Supporting a multicast communication for large scale groups with a high degree of fault tolerance[*]

M. N. El-Derini, H. H. Aly and M. A. Yassin
Computers & Automatic Control Department, Alexandria University, Alexandria , Egypt
Elderini@ieee.org, alyh@computer.org, mohammed_a y@hotmail.com

For some applications, several processes may work together in groups and a sender needs to send messages to a specific group. Such a communication is called multicast communication and its routing algorithms are called multicast routing. Several multicast algorithms have been designed to handle the construction and communication of the groups. Most of the algorithms create for each multicast group, a single shared spanning tree which connects all members of the group. A multicast message is sent through the links of this tree to reach the entire group. In this paper the issue of the fault tolerance over multicast groups will be considered. A failure of a member (router) in a multicast group results in splitting the shared tree connecting that group into several subtrees, and hence a multicast message to that group can not reach all its members. Handling fault tolerance requires two objectives; the first objective is the desire to achieve a high degree of fault tolerance and if possible support unlimited degree of failure recovery. The second objective is the desire to reduce the overhead of storage and control messages required supporting the failure recovery. These objectives are contradicting and hence a compromise is required. Different algorithms have been designed to support different degrees of failure with different requirements and overhead in storage and control messages. In this paper a new algorithm is presented that supports unlimited degree of failure recovery over multicast groups with little information stored in each member.

في بعض التطبيقات، يمكن لمجموعة من الأنشطة أن تعمل معا في مجموعات و يريد المرسل إرسال الرسائل لجميع أفراد المجموعة. هذا النوع من التخاطب يطلق عليه البث الجماعي. و لقد تم تصميم العديد من الخوارزمات لبناء و للتخاطب مع مثل هذه المجموعات. و معظم هذه الخوارزمات تبنى شجرة واحدة تربط جميع أعضاء المجموعة معا، ويتم إرسال الرسائل لكل أعضاء المجموعة عن طريق أفرع هذه الشجرة. في هذا البحث سوف نهتم بالبث الجماعي ذو خاصية السماح بالعطل. في حالة عطل عضو من أعضاء المجموعة يحدث انقسام في الشجرة التي تربط أعضاء المجموعة معا مما يؤدي إلي فشل في عمل المجموعة ولن تستطيع الرسائل الموجهة لهذه المجموعة من الوصول لكل أعضائها. و معالجة خاصية السماح بالعطل تتطلب تحقيق هدفين أساسيين، أولهما هو تحقيق درجة عالية من السماح بالعطل و ثانيهما هو تقليل عبء الخوارزم من حيث كمية البيانات المخزنة وعدد الرسائل المستخدمة لتلافى العطل الموجود. في هذا البحث نقدم خوارزم جديد يسمح بتحقيق درجة عالية من السماح بالعطل مع وجود عبء بسيط للخوارزم المقترح.

Keywords: Multicast, Fault tolerance, Group manager, Shared based tree

1. Introduction

Supporting a multicast communication in a domain requires two main operations, which are the group construction and the group communication. A widely used algorithms which handle the multicasting are the shared based tree (SBT) algorithms [1-6]. The basic idea behind these algorithms is creating a shared spanning tree controlled by a special node called the "Core", and every member in a multicast group keeps a list of its links which belong to this tree. A multicast message is sent first to the core of the group which decapsulates the message and propagates it to all members over the tree links. Since all members of a multicast group are connected by a single tree, in which every member knows from where it receives a multicast message (in-link) and to where it sends it (out-links). A failure in one or more of the members divides the shared tree into several isolated subtrees, and a multicast message to that group will not reach all its members due to that failure. So,

[*] The publication is supported in part by the US-Egypt science and technology joint fund in cooperation with NSE under project ID # NF4-001-002.

a method is required to reconnect the isolated subtrees again and to recover from the failure. Fig. 1 shows a shared tree divided into several subtrees due to a failure occurred in a member of the group.

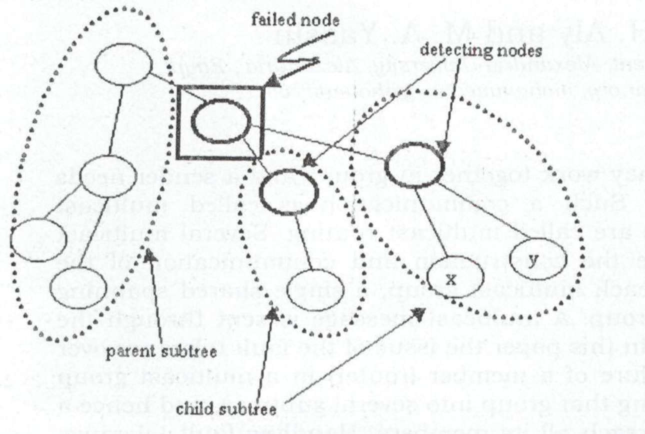


Fig. 1. A shared based tree divided into several subtrees.

