ACSIJ Advances in Computer Science: an International Journal, Vol. 4, Issue 5, No.17 , September 2015
ISSN : 2322-5157
www.ACSIJ.org

ACSIJ
WWW.ACSIJ.ORG

Finding the highest performance for one-to-many content distribution in peer - to- peer networks based on the clustering and loading bandwidth of peer Nodes

Ghaffari, H.R.[1] , Akbari Motlagh, M.[2] ; Akbari Motlagh, A.[3]

[1] PhD in Computer engineering – Software, President of Islamic Azad university, Ferdows branch, Computer engineering group, Ferdows, Iran,
Hamidghaffary53@yahoo.com

[2] Senior Computer Engineering – Software, Islamic Azad University, Ferdows Branch, Department of civil Engineering, Ferdows, Iran,
mobinfun@gmail.com

[3] Senior Civil Engineering – Road, Islamic Azad University, Qayenat Branch, Department of civil Engineering, Qayenat, Iran, aliam394@gmail.com, 00989367013984

Abstract

Considering the problem of IP multicast implementation in routers, in recent years, a lot of alternative methods have been introduced of the application layer multicast (ALM), for one-to-many content distribution. The present study aims to provide a new Algorithm in the field of ALM, to reduce the delay in peer to peer content distribution network (P2P), based on cooperation of M-ary and cluster nodes. All nodes which are close to one another gather in a cluster by means of a fixed number of landmarks that are known nodes. After the close nodes come to each other in a cluster, a tree structure is used to connect them. The algorithm is based on a cooperation between the source node and the content requesting nodes. In this algorithm, the source divides the content into blocks and the blocks are distributed in each cluster through m-ary trees that are all rooted in that source. Based on the mechanism used in this algorithm, all the participating nodes are used as a distributor of content, at least for one time. This algorithm exploits maximum upload capacity of the participating nodes and maximizes the final throughput. Due to the proximity of the nodes in each cluster, the delay in sub-trees of each cluster is less than the delay in similar techniques. On the other hand, the proximity of nodes causes the sub-trees not to be under much stress. Therefore, the final tree will have a very low delay and stress.

Keywords: *one-to-many content distribution, multicast, peer to peer networks, cluster, stress*

1. Introduction

A number of applications such as distribution software, Internet TV, video conferencing, and … require one-to-many content distribution. Multicast is an efficient method to send packets from a transmitter to multiple receivers, and eliminates duplicate packets in the network. On the other hand, with multicast, the transmitter does not need to save the information about all the nodes, therefore the network must be able to perform multicast operation. In IP multicast, multicast operation is performed by the router and all the routers must be capable of performing such an operation [1]. In large-scale networks, the management of the Group and the implementation of multicast routers seem to be impossible. Implementation of the above-mentioned complex problem has caused the multicast at the application layer, or the so-called ALM to be taken into considerations [2]. The current methods for ALM will not change the network infrastructure and unlike the IP multicast that repeats the data in branch routers, the current methods cause data replication in the hosts [3].

Some examples of systems in the ALM field include Scatter cast and Overcast [4]. In both of these systems a single tree is used for content distribution. Compared with letting the source node directly send its content to all other clients, the distribution tree approach reduces the network load of the source, thus achieves more efficient content distribution. In a distribution tree, the intermediate nodes redistribute the content, while the leaf nodes only receive the content. In this situation the upload bandwidths of the leaf nodes are not utilized for content distribution. To solve the problem of inefficiency, CoopNet [5] and SplitStream [6], split the content into multiple stripes and distributed the stripes across separate multicast trees with disjoint interior nodes. Any peer computer could be an interior node in one of the multicast trees, and contribute to forwarding the content. To check the reliability and frequency of large files, we can use Bullet and Fast Replica [7]. If we have n nodes, FastReplica will first divide the file to n equal Subfiles and each subfile is transmitted to a peer node in the group which is subsequently repeated and transmitted to other nodes. In Bullet, the peer nodes are organized into an Overlay tree.

Each node split the content received from the parent into a disjoint set of blocks, with each set sent to a different child node [8]. Afterwards, the child nodes then discovered the missing blocks and the nodes that held the missing blocks, and sent requests to recover the missing blocks. The Mutualcast [9] method divides the content into smaller blocks, since the nodes with broader bandwidth, May further distribution of blocks and nodes while narrower bandwidth may distribute less blocks. In this method, all the content requesting nodes and the non-content-requesting nodes are used. If the bandwidth of the source node is broad, the source will also contribute to content distribution. A P2P system is practically implemented by BitTorrent [10].

These are just a few examples of recent projects is for application at the multicast level. Although distribution strategies of ALM (application-level multicast), are more efficient than direct transmission of content to peers, these methods failed to achieve the most effective and efficient content distribution in networks. Because:

1- The distance between nodes and the source is not taken into consideration in any of these methods.
2- Methods in which the sources make complete use of peer nodes bandwidth for content distribution, will experience a considerable delay in content distribution, because there is a complete connection between the content-requesting nodes in content distribution trees.

Clustercast, is a mechanism for content distribution in P2P networks, in which all nodes close to each other are clustered by the clustering method and the loading bandwidth of the peer nodes is fully used. This mechanism has little control overhead compared to similar algorithms.

The distinctive features of Clustercast are:

First, taking the distance between nodes into consideration, this mechanism clusters the close nodes during a multicast session and increases the distribution performance to the greatest extent possible. Each node with a fixed number of Pings, can find its position and be connected to the multicast group.

Second: Clustercast, using the information about the proximity and closeness of the nodes, the combination of the highest upload capacity, as well as the m-ary trees, minimizes content delivery time in the multicast.

Third: the m-ary trees made in this algorithm are different from the m-ary trees in other algorithms. Because, in this algorithm it is not possible for a node to transmit the blocks to all other nodes. On the other hand, the height of the built m-ary trees is considered fix and equal to 2, to avoid deep structures.

In general, in these algorithm, compared to other algorithms, the values of Stress and Stretch in m-ary trees has decreased significantly, which results from nodes clustering.

The results show that the proposed multicast algorithm, provides a good balance between reduction of delay in end-to-end content distribution and maximum throughput, while maintaining the scalability of the structure and avoiding the topology of the full connection between nodes.

