followed by the receiver-initiated methods.

3.1 IP Based Multicast

Multicast is done by employing RTP (Real-time Transfer Protocol) for message transfer and RTCP (Real-time Transfer Control Protocol) to monitor the QoS of RTP. RTP receivers provide feedback of received packets using the RTCP protocol. This QoS information is used by the source to estimate and adjust the transmission along various paths through a single spanning tree. For instance if too many packets are lost in a particular path, the sender would initiate an aggressive communication over that path [8]. However, since the multicast is performed in the IP layer, analyzing the RTCP reports is very difficult to estimate the loss pattern. This can be easily explained by considering the simple network shown below.

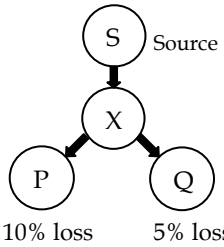

Fig. 1. Simple Multicast Session.

TABLE 1
THE EXTREMES POSSIBLE OF LOSS PATTERNS FOR THE SAMPLE NETWORK IN FIG. 1.

Path	*Extreme1*	*Extreme2*
S,X	No loss	5% loss
X,P	10% loss	5% loss
X,Q	5% loss	No loss

As shown in Table I there are two extreme loss patterns and there are infinite number of patterns between those extremes. This is the problem with simple IP multicasting; it was rectified by M-RTP by employing individual RTP and RTCP sessions for every possible links in the network [9].

3.2 Split Stream Multicast

The previous method does not consider the bandwidth of the network. In this method, instead of using a single spanning tree, multiple spanning trees are implemented. After the spanning trees are constructed, the data is split into many stripes or slices or pieces. A subset of stripes is forwarded through a particular spanning tree. In this way all the stripes are forwarded through different spanning trees.

The sender and receiver both select the number of trees to which they forward and receive from based on the bandwidth of the network. By maintaining a handoff between network bandwidth and data it is taken care that the network doesn't suffer from a local or global bottleneck [10].

3.3 Modeling Multicast based on Network-Performance

Nodes connected to a network through a NIC (network interface card), and their speed of connectivity depends on the connectivity speed of the NIC. For optimizing the multicast operation, the achievable bandwidth should be effectively used. The difference between achievable bandwidth and bandwidth capacity is clearly explained in [11]. The main objective is to maximize the network throughput of all nodes in the network.

The local bandwidth is often limited by the nodes connecting to the network; for example the node may be connected to a gigabit network through a fast Ethernet card, or the access link is shared by many other computers. This is referred as a local bottleneck; if the achievable bandwidth is limited over a wide area network it is termed as a global bottleneck [2].

To optimize the multicast based on network monitoring information; external network monitoring systems like REMOS [12] or Delphoi [13] should be employed exclusively. When multicast is performed based on the network monitoring information, the network monitoring software should be available or accessible to all nodes participating in the multicast. Also the tools exhibit poor performance when used for large network; if a network has N nodes, O(N2) paths should be considered. If the network is dynamically changing; which is very obvious in a grid environment, use of such tools is a not a good option. Also the tools monitor the network properties; but not the properties that are useful for the application, such as achievable bandwidth. Converting the monitored data into application-level details is a hard problem [13].

3.4 Sender-Initiated Multicast

3.4.1 MagPIe

The basic of this approach is to send from the root node to all others simply by ignoring the network information. Then based on the QoS report, resending data will be initiated. In this method multicast is performed as two layers specified by Inter-Cluster and Intra-Cluster graphs. Inter-Cluster graph connects many clusters, and Intra-Cluster graph connects the

nodes within a cluster. To interface between these graphs a coordinator node is employed within clusters [14]. The Intra-Cluster graph puts a high load on the root node, which creates an overall bandwidth bottleneck over the network.

3.4.2 MPICH-G2

MPICH-G2 is an improved version of MagPIe, in which the nodes receiving the data forwards them through a spanning tree, thereby reducing the overhead for the root node. MPICH-G2 uses multilevel topology tree adapt itself to the differences in the achievable bandwidth across the different levels of the network. TCP over WAN acts upon TCP over a LAN and this hierarchy is followed till the leaf nodes. Nodes are grouped into clusters or subtrees based on their ability to communicate with each other relative to a particular level [15].

Finding the optimal spanning tree from a set of possible spanning trees is a difficult task. A standard high-performance spanning tree can be constructed if the network is a homogeneous network. Since the network is heterogeneous and dynamically changing, it is hard to find the optimum spanning tree [6, 7].