Several algorithms have been implemented to handle the fault tolerance issue over multicast groups [6,7-10] such as the centralized algorithm, the self_recovery algorithm[10] and the CMMR failure recovery algorithm[6]. All of these algorithms try to exclude the failed member(s) from the group and to reconnect the isolated subtrees again.

The centralized algorithm provides an unlimited degree of failure recovery but it requires a large amount of storage at a centralized node to keep the complete structure of all groups in its domain.

The self_recovery algorithm also provides an unlimited degree of failure recovery and requires no storage, but the number of control messages required to recover from a failure is prohibitive, especially for large-scale groups.

The CMMR algorithm provides a limited degree of failure recovery (specified during a system installation) with limited requirements of storage at every member.

In this paper an efficient algorithm for handling the fault tolerance over multicast groups is presented. The proposed algorithm provides unlimited degree of failure recovery with small amount of required storage. The algorithm depends on giving every member in a group an identifier called "LVL_ID" which represents its height from the root member, and when a failure occurs this LVL_ID is used to help the members which detect the failure to reconnect with members that have LVL_ID less than their LVL_ID. This operation guarantees a correct reconnection and the recovery will not depend on a specific node.

2. Background and related work

The Self-Recovery algorithm is one of the popular methods for recovering from a node failure in the SBT algorithms because it does not require any additional storage. In the algorithm, the detecting member(s) (child members) send a message called "flush" message down stream to all its descendents which tears down the subtree to individual routers. Then every one of the descendants has to reconnect to the shared-tree on its own as if it is a new member. This is obviously a very expensive method because it needs a large number of messages specially when the failure occurs somewhere close to the root of the tree. It could be even more worse when the detecting members do not sense the failure at the same time which may cause some members to reconnect several times until successfully join the shared tree. For example, in fig. 1, if node "x" detects the failure before node "y" and tries to reconnect then node "y" may rejoin it. After a while node "y" may detect the failure so it sends to "x" a "flush" message to reconnect again. The advantage of this algorithm is that it does not need to store extra information about the group structure to support the fault tolerance, but when a failure occurs it costs a lot to recover from it.

In the Centralized algorithm[10] it specifies a centralized node which keeps the complete structure of the multicast group, when a failure occurs this node can be referenced to give the required information to recover from this failure. Since the centralized node knows the complete structure, it can specify who are the children of the failed node and sends a list of them to the parent of the failed member (the detecting node in this case) which sends a "join" message to every node in this list to reconnect the separated parts. This algorithm can recover from a single failure as well as multiple failures but multiple failures are recovered in multiple steps (every request from the detecting parent to the centralized node recovers one failure level). The performance of the centralized algorithm is better than the performance of the self-recovery algorithm because the number of messages required to recover from a failure is small, but its main disadvantage is the centralization and the massive amount of data stored in the centralized node. The centralized node is considered a special node and its failure is handled by the redundancy.

The CMMR failure recovery algorithm[6] limits the degree of supported failure. This limit should be known during the installation of the network because it specifies the amount of data exchanged between the routers during the group construction. If a single level of failure is supported then every member has to know its grandparent besides its parent. If two levels of failure are supported then every member has to know two grandparents besides its parent. The main disadvantages of CMMR failure algorithm is that the level of required fault tolerance must be predefined and any failure above this level can't be recovered. Also if many levels of failure are supported then a considerable amount of data is stored in each group member and the recovery requires several steps to handle the failure.

3. Proposed fault tolerance algorithm

Handling a failure in a multicast group requires a mean by which we can exclude the faulty member(s) from the group and reconnects the shared tree parts again. When a failure occurs in a member, the spanning tree connecting the group members is divided into isolated subtrees; fig. 1, which causes a failure in the group functionality and a multicast message to that group will not reach all its members. A failure in a multicast group is detected either by the parent of the faulty member or by the children of the faulty member. If the parent is the member who detects the failure then it tries to reconnect the children of the faulty node, but if the children are the members who detect the failure then they try to reconnect with the parent subtree. In the centralized algorithm the detecting member is the parent node whereas in the self-recovery algorithm and the CMMR failure recovery algorithm, the detecting members are the children nodes. Whether the detecting member is the parent or the child, an efficient fault tolerance algorithm should have the following properties:

- It is able to reconnect the separated subtrees again to form a single spanning tree that connects all members of the multicast group.
- The reconnection is done efficiently, so a small number of members should be responsible for the reconnection not the entire subtrees. Also the reconnection method must ensure that an isolated subtree will not be connected, by mistake, with another isolated part.
- The amount of data stored in the group members should be small and independent of the supported fault tolerance degree.
- Supporting a high degree of failure recovery, and if possible supporting unlimited degree of failure recovery.
- The Overall overhead of the algorithm until recovery is considerably small.