2. Related Works

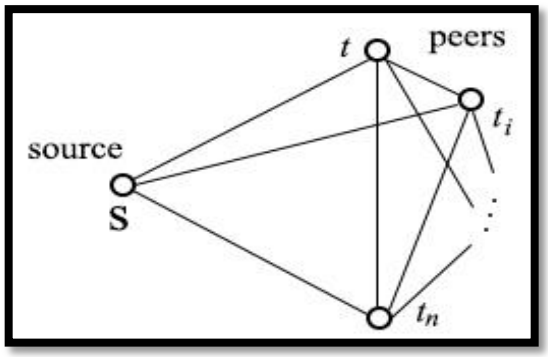

Figure 1: one to many content Distribution

The problem in one to many content distribution shown in Figure 1 is as follows:

This network has a source node that includes content for distribution and several peer nodes, Ti = 1,2,3, ..., N, each of which may or may not request a copy of the content. Both the source node and the peer nodes are end-user nodes. They are usually computers that connect to the Internet through service providers (ISP), cable modem, or university campus and corporate networks. These nodes are not considered backbone nodes or the internet infrastructures. Our target is to distribute the content with maximum throughput to all the destinations. Although this is a straightforward way, the result of the content distribution process depends on the uploading bandwidth of the source node, which is usually limited. Naturally, we want to enlist the help of the peer nodes, and use their upload bandwidths to aide the content distribution.

Let us now examine a number of prior ALM approaches in distributing the content from the source to the peer nodes:

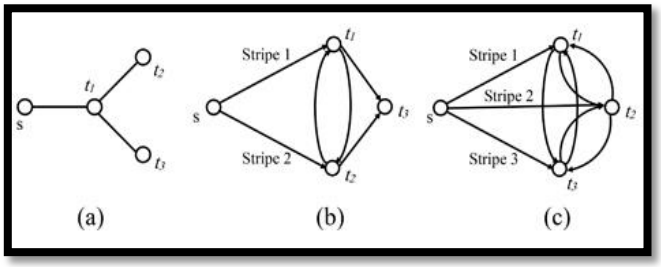

Figure 2: one-to-many content distribution methods
a) Scattercast/Overcast, b) SplitStream/CoopNet, c) FastReplica.

The Scattercast and Overcast [5] are in form of a single distribution tree which is shown in figure 2- (a). In this

configuration, the source node sends data to node t_1, which forwards the data to nodes t_2 and t_3.

The ALM distribution tree utilizes the upload bandwidth of the intermediate node t_1, bandwidths of the leaf nodes t_2 and t_3 are not utilized. To better utilize the bandwidths of the peer nodes, nodes with higher upload bandwidths should be placed upstream, while nodes with lower upload bandwidths should be placed downstream. To solve this problem CoopNet [6], SplitStream [7] divide the content into different stripes and distribute them in a separate multicast tree. As shown in Figure 2-(b), both use stripe plans of content distribution and multicast distribution tree with separate internal nodes. CoopNet / SplitStream are managed by Multicast trees at two levels. The contents are divided into two equal Stripe. The first Stripe is sent to node t_1 and then forwards from t_1 to nodes t_2, t_3. And the second stripe is sent to t_2 and then forwards to nodes t_3, t_1 from there. We should note that the system is improved by utilizes upload bandwidth of nodes t_1, t_2, but has failed to utilize the upload bandwidth of node t_3. To check the reliability and frequency of large files, we can use Bullet [9] and FastReplica [8]. If we have n nodes, FastReplica will first divide the file to N equal Subfiles and each subfile is transmitted to a peer node in the group which is subsequently repeated and transmitted to other nodes. FastReplica is specifically designed for downloading the file. For an N node P2P network, FastReplica distributes the file with N height-2 multicast trees with intermediate degree N-1. A simple example with 3 nodes is shown in Figure 2 (c). FastReplica performs the file distribution in two stages: the First stage is called distribution and second stage is called collection. At the distribution stage, the file is divided into 3 Subfiles and then is sent to nodes t_1, t_2, t_3. After the distribution stage, the collection stage starts. Each of the peer nodes transmits the subfile to the other peer nodes. All peer nodes are used for content distribution in FastReplica. In Bullet, the peer nodes are organized into an Overlay tree. Each node separates the contents from his father in a block set which is different from the set that is sent to the child node. Afterwards, the child nodes find the missing blocks and the nodes that have kept the missing blocks, and send a request to receive them. The Mutualcast [4] method divides the content into smaller blocks, since the nodes with larger upload bandwidth, may redistribute more blocks and the node with smaller upload bandwidth may redistribute fewer blocks. Each content block is assigned to a node for distribution and the responsible node can be a content-requesting node, a non-content requesting node, or even the source itself. The result of this distribution is controlled by the redistribution lines between the source node and the control nodes. This strategy fully uses all the uploading bandwidth in peer nodes and maximizes the delivery power. In addition to that, Mutualcast is simple and flexible. Mutualcast is different from other one-to-many content distribution methods which use a fixed network topology. Mutualcast basic framework is as follows:

The content in Bj block starts to be distributed (j = 1, 2, 1, …, m). Each block Bj, is assigned to a specific node for distribution to other peer nodes. Often, the node responsible for redistribution of block Bj, is the t_i peer node. In this case, the source node sends a copy of the block Bj to the peer node t_i which then redistributes the block Bj by sending a copy of the block to the rest of the peer nodes. However, when the source node uses the frequency bandwidth, the node responsible for distributing the block Bj, could be the source node, in that case, the source node directly sends a copy of the block Bj to each t_i peer node. Mutualcast divides the content into a large number of small blocks. The Size of Mutualcast blocks is based on an agreement between the required distribution and overhead Stripes. The Size of Mutualcast blocks is preferably slightly less than the Maximum Transmission Unit (MTU) of the network. So that each Mutualcast block, can be sent over the network as a single block. Figure 3, shows the Mutualcast distribution network. This network has a source node called S and the content requesting nodes of t_1, t_2, t_3, t_4.