3.4.3 FPFR

Finding the optimal solution can be expressed as a linear programming problem. But the number of constraints grows exponentially with the number of hosts. Finding the exact solution is slow and expensive [16]. The multiple spanning tree approach in [5] determines the maximum multicast throughput if the bandwidth of all the links between the hosts is known apriori but uses a complicated algorithm to find the spanning tree that achieves the maximum throughput. The FPFR tool implements multiple spanning trees, and concurrently uses all the trees. These concurrent trees are termed as multicast trees. FPFR uses repeated DFS (depth-first search) to find the tree spanning all hosts. FPFS 'reserves' bottleneck bandwidth of all links used in the tree. The links with no leftover bandwidth cannot serve for new trees. This search is repeated until all spanning trees that spans over all hosts have been found and considered [1]. FPFR doesn't consider the local bandwidth capacity of the hosts into account. Also FPFR has failed to consider oversubscription of links, which forms capacity bottlenecks.

3.4.4 Balanced Multicasting

Balanced Multicasting is an improvement over FPFR which considers the bandwidth capacity of the hosts in to account. The objective is to create a balanced multicast tree. First the individual hosts are considered and then the bandwidth is considered for clusters of computers [17]. However, Balanced Multicasting; like all other spanning tree based multicasting methods, computes the optimized tree based on the network monitoring data when the multicasting is started. The dynamic nature of the network is not considered which leads Balanced Multicasting not adaptive to the changes in the network.

3.5 Receiver-Initiated Multicast

As explained in the previous sender-initiated methods, finding the optimized multicast trees is a hard problem. Even if the spanning tree is carefully computed; it becomes inefficient, as it is not adaptive to the changes in the network. Therefore many different alternatives has been proposed based on receiver-initiated methods, in which the receiver nodes requests for the data it requires as an alternative to the sender forwarding it over trees.

3.5.1 Bullet

Bullet uses an overlay mesh instead of a tree structure. The overlay mesh delivers higher throughput and reliability when compared to the traditional tree based methods. The data is distributed in a strategic manner to strategic points in the network; the receivers locate and retrieve required data from multiple points in parallel. The data is split in to sequential blocks by the sender, which is in turn spitted as individual objects, which are transmitted to different points in the network. Nodes receive a set of data from their parents; it is the responsibility of the node to locate the peers that hold the missing data [18].

3.5.2 BitTorrent

It is a peer-to-peer file transfer application, specially designed to distribute large files efficiently. The data is logically split in to equal sized pieces of few kb (kilo bytes) in size. A mesh is created between the peers chosen randomly, and inform about the pieces they possess. Then the nodes constantly update about the new pieces they receive. The receiver nodes explicitly request for the piece they require from the nodes reporting the possession of those pieces. The peers to be requested are selected at random from their possession report.

The best part of BitTorrent is that; the nodes are allowed to download only if they upload pieces of data. Uploading data acts as an incentive to the peer that gives a higher chance of being allowed to download data [19].

3.5.3 Chainsaw

This algorithm simply works on the same way as the BitTorrent algorithm. It is designed mainly focusing on the live streaming of data over a large number of hosts. Nodes constantly report their window of availability to their neighbors; these are the pieces of data they possess and willing to transfer. By checking with this availability report the receivers request for the data they require.

Since this algorithm is specifically designed for real-time data transfer like live streaming etc., if a data is not found within a specified time period, termed as 'fall off' the data is considered as lost [20].

Faster nodes download the available pieces quickly and search for more pieces among its neighbours; if the pieces are not available or found among the neighbouring nodes, the faster node has to remain idle until the pieces are available in any one of its neighbours. This idle time is cannot be compensated by any means which is a major drawback found in the random mesh based multicast if the same technique is implemented in a high-performance computing environment. To overcome this problem cluster based algorithms has been proposed.

3.6 Clustered-Receiver Initiated Multicast

In this method the receiving nodes are clustered based on some conditions. The cluster of nodes team up as a single unit to download the required data from the source nodes.

3.6.1 MOB

MOB – Multicast Optimizing Bandwidth; is a clustered receiver initiated multicast, in which the receiver nodes are clustered together. Each node in the cluster requests and receives an equal part of the total required data from peers of other clusters and distributes it to the nodes in its own cluster. Thus a single piece of data is transmitted to a cluster only once. The data is distributed to the local peers automatically.

MOB fails to manage the static load balancing in large or dynamically changing heterogeneous networks [21].

3.6.2 Robber

Robber is a collective data stealing technique. The data is distributed over a random mesh and nodes 'steal' pieces of data from other peers. Also the nodes team up as collective to steal data from other clusters. The details of the nodes and the cluster they belong to are made globaly available information by using Ibis [2].