In the presented algorithm, the detecting members are the child members and the algorithm depends on giving every member in a multicast group a "LVL_ID" identifier which represents its location in its group; specifically its height in the group, this "LVL_ID" is only 3 bytes and helps the other members, in case of failure, to know the relative location of the faulty member and whether they can offer help or not.

3.1. Fault tolerance algorithm details

The implementation of the proposed algorithm is distributed over the different operations of the group as follow:

- *Constructing a new group:* For a domain to support multicasting, it specifies a special node called "Group Manager" (GM). This node keeps track of the existing groups in its domain and keeps a list "Group, Some-members-in-group" which allows the GM to reach the group whenever it needs to. When the first member in a domain wishes to join a group, it sends a "Group-Inquiry" message to the GM to join the group, the GM will reply by a "Group_Response" message to inform the sender to be the root of the group, then the

GM does two more steps to support the fault tolerance, first it assigns this member a "LVL_ID" value equals 1, second it adds this member to its (Group, some-members-in-group) list.

• *Joining a group:* Any new member will join the group by connecting it with an existing member either directly or indirectly by the help of the GM. The new member will be assigned a LVL_ID equals to the LVL_ID of the existing member plus one. A new member may send to the GM his information (his address, group address and his LVL_ID) and if the GM needs, it can add this member to its list. Also the GM has the ability to send a multicast message to a group asking for more information about the members in this group. Fig. 2 shows the shape of the shared tree after constructing the group.

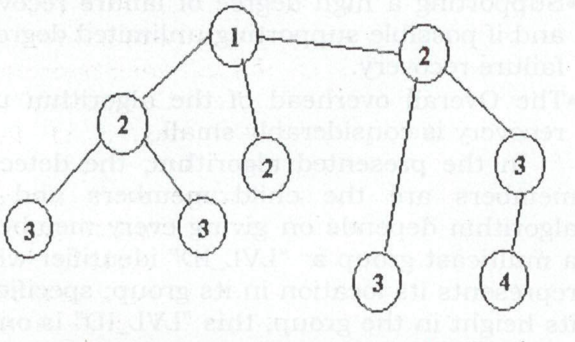


Fig. 2. A multicast group hierarchy with LVL_ID values.

• *Leaving a group:* For a member to leave its group it should be a leaf member with no end terminals or routers connected to it. In this case, there is no need for additional steps to support the fault tolerance. But if the leaving member still has tree branches, it sends to its parent a list of its children to reconnect them.

• *Single member failure:* when a single failure occurs to a member as shown in fig. 3, the children of the faulty member will detect this failure and do the following :

1. When a child member detects its parent failure, it sends a "Re-Join" message directed to the group manager (GM) containing its LVL_ID (Child_LVL_ID). This message is sent hop-by-hop toward the GM and only the members with LVL_ID less than Child_LVL_ID can hold the message and reply with a "Join-Ack" message to connect that child. This ensures that the member who will hold the "Re-Join" message belongs to the parent subtree and hence the connection is done correctly.

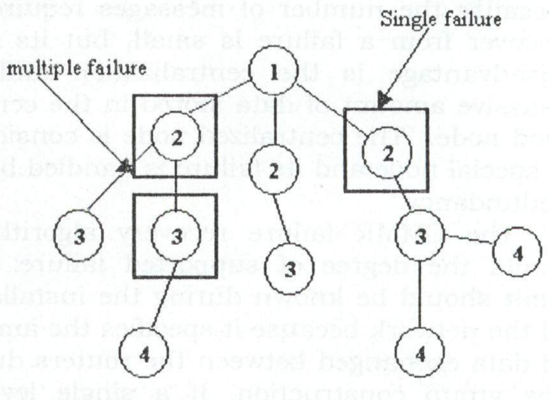


Fig. 3. Single or multiple failures.

2. If the message meets in its way to the GM a member with LVL_ID < Child_LVL_ID then this member will hold the message and send a "Join-Ack" message with its LVL_ID plus 1 to the sender to reconnect it with the group again.

3. If the message does not meet any member with LVL_ID < Child_LVL_ID then it reaches the GM which will search in its (Group, some-members-in-group) list for a member with identifier LVL_ID < Child_LVL_ID and considers this member a candidate member (c_member).

4. The GM sends to c_member a "Rdy-Join" message to be sure that it is alive and its LVL_ID is not changed and it is ready to reconnect the isolated sender, then the GM sets a timer "Rdy-Join-Timer".

5. When the c_member receives the "Rdy-Join" message it sends two messages, one to the GM telling it that it is alive and whether it accepts the job or not, this message is called "Rdy-Join-Ack" and when the GM receives this message it resets its "Rdy-Join-Timer". The second message is sent in case of the acceptance and it is directed to the original sender to reconnect it. This message is a normal "Join-Ack" message with its LVL_ID plus 1. The c_member knows the address of the original sender from the "Rdy-Join" message sent from the GM. If the LVL_ID of the c_member is changed then it sends the new value to the GM in the "Rdy-Join-Ack".