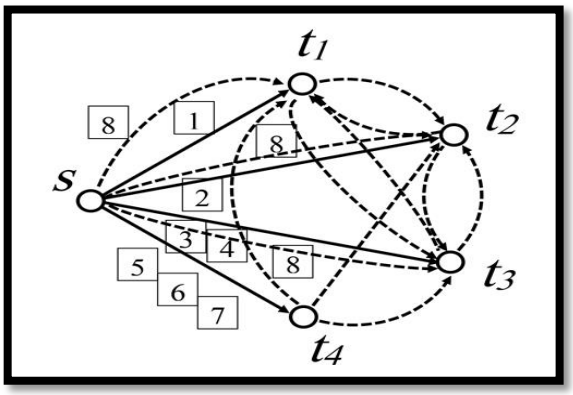

Figure 3: Mutualcast

Among the peer nodes, nodes t_1, t_2, t_3 request a copy of the content from the source s and the t_4 node doesn't request content from source s, yet it helps the content requesting nodes in content distribution by its upload bandwidth. When the blocks 1,2,3,4 are assigned to $t1$, $t2$, $t3$ for redistribution, the responsible node sends the blocks to two other peer nodes. When the blocks 5, 6, 7 are assigned to the t4 node which doesn't requested content, these blocks are sent to peer nodes t_1, t_2, t_3 by t_4. The source node may directly perform content distribution, just like block 8 which is directly sent from the source to peer nodes t_1, t_2, and t_3.

3. Description of the proposed algorithm

Here we aim to maximize the performance as much as

Possible, while taking the closeness of nodes during Mutualcast session as well as their clustering into consideration. In this algorithm, we cluster the nodes using a combination of the highest loading capacity, and the information about their proximity, and by building m-ary trees in each cluster, minimize the content delivery time. This method is generally similar to *Mutualcast* when:

- ✓ the distance between the nodes is the same during a multicast session
- ✓ the entire loading capacity of all the nodes is used
- ✓ there is a full communication between all nodes participating in the multicast session

The m-ary trees that are built in this algorithm are different from those in the Mutualcast method. Because in this method, we do not allow a node to send blocks to all the other nodes and content distribution takes place only within the cluster. According the approach used in this method, the peer to peer networks should have the maximum multicast throughput and use the full uploading capacity of all the content requesting nodes.
As the peer nodes may be in different parts of the world, then in a multicast session in a large heterogeneous environment, the nodes need to cooperate with one another

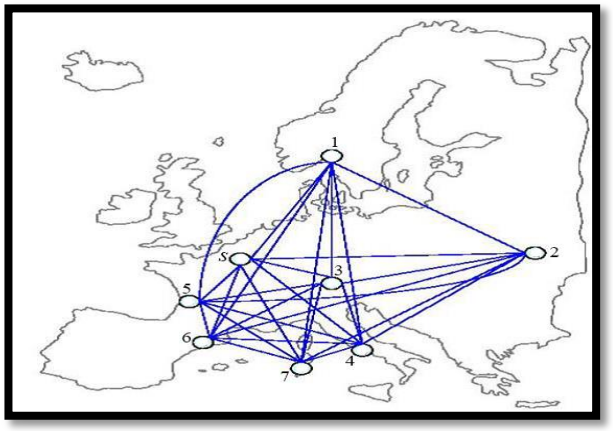

Figure 4: geographic expansion of the nodes participating in multicast

Here, the Mutualcast idea is extended by adding the clusters based on proximity of the nodes. In this method, a number of Landmarks are used to measure the distance between nodes. In this method, we assume that the participating nodes use their full upload capacity. On the other hand, we assume that the rate of access to the network is asymmetric and hence the upload capacity of resources is considered limited. Each peer node can contribute to content distribution in one or more distribution trees. The source divides the content into blocks for distribution, and then creates a set of m-ary trees in each cluster, where the content blocks are distributed. Each peer node receives one or more content blocks from the source. First, the nodes with high upload capacity are selected for content distribution, while the nodes with lower upload capacity are used as the leaves on the multicast trees. When the upload capacity of the selected nodes is lower than a specific level, these nodes are used as leaves in other distribution. We assume that the source knows the upload capacity of all peer nodes as well as their IP address. In addition, the distance between nodes is recognizable.

3.1. How to measure the distance between nodes (RTT)

To measure the distance between nodes, some landmarks are used. Each node can gain access to the IP addresses of the nodes through the DNS. Each node must ping these landmarks which are known nodes and don't belong to the multicast group. In order to cluster the near nodes in one group, each node pings these landmarks in a specific order and calculates its delay for each of them individually. Afterwards, the node multiplies these numbers by their placement, so that the number of the first landmark is multiplied by one, the second number is multiplied by three, and the third is multiplied by nine and the fourth number is multiplied by 27 (the numbers are ternary) and their results are added together. As a result of these calculations a number is obtained (RTT), which is used in the node clustering algorithm. For example, if there are four landmarks and delay obtained from each of them is 50, 120, 75, 210 respectively, the calculated number is equal to:

$$50 * 1 + 125 * 3 + 75 * 9 + 125 * 27 = 6577$$

3.2. How to save distances and the capacity of nodes

Each of the participating nodes is denoted by Pj and their load capacity is denoted by Cj, and then saved in the list C:

$$C = \{C1, \ldots, Cj, \ldots, C_k\}$$

The distance between nodes Pi, Pj is denoted by Di (j) and saved in the Di list:

$$Di = \{ Di(1), \ldots, Di(j), \ldots, Di(k) \}$$

The normal distance for each peer node is calculated from the following formula:

$$D_i^n(j) = \frac{D_{i(j)}}{max\{D_{i(j)}\}_{D_{i(j)} \epsilon D_i}} \quad , \quad 0 < D_i^n(j) \leq 1 \qquad (1)$$

The Normal load capacity (C_i^n) for each peer node is calculated from the following formula:

$$C_j^n = \frac{C_j}{\max\{C_i\}_{C_i \epsilon C}} \quad , \quad 0 < C_j^n \leq 1 \qquad (2)$$

The Values of $Di^n(j)$ and Cj^n are saved in the following lists:

$$D_i^n = \{ D_i^n(1), \ldots, D_i^n(j), \ldots, D_i^n(k) \},$$
$$C^n = \{ C_1^n, \ldots, C_i^n, \ldots, C_k^n \}$$

In the m-ary tree construction algorithm, these lists are used to select the desired nodes. Nodes with higher C_j^n and lower D_i^n (j), have higher efficiency for selection.

3.3. Calculation of the preliminary loading capacity

At this stage, an additional parameter that is effective in node selection is introduced:

PDR: Preliminary delivery rate

This factor is used to determine how many times each node, as a peer node in a different distribution trees, can be used as a distributor in each cluster.