The nodes that have no more work to do; in this case if a node has downloaded all the pieces that it should download from the sender, it starts to steal pieces from other clusters. Thus the faster nodes download more number of pieces for their cluster, thus more pieces is available locally for the slower nodes [2].

3.7 Inference from different methods of Multicast

In the receiver initiated methods Bullet, BitTorrent and Chainsaw it is proved that meshes outperform tree structures.

MOB and Robber instead of employing random meshes over the network, clustering neighbour nodes and adding peers from other clusters performed better than random mesh based multicast methods. Thus clustering of nodes in efficient than other methods of multicasting.

4 Problem Description

In Robber, if a node requires a piece of data which is not available in its cluster; it literally means the node has downloaded all pieces available in its cluster; which also means that the particular node is faster than its neighbours. The unavailability of a piece in its neighbours forces the faster nodes to be idle until availability of that particular piece notified by any one of its neighbours. To solve this problem, the nodes in Robber add peers (global peers) from other clusters, so that a fast node after finishing its work can search and download a required piece, not available in its cluster from nodes belonging to other clusters.

In this approach, each node is connected to a set of nodes from other clusters, termed as global peers. If a faster node and a slower node are connected as global peers, then the slow node has to serve both its local and global peers which would grow the load over the slow node, since it is forced to serve for both local and global peers.

Robber uses Ibis to globalize the information about the clusters and the nodes in the clusters. Thus each node knows the required information about the nodes. Ibis is a third party application which should be installed in all the computers taking part in the multicast operation.

Though the information about the nodes and cluster are available globally, the information about the availability of a data packet in a cluster is not readily known to any node in the system when robber is employed. Thus the node has to probe all the global peers connected to it, to find out the availability of the required piece of data.

To overcome the problems identified in cluster based algorithms MOB and Robber, RIPD has been proposed, which will be described in the following sections.

5 RIPD – Receiver-Initiated Parent-Driven Multicast

In clustered receiver initiated multicast, only the end nodes or the receiver nodes plays an active role, whereas in RIPD the end node's activities are controlled and directed by their parent nodes. Thus a hierarchy is maintained in between the nodes in the grid environment. In RIPD there are cluster heads for each and every cluster of nodes. Each cluster is partially controlled by its cluster head, by directing the nodes in its clusters for downloading or uploading data. Either the cluster head notifies the node about the availability of the packet it is requesting for or it notifies the node if some other node requests for the packet it is having.

The key issues in RIPD are determining the cluster heads and establishing connection between nodes belonging to different clusters whenever needed.

5.1 Selecting the Cluster Head

The selection of cluster head in RIPD is based on the preferential criteria (faster nodes in RIPD). All nodes in a cluster can directly communicate to any node within the cluster. Cluster heads acts as the gateway for establishing connection between nodes of different clusters, but do not interfere with the data transfer between nodes. In this paper cluster head selection is based on AHP [23] algorithm. AHP implements efficient methods to select cluster head, and helps to minimize the number of cluster head changes.

5.2 Establishing Connection

As RIPD is a receiver initiated multicast any data transfer is initiated by the receiver node only. A node requests for all the data packets available within its cluster, and thus, when no more new packets are available within its cluster, the node requests its cluster head to check for new packets. The establishment of connection between nodes from different clusters has two different approaches as described below. There are different scenarios for the same which differ by the type of request made by the node.

5.2.1 Scenario 1

A node 'n' after downloading all the available packets in its cluster requests for a packet i or a sequence of packets from i to i+j it needs to download. This request is forwarded to the

cluster head. The cluster head checks for the availability of the requested packet by forwarding the messages to other cluster heads. From all the messages received from other cluster heads, the cluster head selects a particular node 'm' from which the data should be downloaded and initiates node 'n' to establish a connection with node 'm'.

Consider the two clusters (ni, hn ϵ C1; i = 1 to 10) and (mj, hm ϵ C2; j = 1 to 10), the indexes of the pieces are from 1 to 100.

Pseudocode for forwarding a request:

```
Step 1: ni -> hn (req, 20, 40);
Step 2: hn -> hm (req, 20, 40);
Step 3: do
Step 4:        hm -> mj (req, 20, 40);
Step 5:        j = j + 1;
Step 6: while (j < 10)
```

Pseudocode for forwarding the response message:

```
Step 1: hm collects responses from all nodes of the form
 mj -> hm (resp,p,p+q); [where p <=20; p+q => 40]
Step 2: bestNode = selectBestNode (respList, weightMatrix);
Step 3: hm -> hn (initConnection, bestNodeID);
Step 4: hn -> ni (initConnection, bestNodeID);
```

5.2.2 Scenario 2

A node 'n' after downloading all the available data packets requests for new data packets to its cluster head. The cluster head checks for new data packets by requesting the data packets available to be shared by other clusters. After receiving the sequences the cluster head decides from where it should download, and store the list of new packets for future reference.