6. If the GM does not receive a "Rdy-Join-Ack" before the "Rdy-Join-Timer" timer expires, it concludes that this c_member is failed so it removes it from the (Group, some-members-in-group) list and tries to find another member.
7. When the original sender receives a "Join-Ack" message with its new LVL_ID it sends to its children a "Chng-LVL-ID" message with their new LVL_ID. This "Chng-LVL-ID" message propagates from a member to its children until it reaches the leaf members and at each level LVL_ID value is incremented by 1.
8. For the GM to update its list periodically, it sends a message to the members in its list asking them for any changes in their LVL_IDs. If the faulty member has many children then every child do the same to reconnect itself and the rest of its subtree.

• *Multiple members failure:* The handling for multiple failures is exactly the same as handling a single failure. This is why the algorithm supports unlimited degree of fault tolerance without extra overhead.

• *Failure of the root member:* The failure of the root member requires a special handling as follow :
1) When the GM receives a "Re-Join" message with LVL_ID = 2 asking for reconnection, it deduces that the root member (member with LVL_ID =1) failed and another member should be selected to be the root of the group.
2) The GM sends to the requesting member a "Group_Response" message including its address (which means that it does not need to connect with other members) with LVL_ID = 1 indicating that this member becomes the new root for the group.
3) After the child member (which becomes the root) sets its LV_ID to 1 it sends a "Chng-ID" message to all its subtrees to change their LVL_ID.
4) If the GM receives many "Re-Join" messages simultaneously from the child members, it selects one member to be the new root for the group; takes LVL_ID =1 as described above; then it selects this member to be the candidate member (c_member) to reconnect the other children. The selection can be done easily by selecting the requester of the first "Re-Join" message received to be the new root of the group.

• *Failure of the GM:* The failure of the GM is handled by the redundancy.

4. Fault tolerance algorithm evaluation

The following points are the main differences between the proposed algorithm and the previous algorithms:

a) Minimum storage: In the "Centralized" algorithm there is a node that keeps the complete hierarchy of all groups in its domain which needs a massive amount of data specially when the group size is large, also it stores the address of every group member and that address, in IPv6 protocol, is 16 bytes long. In the CMMR failure technique if *'n'* levels of failure recovery are supported then (*n * 16)* bytes are required at each member. In the proposed algorithm only (3 bytes) are required as a LVL_ID identifier at every group member to support any degree of failure.

b) Unlimited degree of fault tolerance: In the previous algorithms like "Centralized" and "Self-Recovery" unlimited degree of failure recovery is supported but with major drawbacks either in storage or in messages number. In "CMMR" an unlimited degree of failure recovery is not supported. In the proposed algorithm, unlimited degree of failure recovery is supported with a small amount of stored data but it also has drawbacks, which are its requirement for a hop_by_hop transmission and the adjustment of the LVL_ID values after the recovery. But the simulation results show that the overhead of the proposed algorithm is less significant than the overhead of the previous algorithms.

c) Scalability: Since the proposed algorithm supports unlimited number of fault tolerance with small amount of stored data that does not depend on the failure degree, then the algorithm works well in case of large scale multicast groups. Also the scalability increases since the algorithm allows any group member with LVL_ID less than the faulty member LVL_ID to rejoin the isolated subtrees.

d) Efficiency of handling multiple failures: Even if the multiple failures are handled in the previous algorithms, it is not recovered directly (in one step). For example, In CMMR and centralized algorithms, If there are *'n'* failures then it requires *'n'* steps to recover from them. But in the proposed algorithm only one step is required to recover from multiple failures.

e) Adding extra overhead after reconnecting the isolated subtrees: In the centralized algorithm the overhead after the recovery is represented by informing the centralized router by the changes done in the group structure. In CMMR and self-recovery algorithms there is no extra overhead after reconnecting the isolated subtrees. In the proposed algorithm there is an extra overhead after reconnecting the isolated subtrees which is required to adjust the LVL_ID values after the reconnection, so we need to send a message to all members in the isolated subtrees. But this overhead is also required in CMMR and self-recovery algorithms but not after the reconnection, it is a necessary step to reconnect the isolated subtrees.

f) Time required to recover from a failure: All the previous algorithms use the direct transmission to recover from a failure. In the proposed algorithm, part of it depends on a hop_by_hop transmission to recover from a failure. This hop_by_hop transmission may increase the time required to recover from the failure, but on the other hand it tries to reduce the number of hops passed by the messages.

g) Performance measures boundary: The required time to recover from a failure in the centralized and CMMR algorithms is unbounded because the recovery operation requires *'n'* steps done in serial to recover from a failure of degree *'n'*. But in the proposed algorithm, only one step is required and it is done completely in parallel, so the required time has an upper bound which is sending one message to GM and receiving the reply. Also the required storage in the CMMR algorithm is unbounded as it depends on the supported failure degree, but the required storage in the proposed algorithm is bounded and it is independent of the failure degree.