- $$\mathrm{PDR_c} = \frac{\sum_{j=1}^{k_c} C_j}{k(K_c - 1)} \qquad (3)$$

K_c: The number of nodes per cluster

$$\text{Preliminary capacity} = \frac{\text{total cluster capacity}}{\text{The total number of content} - \text{requesting nodes (the number of cluster nodes} - 1)}$$

3.4. Calculating the value of m in m-ary trees (the number of clusters)

Here, we aim to use all the participating nodes for construction of m-ary trees. The initial number of these trees is equal to the number of content requesting nodes. To avoid deep structures, the height of the trees is considered fixed and equal to two, therefore, the maximum number of content requesting nodes in each m-ary tree is equal to m (m + 1).

Now, assuming that the number of content requesting nodes in the multicast group is already specified (k), the source will determine the value of m according to the following formula:

$$m = \left[\frac{1}{2}\sqrt{4k+1} - \frac{1}{2}\right] \qquad (4)$$

3.5. How to cluster nodes

Considering the calculated value for m and RTT, the nodes are divided into m clusters. This process is carried out in the following order:

1. Selection of m nodes as clusters' center of gravity. (Ramin Rajabioun [11])
2. Calculation of the distance between each cluster and the selected canters of gravity.

   $$|\text{center of gravity(RTT)} - \text{cluster (RTT)}|$$

3. The nearest value obtained for each node, is selected as its cluster.
4. New center of gravity for each cluster is recalculated based on the nodes in that cluster.

   $$\text{New center of gravity} = \frac{\text{total cluster capacity}}{\text{the number of cluster nodes}}$$

5- This process goes on until the centers of gravity change.
6. Examination of clusters, so that each cluster includes at least two nodes.
7. Examination of clusters, so that each cluster includes no more than m + 1 nodes
8- Saving information about the nodes clusters in CL list:
CL= {L (p1), …, L (pj), …, L(pk)}

To clarify the point, we implement the clustering algorithm on 8 nodes. The distance between nodes (RTT) is calculated in accordance with Section 3-1.
Figure 5 shows the different stages of clustering.

	RTT	96	264	140
1	196	100	68	56
2	264	168	0	124
3	152	56	112	12
4	140	44	124	0
5	100	4	164	40
6	88	8	176	52
7	120	24	144	20
8	96	0	168	44

(a)

	RTT	94.67	264	152
1	196	101.33	68	44
2	264	169.32	0	112
3	152	57.33	112	0
4	140	45.33	124	12
5	100	5.32	164	52
6	88	6.67	176	64
7	120	25.33	144	32
8	96	1032	168	36

(b)

	RTT	101	264	162.67
1	196	95	68	33.33
2	264	163	0	101.33
3	152	51	112	10.66
4	140	39	124	22.66
5	100	1	164	62.66
6	88	13	176	74.66
7	120	19	144	42.66
8	96	5	168	66.66

(c)

Figure 5: the stages of node Clustering

First, the number of clusters is calculated according to the formula 4 (m = 3). Afterwards, the nodes 96, 264, and 140 are randomly selected as the center of gravity. In Figure 5, the left column of the tables shows the number of nodes and the next column shows the delay obtained for each node. In the next stage, the distance of each node from the center of gravity nodes should be calculated. For example, for the first node:

196 – 96 = 100 *** 196 – 261 = 68
196 – 140 = 56

The smallest number obtained is the cluster of each node. For example, cluster 3 is selected for node 1. Clusters for

the rest of the nodes are selected in the same way, the cluster is identified. Figure 5 - (a) shows the first stage of clustering. At the end of this stage, the clusters of each node are as follows:

Cluster 1: {8, 7, 0} **cluster 2**: {2} **cluster 3**: {1, 3, 4, 5}

After this stage, the new centers of gravity are calculated according to the nodes of each cluster:
C1 = (100 + 88 + 96) / 3 = 284 / 3 = 94.67
C2 = 264
C1 = (196 + 152 + 140 + 120) / 4 = 608 / 4 = 152
As the centers of gravity have changed, the previous stage should be repeated. Figure 5 - (b) shows the second phase of clustering. At the end of the second phase, the nodes of each cluster are as follows:

cluster 1:{8, 6, 5, 7}**cluster 2**: {2} **cluster 3**: {1, 3, 4}

After this stage, the new centers of gravity are calculated according to the nodes of each cluster:

$$C1 = (100 + 88 + 120 + 96) / 4 = 404 / 4 = 101$$
$$C2 = 264$$
$$C3 = (196 + 152 + 140) / 3 = 488 / 3 = 162.67$$

As the centers of gravity have changed, the previous stage should be repeated. Figure 5 - (c) shows the third phase of clustering. At the end of the third phase, the nodes of each cluster are as follows:

cluster1 :{8, 6,5,7} **cluster 2**: {2} **cluster 3**: {1, 3, 4}

After this stage, the new centers of gravity are calculated according to the nodes of each cluster and as the centers of gravity have not changed compared to the third stage, this stage comes to an end. Since m-ary trees are used for content distribution, none of the cluster should have more than m + 1 node, and since we aim to make use the full capacity of the participating nodes, none of them should include less than 2 nodes. Cluster 2 has only one node and at least one other node should be added to that. After checking the clusters, the nearest node to node (2) is selected so that the capacity of node 2 is used for content distribution and the delay in content distribution reduces at the same time. The final Clustering is as follows:

cluster 1: {8, 6, 5, 7} **cluster 2**: {2, 1} **cluster 3**: {3, 4}

3.6. How to build m-ary trees in each cluster

1- First the values of Cj^n, Di^n and CL is calculated for each of the nodes, and the PDR

2. Selection of a node in each cluster as the distributor node, which has the highest capacity among the nodes of the cluster.
3. The rest of the cluster nodes are assigned, as leaf, to the selected nodes in each cluster and the remaining capacity of the selected nodes is replaced by the previous values in the Cj^n list.
4. Selection of the nodes in each cluster for building distribution trees continues until no new capacity is left for building them.
5. Afterwards, a node with the largest value of Cj^n is selected from C^n, so that it is not used previously.
6. When the loading capacity of the next node in the cluster is less than PDR in that cluster, a new PDR time is calculated for the cluster.
7- Finally, a time comes when all the nodes are being used, but the set of m-ary trees is not yet complete. The nodes in each cluster, whose capacity is not fully used, are examined by calculation of the new value for PDR.
8. At each stage, when there are more than m nodes with the same upload capacity, the node that is closer to the source will be selected.
9. When the number of m-ary trees is equal to K, the set is complete.
10- In the next stage, the duplicate trees are identified and removed in order to reduce the number of m-ary trees. The advantage of reducing m-ary trees is that we can reduce the number of blocks sent from the source, and increase the size of blocks in return. Another advantage of the reduced set of m-ary trees is that linear programming (LP: Linear Programming) can be used to calculate the optimal size of the blocks to maximize the output function (f) in the set of distribution trees.
To clarify the point, we implement the Clustercast algorithm with 9 content requesting nodes in the multicast group of K and with a transmitter, we will implement. Based on the number of content-requesting nodes (K), 9 m-ary trees are created to distribute the blocks x_1 to x_9. Table 1 shows the distance between the content requesting node and the source which is obtained based on the explanations in section 3-1.
Table 2 shows the load capacity of the content-requesting nodes and the source.
To determine the number of clusters and m-ary distribution trees by the formula (4), the value of m is equal to:

$$m = \left\lceil \frac{1}{2}\sqrt{4*9+1} - \frac{1}{2} \right\rceil = \left\lceil \frac{1}{2} * 6.08 - \frac{1}{2} \right\rceil = \left\lceil 3.04 - \frac{1}{2} \right\rceil = \lceil 2.54 \rceil = 3$$

In the next stage, the content-requesting nodes are clustered according to the descriptions provided in 3-5 section and the calculated value for m. The clustering is as follows:
Cluster 1: { P_3, P4, P_6, P_8} **Cluster 2**: {P_7, P_5, P_9}
Cluster 3: {P_1, P_2}

Table 1: distance between nodes and the source

Pi	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**source**
1	0	64	114	78	164	102	238	104	128	**140**
2	0	0	50	14	100	38	174	40	64	76
3	0	0	0	36	50	12	124	10	14	26
4	0	0	0	0	86	24	160	26	50	62
5	0	0	0	0	0	62	74	60	36	24
6	0	0	0	0	0	0	136	2	26	38
7	0	0	0	0	0	0	0	134	110	98
8	0	0	0	0	0	0	0	0	24	36
9	0	0	0	0	0	0	0	0	0	12

Table 2: the Capacity of the resource and nodes

Pi	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**source**
Ci	100	500	300	400	200	300	200	600	500	400

According to formula 3, the preliminary loading capacity for each cluster is calculated in the following way:

$$PDR_1 = \frac{\sum_{j=1}^{k_1} C_j}{k(K_1 - 1)} = \frac{600 + 300 + 400 + 300}{9(4-1)} = \frac{1600}{27} = 59.26$$

$$PDR_2 = \frac{\sum_{j=1}^{k_2} C_j}{k(K_2 - 1)} = \frac{500 + 200 + 200}{9(3-1)} = \frac{900}{18} = 50$$

$$PDR_3 = \frac{\sum_{j=1}^{k_3} C_j}{k(K_3 - 1)} = \frac{500 + 100}{9(2-1)} = \frac{600}{9} = 66.67$$

Table 3 shows the remaining capacity of nodes, the preliminary uploading capacity of each cluster and the distributor node in each m-ary tree. The content distribution trees are shown in Figure 6. Trees in dotted lines show the duplicate trees and XI is the new block for content distribution. Relevant clusters are shown in the first content distribution tree. In order to distribute block X_1 in the first m-ary tree, nodes P_8 from the first cluster, P_9 from the second cluster and P_2 from the third cluster which have the highest capacity (the highest level of $C_j n$), are selected.

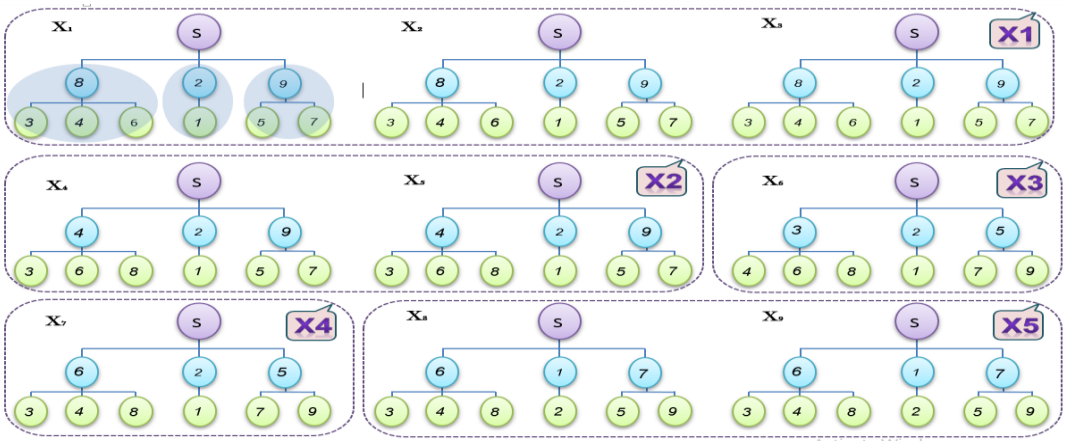

Figure 6: the preliminary content distribution trees, duplicate trees: trees in dotted lines

Table 3: the preliminary Upload capacity, the remaining capacity of nodes in each distribution tree

Cluster1					***Cluster 2***				***Cluster3***			***The m-ary distribution tree***
3	4	6	8	PDR1	5	7	9	PDR2	2	1	PDR3	
300	400	300	**422.2**	59.26	200	200	**400**	50	**433.3**	100	66.67	First
300	400	300	**244.4**	59.26	200	200	**300**	50	**366.6**	100	66.67	second
300	400	300	**66.6**	59.26	200	200	**200**	50	**299.9**	100	66.67	third
300	**222.2**	300	66.6	59.26	200	200	**100**	50	**233.2**	100	66.67	fourth
300	**44.4**	300	66.6	59.26	200	200	0	50	**166.5**	100	66.67	fifth
122.2	44.4	300	66.6	59.26	100	200	0	50	**99.8**	100	66.67	sixth
122.2	44.4	**122.2**	66.6	59.26	0	200	0	50	**33.1**	100	66.67	seventh
122.2	44.4	**82.7**	66.6	13.16	0	100	0	50	33.1	**33.1**	66.67	eighth
122.2	44.4	**43.2**	66.6	13.16	0	0	0	50	33.1	**25.9**	7.38	ninth