Considering the same example of clusters described in the previous scenario.

Pseudocode for forwarding a request:

```
Step 1: ni -> hn(req);
Step 2: hn -> hm(req);
Step 3: do
Step 4:        hm -> mj(req);
Step 5:        j = j + 1;
Step 6: while (j < 10)
```

Pseudocode for forwarding the response message:

```
Step 1: hm collects responses from all nodes of the form
 mj -> hm(avail, p, q); [where p <=1; p+q => 100]
Step 2: bestNode = selectBestNode (respList, weightMa-
 trix);
Step 3: hm -> hn(initConnection, bestNodeID);
Step 4: hn -> ni(initConnection, bestNodeID);
```

5.3 Data Transfer

When a connection is made, the receiver node directly requests the sender for the pieces it needs. Thus the cluster heads in RIPD only guides the node to make connection whenever necessary. After a connection is made the nodes directly communicate, transfers the required data and terminates the connection when transfer is complete.

Considering the same example of clusters described in the previous scenario. If the pieces from p to q requested by ni are available in mj the actual data transfer takes place as follows.

Pseudocode for actual data transfer is as follows:

```
Step 1: ni -> mj (init);
Step 2: mj -> ni (initAck);
Step 3: do
Step 4:        ni -> mj (req, p); [p is the index of the requested
 piece]
Step 5:        mj -> ni (data, p, actualData)
Step 6:        p = p + 1;
Step 7: while (p < q)
Step 8: ni -> mj (termConnection);
Step 9: mj -> ni (termConnection);
```

6 RIPD – Cluster Head Selection

The steps of cluster head selection in RIPD algorithm is as follows. Initially the nodes of a cluster n ϵ C elects for the cluster head node hn for C, where hn ϵ nC. The elected node serves as the hn for C until there are 2 nodes in the cluster including the cluster head. Thus if there are only two nodes (n,hn) in cluster C, hn continues to serve as the cluster head for C. The election is based on the factors that directly influence the network performance; the speed of node's NIC (Network Interface Card) and its processing speed. The node scoring highest weight in AHP wins the chance to serve as hn.

Calculate overall local weights of nodes based on the deciding factors (speed of NIC and processing speed). If the cluster C has k number of nodes, there are 2k numbers of deciding factors which are

NIC speed (α) = {a1, a2, a3, ..., ak}
Processing Speed (β) = {p1, p2, p3, ..., pk}

From these values the weight matrix is derived, the hn is selected based on the weight matrix. Weight matrix is updated whenever a new node joins the cluster. The node with highest weight in the weight matrix serves as the hn. If the head node leaves the network for any reason, the node with next high value is selected as hn.

7 Sequence Windows and Message Formats

The important feature of RIPD is its ability to make connection dynamically and also its ability to serve new nodes instantly with the same efficiency. The nodes are connected based on the request it makes. The nodes request for new pieces to download whenever they find no new pieces to download from their own cluster. There are different message formats used in RIPD as explained as follows. Before describing about the message formats, some introduction has been given about the different windows maintained by the nodes for ease of communication.

7.1 Window of Initiation

This is the list of pieces the node is bound to download from the source node, only after downloading all the pieces in the window in initiation a node can seek for new pieces from the other nodes.

7.2 Window of Possession

This window shows the sequence number of the pieces downloaded by the node.

7.3 Window of Availability

This is the sequence number of pieces the node is willing to upload to any node that needs them. A node will share only a single set of pieces from i to i+j. For any given time a node will not list different sequence of pieces in the available window.

7.4 Window of Interest

This is the sequence number of pieces a node is willing to download from other nodes.

7.5 Window of Desire

If a node finds a node willing to send the packets it is interested to download, a node forms a window of desire, which is usually a subset of window of interest and downloads pieces listed in this window.

Usually a node desires and downloads a set of continuous pieces, rather downloading everything in random fashion. This is because just mentioning the start and end of any sequence mentioned above will be less expensive when compared to specifying the index of each and every piece.

The message formats mentioned in Table 2 are used by the nodes for efficient communication between nodes to share the information about the status of any node participating in the multicast.

TABLE 2
MESSAGE FORMATS USED IN RIPD.