Table 1. shows a comparison among the previous fault tolerance algorithms and the proposed one.

4.1. Simulation model

The Timed Colored Petri Net (TCPN)[20] is used to model the algorithms. Figs. 4, 5 and 6 show the TCPN model for the proposed, centralized and the self-recovery algorithms.

In the models, all routers are represented as places and all messages are represented as tokens.

Fig. 4. shows the model for the proposed algorithm. The scenario of the simulation is as follow, when a failure occurs token K_1 is moved from the source place and placed in one or more of the group members (according to the failure degree) by transition T_0. The failure is detected by the child members of faulty member and start sending "Re-Join" messages to rejoin the group. These messages are sent hop by hop to GM. The sending of the "Re-Join" messages is represented by transition (T_1), which generates new tokens (K_2), which represents the "Re-Join" messages. During the movement of the "Re-Join" messages it may pass one of three options.

The first option is a normal router or group member with LVL_ID >= message LVL_ID (represented as place P_3) this option is just re-send the message to the next hop toward the GM (represented as transition T_3).

The second option is a group member with LVL_ID < message LVL_ID (represented as place P_4) this option will hold the message and sends a "Join-Ack" message to reconnect the original sender and move to the "after reconnecting the subtrees" part (represented as transition T_4). The third option is a GM (represented as place P_2) which holds the "Re-Join" message (Token K_2) and selects a member with LVL_ID < message LVL_ID to rejoin the original sender and sends to it a "Rdy-Join" message (represented as token K_3). So when K_2 or K_3 reaches to a group member with LVL_ID < message LVL_ID (place P4) we transfer to "after reconnecting the subtrees" part.

The formal structure of TCPN model is as follow:

a) Structure

N = (T, P, L, I, O, F), where

T is the set of transitions, P is the set of places, L is the set of links, I is the set of input

Table 1
A comparison between different failure algorithms

		Centralized	CMMR	Self-Recovery	Proposed
Storage	Location	Single centralized node	Every group member	----------------	Every group member
	Size	Complete group hierarchy	Depends on the level of recovery required, every level needs 16 bytes	No storage needed	3 bytes are needed for LVL_ID identifier
Detecting node		Parent node	Child nodes	Child nodes	Child nodes
Recovering responsibility		Only parent is responsible to reconnect isolated subtress.	Each detecting child is responsible for reconnecting its subtree.	Every node in the isolated subtrees rejoin again by itself	Each detecting child is responsible for reconnecting its subtree.
Degree of fault tolerance		Unlimited	Predefined	Unlimited	Unlimited
No. of steps required to recover from n failure		n steps	n steps (only if supported)	1 step. But every node may rejoin several times.	1 step.
Maintained Lists		The centralized node keeps the whole structure of the group.	The CM maintains a (Group, Cores) list and a (Candidate Cores) List	The Core maintains a list containing a brief view about the group.	The GM maintains a list containing a brief view about the group
Handling a leaf member failure		Requires overhead as any other member	----------------	----------------	-----------------
Message transmission		Direct	Direct	Direct	Hop_by_hop and Direct