The remaining capacity of the distributor nodes for the second distribution tree is as follows:
Remaining Capacity: Capacity of node - (preliminary loading capacity * number of children)

$$R\_C\,(p8) = 600 - (59.26 * 3) = 600 - 177.78 = 422.2$$

$$R\_C\,(p2) = 500 - (66.67 * 1) = 500 - 66.67 = 433.3$$

$$R\_C\,(p9) = 500 - (50 * 2) = 500 - 100 = 400$$

Comparing the remaining capacity of the nodes in each cluster with PDR on the same cluster, we can see that nodes P_8, P_2, P_9 can still be used as feeder nodes in the second and third distribution trees. The remaining capacity of nodes P_8, P_2, P_9 after the distribution of block x_3 will be equal to 66.6, 299.9, and 200 respectively. In the fourth distribution tree, as the remaining capacity of node P8 is less than the preliminary loading capacity × Number of children, it cannot be used as distributor in cluster number one. P_8 = 66 .6 < (59 .26 × 3). In the next stage, the node P_4 which has a higher level of C_j^n, will be selected as the distributor node from among the nodes of cluster 1, By comparing the capacity of the remaining nodes, it is demonstrated that the nodes P_4, P_2, P_9 can still be used as feeder nodes in the fifth distribution tree. At the end of this stage nodes P_9, P_4 cannot be used as a distributor of block x_6. In cluster 1, the node P_3 is selected from among of remaining nodes (P_3, P_6). The nodes P_3, P_6 both have equal capacities, but the value of $D_i^n(j)$ is less in node P_3. In cluster 2 as well, the node P_5 is selected from among the remaining nodes (P_5, P_7). The nodes P_5, P_7 both have equal capacities, but the value of $D_i^n(j)$ is less in P_5. By comparing the capacity of the remaining nodes, it is obvious that node P_3 cannot be used as a distributor of block x_7. In cluster 1, node P_6 is selected to distribute block x_7. At the end of this step in cluster 1, the node P6 cannot be used to distribute x_8, and, on the other hand, the capacity of all the cluster nodes is used, but the number of content distribution trees is still lower than the number of content requesting nodes (K), and the remaining capacity of the node P_6 should be used by a new PDR.

$$PDR_1 = \frac{\sum_{j=1}^{k_1} C_j}{k(K_1 - 1)} = \frac{66.6 + 44.2 + 122.2 + 122.2}{9(4-1)} = \frac{355.2}{27} = 13.16$$

Furthermore, the nodes P_2, P_5 cannot be used to distribute the block x_8, therefore, the node P_7 in cluster 2, and the node P_1 in cluster 3 cannot be used as distributors. At the end of this stage in the cluster 1, the nodes P_1 can be used to distribute the block x_9, and on the other hand, no other node is remained to be selected. In this situation, the source should calculate a new PDR for cluster 3:

$$PDR_3 = \frac{\sum_{j=1}^{k_3} C_j}{k(K_3 - 1)} = \frac{33.3 + 33.1}{9(2-1)} = \frac{66.4}{9} = 7.38$$

All the content distribution trees are shown in Figure 6. In this figure, we can see that duplicate distribution trees are used to distribute blocks x_1, x_2, x_3, and x_4, x_5, and x_8, x_9. Therefore, we can remove the duplicate trees. As shown in Figure 6, the trees in the dotted line are the duplicate trees and after removing these trees, only five distribution trees will be left.

3.7. Calculation of algorithm performance

In the Clustercast network, the bandwidth of the source node, is the most valuable resource, i.e. where content is distributed. If the upload bandwidth of the source node is used up, we can't accelerate content distribution, even if there are still peer nodes with available upload bandwidths. Clearly, if the source sends the content blocks to all the content-requesting nodes (N_1), through delivery links at rate B, it will consume $N_1 \cdot B$ of the upload bandwidth of the source. In other words, if the source node sends the content blocks to t_1 content-requesting node (which in turn distributes the content among other content requesting nodes) at the rate of B, only an amount B of the upload bandwidth of the source node is needed. Apparently, as long as we have more than one content-requesting node, the source node should forward a lot of content blocks to the content-requesting nodes. We assume that the Clustercast network consists of a source node with the upload bandwidth of Cs, and N (N> 1) number of content requesting peer nodes with average bandwidth of C_i in each cluster. Applying the distribution route selection strategy above, the distribution throughput of the Clustercast network, which is defined as the amount of content multicast to the content-requesting peer nodes per second is:

$$f = \begin{cases} C_s, & C_s \le (C_1 + C_2 + \ldots + C_m), \\ (C_1 + C_2 + \ldots + C_m) + \dfrac{C_s - (C_1 + C_2 + \ldots + C_m)}{N}, & C_s \ge (C_1 + C_2 + \ldots + C_m) \end{cases}$$

With $C_i = \frac{N_{ci}}{N_{ci}-1}\ C_{Ni}$

C_s	The source node capacity
C_{Ni}	The average capacity of nodes in each cluster
C_i	The average bandwidth of each cluster
N_{ci}	The number of nodes per cluster
$\frac{C_s - (C_1 + C_2 + \ldots + C_m)}{N}$	Content distribution by the source
N	The total number of content-requesting nodes

Table 4: output parameters in Clustercast

It shows that if the source fails to fully use the upload bandwidth of the content-requesting nodes, then the distribution throughput will be limited only to the upload bandwidth in the source node. All N_1 content-requesting peer nodes receive content at the rate of the upload

bandwidth of the source node. It should be noted that for any path or distribution tree, the amount of network resources consumption for each peer nodes, is independent of the upload bandwidth. Therefore, instead of examining the bandwidth assignment problem for each individual peer node, we examine that for each set of paths. To clarify the point, the output of the example provided in 3-6 section is calculated as below:

$$N_{c1} = 4\ , N_{c2} = 3\ , N_{c3} = 2\ , C_s = 4000$$

$$C_{N1} = \frac{600 + 300 + 400 + 300}{4} = \frac{1600}{4} = 400$$

$$C_1 = \frac{N_{c1}}{N_{c1} - 1}\ C_{Ni} = \frac{4}{4-1} * 400 = 533.3$$

$$C_{N2} = \frac{500 + 200 + 200}{3} = \frac{900}{3} = 300$$

$$C_2 = \frac{N_{c2}}{N_{c2} - 1}\ C_{N2} = \frac{3}{3-1} * 300 = 450$$

$$C_{N3} = \frac{500 + 100}{2} = \frac{600}{2} = 300$$

$$C_3 = \frac{N_{c3}}{N_{c3} - 1}\ C_{N3} = \frac{2}{2-1} * 300 = 600$$

$$f = \left\{(C_1 + C_2 + C_3) + \frac{C_s - (C_1 + C_2 + C_3)}{N} = 1583.33\right.$$
$$\left. + \frac{4000 - 1583.33}{9} = 1851.84\right.$$