Messages	*Format*
Start, Connection Ack, Blank Request, Finish, Stop	opcode(byte)
Request Pieces, Available	opcode(byte), piece start index(integer), end index(integer)
Desire	opcode(byte), piece index(integer)
Initiate Connection	opcode(byte), nodeID(integer)
Data	opcode(byte), piece index(integer), data field(few kb)

All the messages mentioned in Table 3 are sent and received by every node in the grid environment but initiate connection is used only by the hn to direct the receiver node to download piece from the node mentioned in nodeID. This is because, all the other messages are transferred to nodes in other clusters using hn as a gateway, so that the hn can guide the nodes to download piece from a node of its choice, so that the makespan is reduced.

7 SIMULATION SETUP AND RESULTS

RIPD is evaluated under four different scenarios with changes in the number of nodes and the size of clusters. There are different test cases which are introducing new nodes during the multicast, simulate failure by killing many nodes from the environment, and also by killing almost 70% of the cluster heads during multicast to prove the robustness of the algorithm. The simulation setup consists of nodes with 1 MB/s network speed. There are 3 processing elements used in each node capable of processing at 327 or 377 or 477 MIPS. The processing elements are shared unevenly so that some nodes will have a higher processing speed than the others. Each cluster is employed with a hub node which is used to direct traffic control between different clusters.

When a multicast is initialized using RIPD, the algorithm starts actual multicast only after selecting cluster heads for each clusters in the grid, after which the performance gradually increases to an average of 6.4 MB/s. This performance is plotted in the Fig. 2.

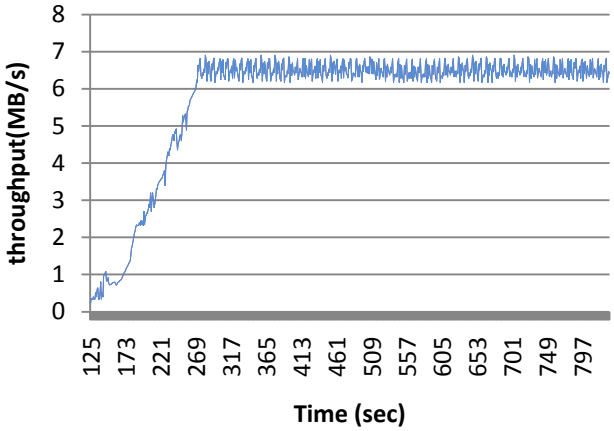

Fig. 1. Initializing multicast using RIPD.

To test the robustness and flexibility of RIPD, some test cases are performed during the simulation test. First the flexibility of RIPD is tested by introducing 50% more new nodes during the multicast.

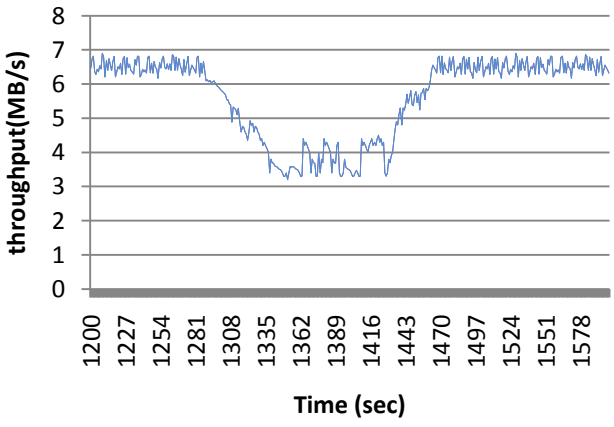

Fig. 2. Performance of RIPD - Addition of new nodes.

In our test case 100 nodes in the network is increased to 150 nodes. The performance drops upon the addition of new nodes, but stabilizes and quickly rises to the average performance as shown in the Figure 2.

From the above test case it is clear that RIPD is flexible to dynamic changing environments where new nodes joins multicast often. Since the recovery time of RIPD is low there will not be a big impact of the addition of new nodes to the multicast.

The robustness of RIPD should be proved, as there is a possibility of nodes to disconnect from the grid during multicast. To test such case, random node failure is initiated to the network. Thus a mass node failure which also includes the head nodes is introduced in the network. In our test case, for a total number of 200 nodes, 100 nodes including 20 cluster head are selected at random and disconnected from the network. The performance of RIPD on such a case is shown in Figure 4, where it is clear that the overall performance is affected for around 300 seconds.