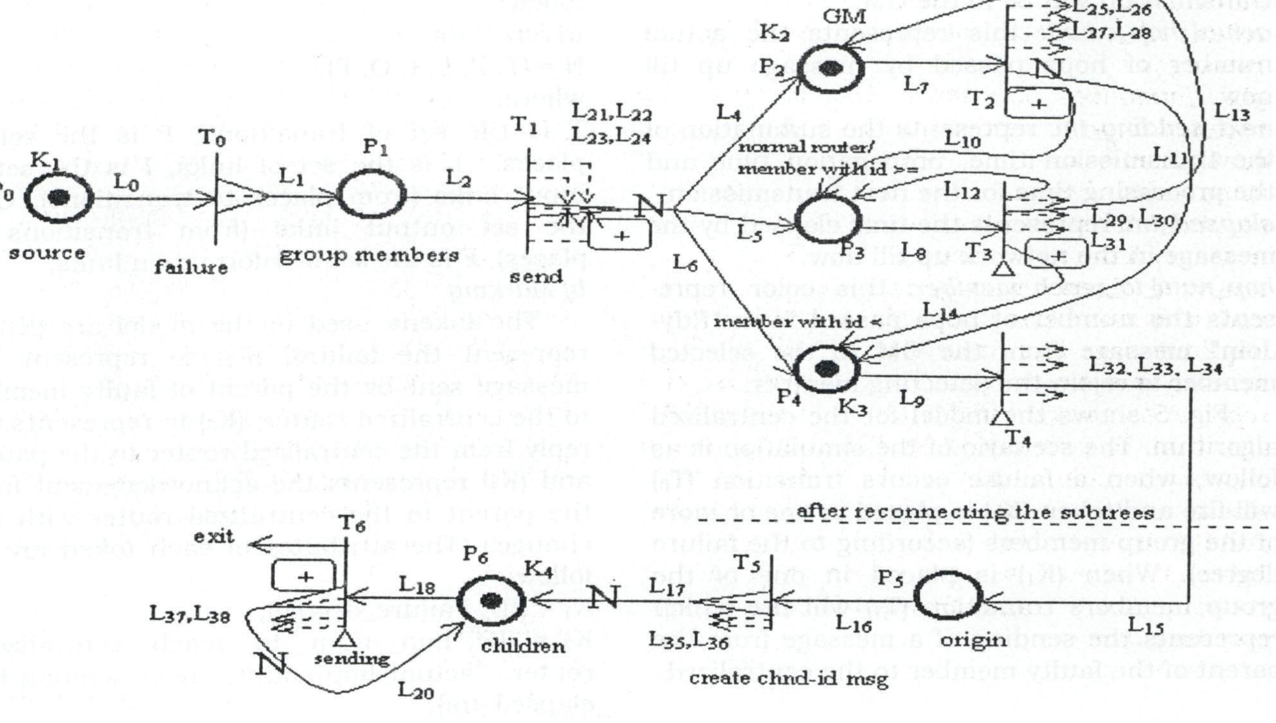


Fig. 4. TCPN model for the proposed algorithm.

links (from places to transitions), O is the set output links (from transitions to places), F is the set of information links.
b) Marking
The tokens used in this model are (K_1) to represent failure, (K_2) to represent the "Re-Join" messages, (K_3) to represent "Rdy-Join" messages and token (K_4) to represent the "Chng-LVL-ID" messages with the following attributes:

K_1 = (ID, Failure_degree).
K_2=(ID, hop_num_to_reach_GM, ctual_ hop_ count, next_sending_tm, elapsed_tm)
K_3=(ID, hop_num_to_reach_member, actual_ hop_ count, next_sending_tm, elapsed_tm)
K_4 = (ID, next_sending_tm, elapsed_tm).

Where:
ID: is the message identifier
Failure_degree: represents the failure degree.
hop_num_to_reach_GM: since the message is sent hop-by-hop to the GM then it ultimately must reach the GM. This is specified in our model by a color "hop_num_to_reach_GM" which will be assigned a value before the first transmission of the message, and if actual hop count $K_2(3)$ reaches this value then the next transmission will be to the GM.
actual_hop_count: this represents the actual number of hops passed by message up till now.
next_sending_tm: represents the summation of the transmission time, propagation time and the processing time for the next transmission.
elapsed_tm: represents the time elapsed by the message in the network up till now.
hop_num_to_reach_member: this color represents the number of hops passed by a "Rdy-Join" message from the GM to the selected member to rejoin the detecting member.

Fig. 5 shows the model for the centralized algorithm. The scenario of the simulation is as follow, when a failure occurs transition (T_0) will fire and token (K_1) is placed in one or more of the group members (according to the failure degree). When (K_1) is placed in one of the group members transition (T_1) will fire which represents the sending of a message from the parent of the faulty member to the centralized router asking for the children of the faulty member, the message is represented as token (K_2). The existence of token (K_2) in place (P_3) will fire transition (T_3) only, which transmits it to either place (P_2) or place (P_3) according to a specific condition (color in K_2). These transmissions represent the message in its way to the centralized node. When (K_2) reaches the centralized node (represented as place P_2) it holds the token and sends its reply to the parent member, this is represented as a new token (K_3). The existence of token (K_3) in place (P_3) will fire transition (T_4) only, which transmits it to either place (P_3) or place (P_4) according to a specific condition (color in K_3). These transmissions represent the message in its way from the centralized node to the parent member. When token (K_3) reaches place (P_4) it reconnects the children of the faulty member and enters the "after reconnecting the subtrees" part, in which the parent member sends an acknowledgment to the centralized router with the changes done in the group hierarchy, this is represented as token (K_4) transmitted by transitions (T_5) and (T_6) until reaches the centralized router.

The formal structure of TCPN model is as follow:
a) Structure
N = (T, P, L, I, O, F),
where:
T is the set of transitions; P is the set of places, L is the set of links, I is the set of input links (from places to transitions), O is the set output links (from transitions to places), F is the set of information links.
b) Marking
The tokens used in the model are (K_1) to represent the failure, (K_2) to represent the message sent by the parent of faulty member to the centralized router, (K_3) to represents the reply from the centralized router to the parent and (K_4) represents the acknowledgment from the parent to the centralized router with the changes. The attributes of each token are as follow:
K_1 = (ID, Failure_degree),
K_2 = (ID, hop_ num_ to_ reach_ centralized_ router, actual_hop_count, next_sending_tm, elapsed_tm),