4. The evaluation scenario

We implemented a software called multicast distribution simulator (Clustercast) based on the proposed algorithm, The input of this tool consists of the information about the content-distribution source, content requesting nodes and their output, the content distribution paths based on the separation of algorithms and comparison of the output and performance of the algorithm based on distribution parameters. We performed the evaluation for different multicast groups with 6, 7 and 10 content-requesting nodes. In each case, the source capacity ranged from 1000 to 6000 kilobits per second, while the capacity of the content-requesting nodes was considered fixed. The Loading capacity of the content requesting nodes and the distance between them at each will be used. The Loading capacity of content-requesting nodes P_1, P_2, P_3, P_4 P_5, P_6, P_7, P_8, P_9, P_{10} is considered 100, 500, 300, 400, 200, 300, 200, 300, 400, 100 kilobits per second respectively. The distance between the content-requesting nodes and the source is calculated in accordance with Section 3-1. These values in the real environment can be obtained from SmokePing [12] website. We assume that in all stages of evaluation, the source has the enough capacity and can directly distribute the content among the content-distributing nodes.

4.1. Results

The following table shows a comparison between the m-ary trees, in the Mutualcast, NonClustering [13], Clustercast methods as well as the results of their output.
The results show that in a heterogeneous multicast group, the proposed algorithm of Clustercast, Mutualcast, NonClustering are quite equal if the loading capacity of the source is enough to send the content blocks directly to all the content-requesting nodes. Figure 11 shows the comparison between the maximum output values for the assessed algorithms. In the next part, content delivery delay in the algorithm Clustercast is compared to that in two algorithms Mutualcast, NonClustering. Figure 7 shows the greatest content delivery delays in general and for the distribution of all the blocks in various groups of content-requesting nodes in different multicast groups. When we used 6 content-requesting nodes, the delay time in the Mutualcast algorithm compared to that in the Clustercast and NonClustering proposed algorithms, was reduced by %29 and %9 respectively. And when 7 content-requesting nodes were used, the delay time in the Clustercast algorithm was reduced by 42 and 27 percent compared to that in Mutualcast and NonClustering algorithms respectively. And when 10 content-requesting algorithms were used, the delay time for the Clustercast algorithm, compared to that in Mutualcast and NonClustering was reduced by 40 and 21 percent's respectively. Improvement in the performance of the Clustercast algorithm is because of the fact that: 1-the nodes are clustered based on their distance from each other. 2-in each cluster, the deep structures are avoided in construction of m-ary trees. Figure 8 shows the delay in content distribution in general and when the duplicate trees are removed. The advantage of this reduction in delay time is that we can reduce the number of blocks sent from the source and in turn increase their size and minimize the delay in content distribution. As seen in Figure 8 the delay time in the proposed algorithm is less than that in other algorithms.
Stress is equal in Clustercast, NonClustering algorithms, but it's much greater in Mutualcast algorithm. With an increase in the number of nodes, the level of stress will increase accordingly. In The Clustercast, NonClustering, algorithms, have less stress compared to Mutualcast algorithm, which results from using shallow m-ary trees. In we have a very high stress which results from the fact that each content block is assigned to one node for distribution, and the node distributes the content among all the content-requesting nodes. Figure 9 shows the average stress in the evaluated algorithms.

Table 5: Comparison of total output in the assessed algorithms

Mutualcast output (kbps)			**NonClustering output (kbps)**			**Clustercast output (kbps)**			Source capacity **(kbps)**
The number of content requesting nodes			The number of content requesting nodes			The number of content requesting nodes			
6	7	10	6	7	10	6	7	10	
467	429	380	467	333	333	467	429	380	1000
633	571	480	633	571	468	633	571	480	2000
800	714	580	800	714	568	800	714	580	3000
967	857	680	967	857	668	967	857	680	4000
1133	1000	780	1133	1000	768	1133	1000	780	5000
1300	1143	880	1300	1143	868	1300	1143	880	6000