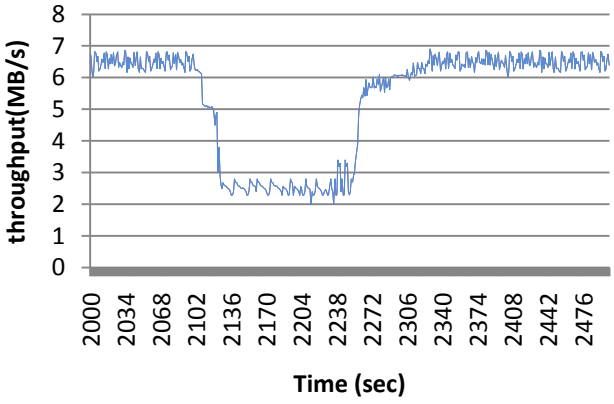

Fig. 3. Performance of RIPD - Node failure.

This test case is to test the robustness of RIPD when many number of cluster heads fail all of a sudden. To test such a case a mass failure of cluster heads is introduced in the network. When 20 cluster heads of the total 30 clusters are mad to fail in the network, the network experiences a failure condition for less than 60 seconds. Since AHP algorithm used, the next possible cluster head is already known, the failure time also constitutes to the time taken for the new cluster head to gather information about the cluster.

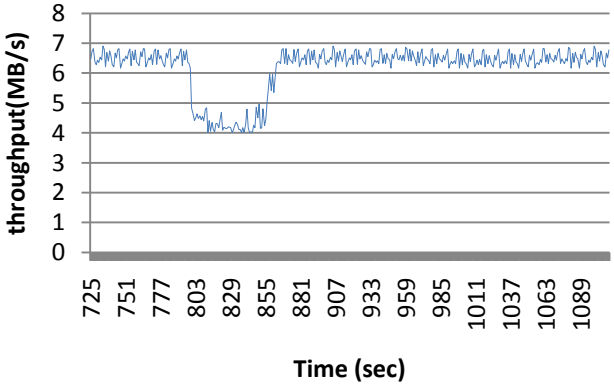

Fig. 4. Performance of RIPD - Cluster head failure.

The performance of RIPD has been compared with BitTorrent and Robber algorithms under different environments by changing the speed of the NIC of nodes participating in multicast, by changing the size of data to multicast. The performance comparison based on these different test cases has been discussed below.

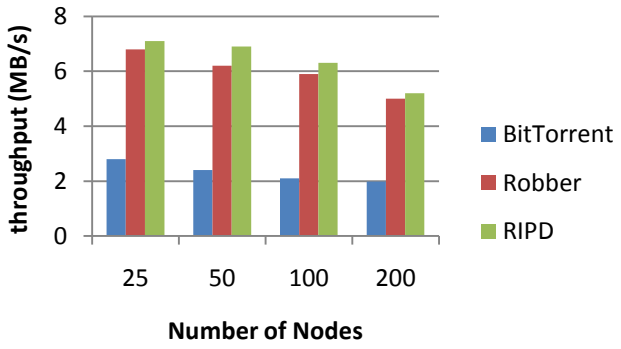

Fig. 5. Performance comparission of RIPD BitTorrent and Robber based on the number of nodes taking part in the multicast.

In the simulation environment the root node sends 100 MB of data to all other nodes, the intra cluster speed is set to 500 KB/s and the inter cluster speed is set to 100 KB/s.

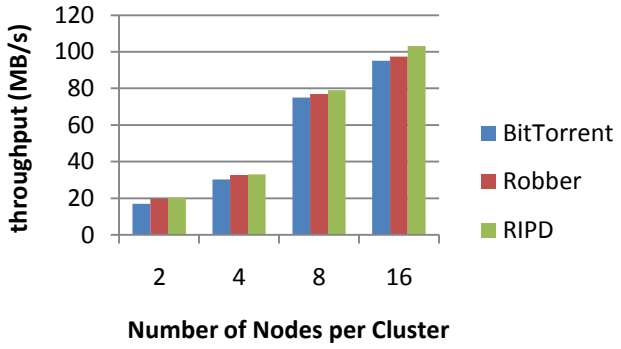

Fig. 6. Performance comparission of RIPD BitTorrent and Robber in a homogeneous environment, based on the number of nodes in a cluster for a total number of 96 nodes.

In the simulation environment the root node sends 1 GB of data to all other nodes. Each node has a 100 Mbit NIC. Nodes in the same cluster communicate directly with each other, whereas the nodes of different cluster has to connect via the root node. After the connection has been established the nodes communicate diectly with the help of NIC address. There is no constraint related to the speed of node and hence, a connection is established whenever there is a free node in any cluster. Thus the communication is faster and the troughput also shoots up to high values. The same cannot be expected in a heterogeneous network as the nodes are of different speed and hence the hn cannot make instant connection, a node has to wait for a suitable node of other cluster to be free to download data.