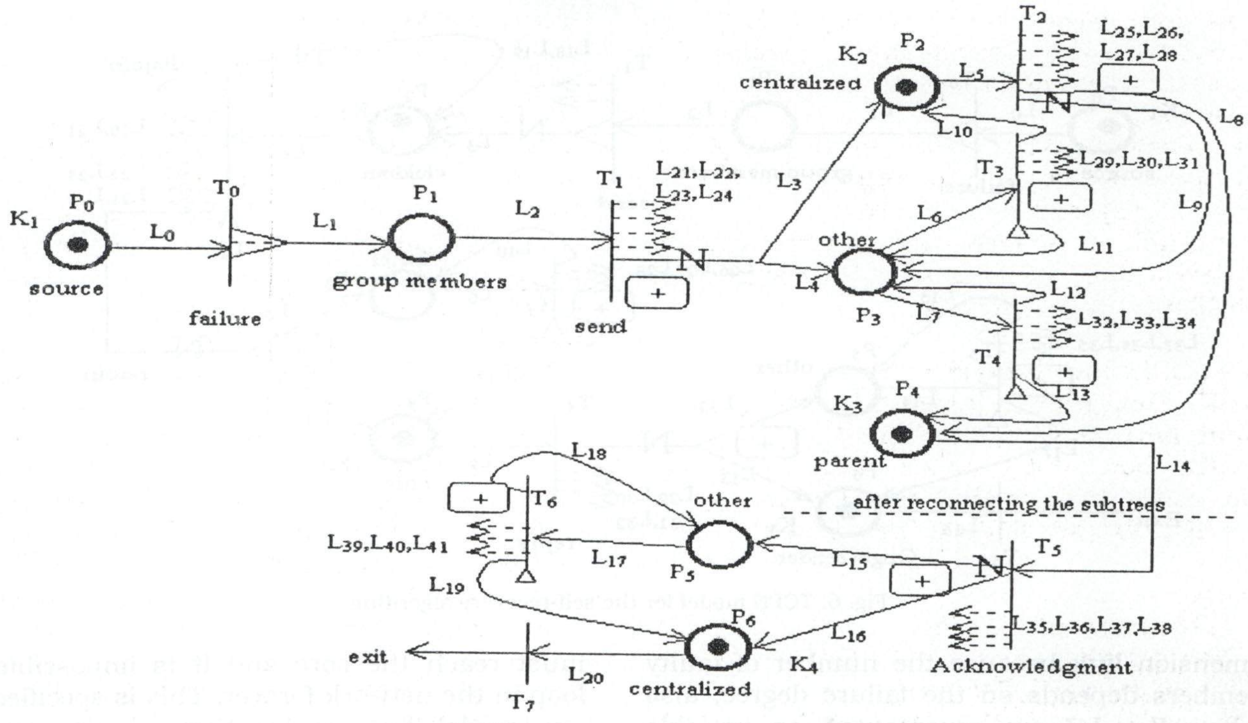


Fig. 5. TCPN model for the centralized algorithm.

K3 = (ID, hop_ num_ to_ reach_ parent_ member, actual_ hop_ count, next_ sending_ tm, elapsed_tm),
K4 = (ID, hop_ num_ to_ reach_ centralized_ router, actual_hop_count, next_sending_tm, elapsed_tm).
Where:
ID: is the message identifier
failure_degree: represents the failure degree.
hop_num_to_reach_centralized_router: since the message is sent directly to the centralized router it ultimately must reach it. This is specified in our model by "hop_num_to_reach_centralized_router" color.
actual_hop_count: this represent number of hops passed by message up to now.
hop_num_to_reach_parent_member: this color represents the number of hops passed by the reply from the centralized node to the parent node.
next_sending_tm: represents the summation of the transmission time, propagation time and the processing time for the next transmission.
elapsed_tm: represents the time elapsed by the message in the network up till now.

Fig. 6. shows the model for the self-recovery algorithm. The scenario of the simulation is as follow, when a failure occurs transition (T$_0$) will fire and token (K$_1$) is moved to one or more of the group members (according to the failure degree), then the children of the faulty member detect the failure and start sending disjoin messages (represented as token K$_2$) to their children if any, so transition (T$_1$) generates a new token (K$_2$) which propagates form a member to its children until reaches the leaf members. Every member which receives a disjoin message sends a rejoin message (represented as token K$_3$) to the core of the group to rejoin again, so transition (T$_2$) generates a new token (K$_3$) this token is transmitted in the network by transition (T$_3$) until reaches the core of the group. When the token reaches the core (P$_4$) it sends a "Join-Ack" message as a reply to the original sender (represented as token K$_4$), so transition (T$_4$) generates a new token (K$_4$) which is transmitted in the network by transition (T$_5$) until reaches the original sender. Link (L$_1$) is represented as variable