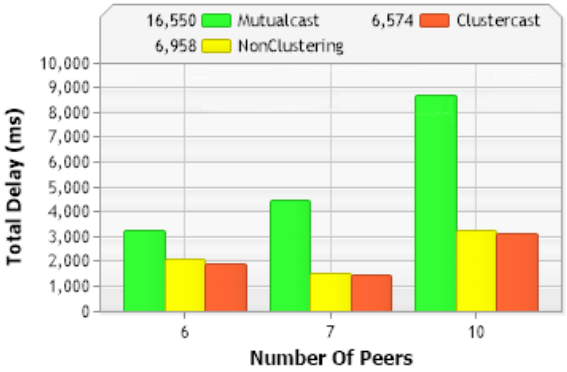

Figure 8: comparison of the greatest delay by removing the duplicate trees

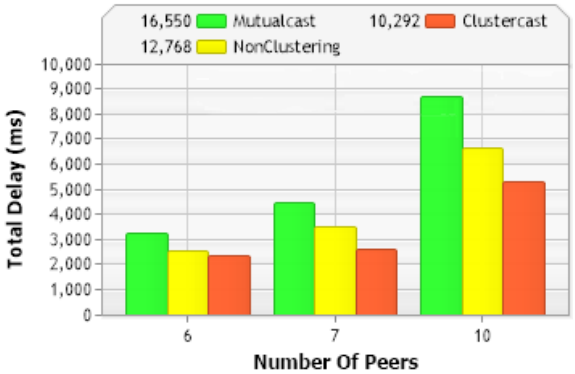

Figure 7: comparison of the greatest delays

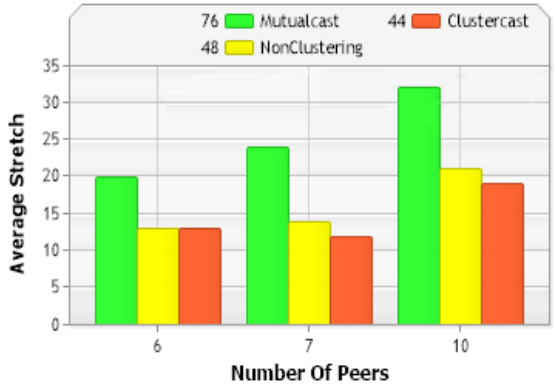

Figure 10: Comparison of average Stretch

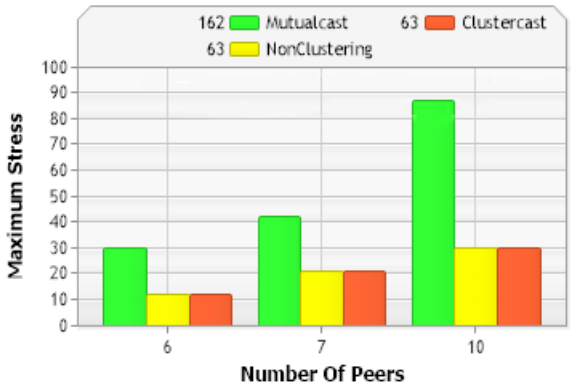

Figure 9: Comparison of average Stress

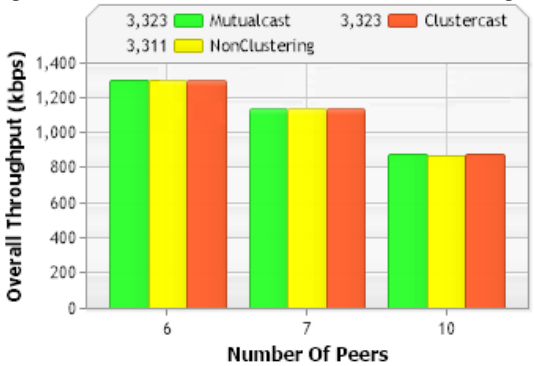

Figure 11: Comparison of overall output

Figure 10, shows the average of the highest Stretch in the evaluated algorithms. The level of Stretch in Clustercast and Mutualcast algorithms is the same when 6 content-requesting nodes are used in the multicast group. But the level of stress in NonClustering algorithms is less in that situation. However, when we used 7 and 10 content-

requesting nodes in the multicast group, the level of Stretch in the proposed algorithm was less than that in Mutualcast, NonClustering algorithms. In the Clustercast proposed algorithm, the level of stretch is lower than that in the other algorithms, and this is because of the fact that clustering and shallow m-ary trees are used in the Clustercast algorithm.

5. Conclusion

Finding a tree with maximum output and minimum delay in content distribution is vital for multicast sensitive activities. In this paper we present a new algorithm in the area of ALM, for multicast content distribution in peer to peer networks. This algorithm is called Clustercast which is very simple and very flexible for content distribution. This algorithm clusters the nodes that are close to one another and has little control overhead compared to similar algorithms. Here we compared Clustercast algorithm with Mutualcast, NonClustering algorithms. Mutualcast algorithm has the highest output, while the content distribution delay in this algorithm is very high. The NonClustering algorithm has also a very high output and content distribution delay in this algorithms is much lower than that in Mutualcast algorithm. In the Clustercast algorithm, each node with a fixed number of pings can find its position and connect to the multicast group. Clustercast clusters nodes using the information of their closeness and proximity, and minimizes content distribution delay by using a combination of highest loading capacity in nodes, and by constructing m-ary trees. In this algorithm, the values of Stress and Stretch in m-ary is significantly reduced compared to other algorithms, and this reduction results from node clustering. Our results show that the proposed algorithm provides a good balance between reductions of the delay in distribution of end-to-end content and maximum output performance. However, this algorithm maintains the scalability of the structure and avoids the full connection topology among the nodes.

6. Suggestions

The Extensiveness of multicast subjects in the research communities, still allows the interested researchers to carry out new research in this domain. Suggestions for further research in this domain include:

- ✓ Further investigation in order to find new ways of node clustering in which content distribution delay is minimized
- ✓ Further investigation in order to find a new tree structures in clusters with properties such as lower depth or less stress
- ✓ The impact of nodes with very low load capacity in construction of m-ary trees.
- ✓ Investigation of nodes with low download capacity in the multicast distribution

References

[1] S.Banerjee, B.Bhattachariee, C.Kommaredy; "Scalable Application Layer Multicast", In ProCeeding Of ACM Sigcomm, August 2002.

[2] M. Castro, P. Druschel, A-m. Kermarree, A. Rowstron, "SCRIBE: A Large Scale and Decentralized Application Level Multicast Infrastructure" IEEE Journal On Selected Areas In Communications (JSAC), Vol. 20, No.8.2002,PP.1489-1499.

[3] M. Kwon, S Fahmy ; "Topology-Aware Overlay Network For Group Communication", Miami Florida USA, Nossdav.2 May 12-13,2002.

[4] Allani, Mouna, Benoît Garbinato, and Peter Pietzuch. "Chams: Churn-aware overlay construction for media streaming." Peer-to-Peer Networking and Applications 5, no. 4 (2012): 412-427.

[5] Alkubeily, M., Bettahar, H., & Bouabdallah, A. A new Application-Level Multicast technique for stable, robust and efficient overlay tree construction. Computer Networks, 55(15), 3332–3350. (2011).

[6] M. Castro, P. Druschel, A-M. Kermarrec, A. Nandi, A. Rowstron and A. Singh, "SplitStream: High-bandwidth content distribution in a cooperative environment", In Proc. of the International Workshop on Peer-to-Peer Systems, Berkeley, CA, February,2003.

[7] D. Kostic, A. Rodriguez, J. Albrecht, A. Vahdat, "Bullet: High Bandwidth Data Dissemination Using an Overlay Mesh", 19th ACM Symposium on Operating Systems Principles, October 19-22, 2003, the Sagamore, New York.

[8] J. Lee and G. Veciana, "On application-level load balancing in FastReplica," Computer Communications, vol. 30, 2007.

[9] J. Li, P. A. Chou, C. Zhang, "Mutualcast: An Efficient Mechanism for One-To-Many Content Distribution", In Proc. of ACM SIGCOMM ASIA Workshop, April 2005.

[10] BitTorrent user activity traces. (2008) [Online]. Avaialable: http:// www.cs.brown.edu/ pavlo/torrent/

[11] 11Ramin Rajabioun ،Cuckoo Optimization Algorithm ،ELSEVIER،2011- Pages:5508:5518

[12] C. Semeria, T. Maufer; "Introduction to IP Multicast Routing", draft-ietf-mboned-intro- multicast- 03.txt.

[13] Francisco de Asís López-Fuentes, "Proximity-Aware Collaborative Multicast for Small P2P Communities" Parallel and Distributed Processing Symposium, 2007. IPDPS 2007. IEEE International