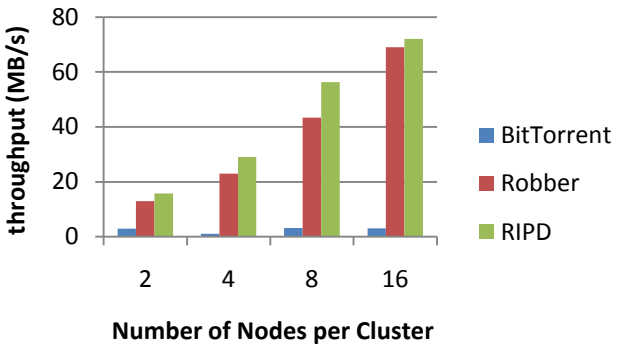

Fig. 7. Performance comparission of RIPD BitTorrent and Robber in a heterogeneous environment, based on the number of nodes in a cluster for a total number of 96 nodes.

In this environment the 26 nodes are enabled with 500KB/s NIC, 35 nodes are enabled with 10 Mbit NIC and the other 35 is enabled with 100 Mbit NIC. There is a huge variation in the connecting speed of nodes in the environment. The root node sends 1 GB of data to all other nodes and the performance comparission is shown in Figure 8.

7 Conclusion

The time of completion, makespan of large-data multicast depends on the achievable bandwidth than can be obtained between the interconnecting links of a network. Perfect QoS can be provided if the control messages are received properly. But as we saw the drawback in the IP Multicast method; even after knowing the exact tree structure and the losses at the destination nodes, finding the exact loss pattern was not possible. In sender-initiated methods the nodes are arranged across a tree structure and data is forwarded through the tree structure; which may sound good, but may push the network to a local or global bottleneck condition. To provide efficient multicast an optimized spanning tree must be calculated, it is a NP-Hard problem if the spanning tree is optimized to maximize the throughput. Also it is hard to find the best spanning tree for a varying and dynamically changing network.

To overcome these problems found in sender-initiated algorithms receiver-initiated algorithms has been proposed. In receiver based methods random meshes are constructed among the receiver nodes; the receivers request the pieces of data they require either from the root node or the neighbor nodes. The basic idea is to use the actual network links among nodes to multicast by automatically adapting to the achievable bandwidth.

The use of meshes gives rise to a new problem of letting the faster nodes to be idle due to the unavailability of data in its neighbor nodes and thus cluster based algorithms has been proposed.

In this paper new of introducing cluster head nodes in every cluster of nodes has been proposed. The basic cluster head selection method has been implemented in this approach. The cluster head are used to make connections dynamically between nodes from different clusters based on their connection speed. Thus the senders from other clusters do not suffer from overhead of serving nodes in local and global clusters simultaneously. The information about the nodes and clusters are available globally and hence the use of software applications like Ibis has been avoided.

RIPD improves the makespan of multicast without the need of external network information. It has proved its robustness in network failure and also it works well in the dynamically changing environment also. This algorithm can be improved by introducing a high level cluster head selection method by checking the reliability of the nodes, and more parameters can be accounted for making dynamic connections between nodes of different clusters.

References

[1] R. Izmailov, S. Ganguly, and N. Tu, "Fast Parallel File Replication in Data Grid," *Proc. Future of Grid Data Environments Workshop (GGF-10)*, Mar. 2004.

[2] M. den Burger and T. Kielmann,"Collective Receiver-Initiated Multicast for Grid Applications" *IEEE Transactions on Parallel and Distributed System*, vol. 22, no.1, pp. 231-244, Feb 2011, doi: 10.1109/TPDS.2010.76. (IEEE Transactions).

[3] V. Kumar, A. Grama, A. Gupta, and G. Karypis. "Introduction to Parallel Computing." *The Benjamin/Cummings Publishing Company*, Inc., 1994.

[4] H. Rangwala, E. Lantz, R. Musselman, K. Pinnow, B. Smith, and B. Wallenfelt, "Massive Parallel BLAST for the Blue Gene/L," *Proc. High Availability and Performance Computing Workshop (HAPCW '05)*, Oct. 2005.

[5] O. Beaumont, L. Marchal, and Y. Robert, "Broadcast Trees for Heterogeneous Platforms," *Proc. 19th Int'l Parallel and Distributed Processing Symp. (IPDPS '05)*, Apr. 2005.

[6] M. R. Garey and D. S. Johnson. Computers and Intractability, "A Guide to the Theory of NP-Completeness." *W. H. Freeman and Company*, 1979.

[7] O. Beaumont, A. Legrand, L. Marchal, and Y. Robert. "Pipelining broadcasts on heterogeneous platforms." *International Parallel and Distributed Processing Symposium IPDPS'2004*, IEEE Computer Society Press, 2004.

[8] A. Adams, T. Bu, J. Horowitz, D. Towsley, R. Caceres, N. Duffield, F. Lo-Presti, S. Mon, and V. Paxson. "The use of end-to-end multicast measurements for characterizing internal network behaviour." *IEEE Communications Magazine, 38(5)*, 2000.

[9] R. Cohen and G. Kaempfer, "A Unicast-Based Approach for Streaming Multicast" *Proc. IEEE INFOCOM*, Apr.2001.

[10] M. Castro, P. Druschel, A. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, "SplitStream: High-Bandwidth Multicast in Cooperative Environments" *Proc. 19th ACM Symp*, Oct.2003.

[11] B. Lowekamp, B. Tierney, L. Cottrell, R. Hughes-Jones, T. Kielmann, and M. Swany, "A Hierarchy of Network Performance Characteristics for Grid Applications and Services," *Proposed Recommendation GFD-R-P.023, Global Grid Forum*, 2004.

[12] T. Gross, B. Lowekamp, R. Karrer, N. Miller, and P. Steenkiste, "Design, Implementation and Evaluation of the Remos Network," *J. Grid Computing, vol. 1, no. 1*, pp. 75-93, May 2003.

[13] J. Maassen, R.V. van Nieuwpoort, T. Kielmann, K. Verstoep, and M. den Burger, "Middleware Adaptation with the Delphoi Service," *Concurrency and Computation: Practice and Experience, vol. 18, no. 13*, pp. 1659-1679, Nov. 2006.

[14] T. Kielmann, R.F. Hofman, H.E. Bal, A. Plaat, and R.A. Bhoedjang, "MagPIe: MPI's Collective Communication Operations for Clustered Wide Area Systems," *Proc. ACM SIGPLAN Symp. Principles and Practice of Parallel Programming (PPoPP)*, pp. 131-140, May 1999.

[15] N.T. Karonis, B.R. de Supinski, I. Foster, W. Gropp, E. Lusk, and J. Bresnahan, "Exploiting Hierarchy in Parallel Computer Networks to Optimize Collective Operation Performance," *Proc. 14th Int'l Parallel and Distributed Processing Symp. (IPDPS '00)*, pp. 377-384, May 2000.

[16] Y. Cui, B. Li, and K. Nahrstedt, "On Achieving Optimized Capacity Utilization in Application Overlay Networks with Multiple Competing Sessions," *Proc. 16th Ann. ACM Symp. Parallelism in Algorithms and Architectures (SPAA '04)*, pp. 160-169, June 2004.

[17] M. den Burger, T. Kielmann, and H.E. Bal, "Balanced Multicasting: High-Throughput Communication for Grid Applications," *Proc. Conf. Supercomputing (SC '05)*, Nov. 2005.

[18] D. Kosti_c, A. Rodriguez, J. Albrecht, and A. Vahdat, "Bullet: High Bandwidth Data Dissemination Using an Overlay Mesh," *Proc. 19th ACM Symp. Operating System Principles (SOSP-19)*, Oct. 2003.

[19] B. Cohen, "Incentives Build Robustness in BitTorrent," *Proc. First Workshop Economics of Peer-to-Peer Systems*, June 2003.

[20] V. Pai, K. Kumar, K. Tamilmani, V. Sambamurthy, and A. Mohr, "Chainsaw: Eliminating Trees from Overlay Multicast" *Proc. Fourth Int'l Workshop Peer-to-Peer Systems (IPTPS '05)*, Feb. 2005.

[21] M. den Burger and T. Kielmann, "MOB: Zero-Configuration High-Throughput Multicasting for Grid Applications," *Proc. 16th IEEE Int'l Symp. High Performance Distributed Computing (HPDC'07)*, June 2007.

[22] Philip K.McKinley, Hong Xu, Abdol-Hossein E and Lionel M.Ni "Unicast-Based Multicast Communication in Wormhole-Routed Networks" *IEEE Tran on Parallen and Dist Systems* Dec 1994

[23] C.-C. Chiang, H.-K. Wu, W. Liu and M. Gerla, Routing in Clustered Multihop, Mobile Wireless Networks with Fading Channel, in: *The IEEE Singapore International Conference on Networks*, 1997, pp. 197–211.