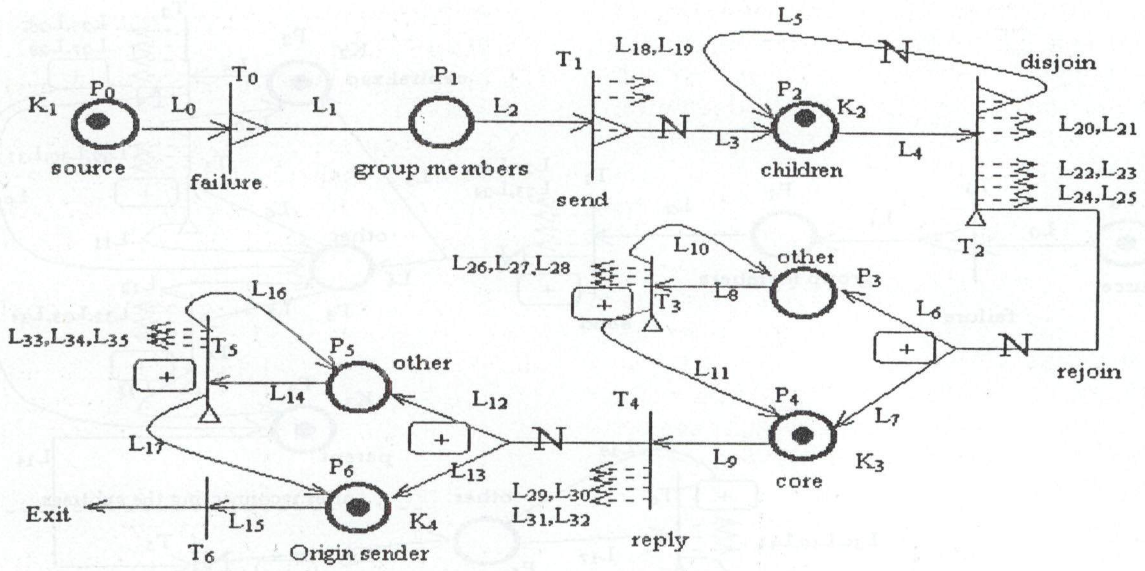


Fig. 6. TCPN model for the self-recovery algorithm.

dimension link because the number of faulty members depends on the failure degree, also links (L_3, L_5) are represented as variable dimension links because the number of children changes from node to node.

The formal structure of TCPN model is as follows:

a) Structure

N = (T, P, L, I, O, F), where:

T is the set of transitions, P is the set of places, L is the set of links, I is the set of input links (from places to transitions), O is the set output links (from transitions to places), F is the set of information links.

b) Marking

The tokens used in this model are (K_1) to represent failure, (K_2) to represent the "disjoin" messages, (K_3) to represent "rejoin" messages and token (K_4) to represent the "reply" messages with the following attributes:

K1 = (ID, Failure_degree).

K_2 =(ID, next_sending_tm, elapsed_tm)

K_3 = (ID, hop_num_to_reach_core, actual_hop_count, next_sending_tm, elapsed_tm)

K_4 = (ID, hop_ num_to_ reach_ sender, actual_ hop_ count, next_ sending_ tm, elapsed_ tm)

Where:

ID: is the message identifier

Failure_degree: represents the failure degree which will be simulated.

hop_num_to_reach_core: since the message is sent to the group's core then it ultimately must reach the core and it is impossible to loop in the network forever. This is specified in our model by a color "hop_num_to_reach_core" which will be assigned a value before the first transmission of the message, and if actual hop count $K_3(3)$ reaches this value then the next transmission will be to the core.

actual_hop_count: this represents the actual number of hops passed by message up till now.

next_sending_tm: represents the summation of the transmission time, propagation time and the processing time for the next transmission.

elapsed_tm: represents the time elapsed by the message in the network up till now.

hop_num_to_reach_sender: when the core receives a "Re-Join" message it replies with a "Join-Ack" message which will reach the original sender in a limited hop count. This is presented in our model by the "hop_num_to_reach_sender" color which will be assigned a value before the first transmission of the message (token K_4) and when actual hop count $K_4(3)$ reaches this value the next transmission will be to the original sender.

4.2. Simulation workload

The workload and the distributions used for the simulation purpose are:

i) Ave. Node Degree (D): In real networks, with small or moderate sizes, this parameter ranges approximately between 3.5 or 5. The parameter has Binomial (T, P) distribution, where T is the maximum number of children for a member in the group.

ii) Failed Node (F): This parameter represents the faulty member. Its distribution is U [1... G].

iii) Number of hops to reach GM, Centralized node or the Core (H): This parameter represents the maximum number of hops that a message requires to reach the GM (in case of the proposed protocol), the centralized node (in case of the centralized algorithm) or the core (in case of the self-recovery algorithm). This parameter has a geometric (P) distribution.

iv) Prob. That "Re-Join" message meets in its way to GM member with LVL_ID < message-LVL_ID (Pr): This parameter is used only in the proposed protocol and it depends on the group size and the location of failed member in the group. Its distribution is Bernoulli (P).

v) Transmission Time (Tt): In real WANs which use an ATM structure with data rate 155.52 or 622.08 Mbps the mean transmission time is approximately (2 μsec). The parameter has a Gamma ($\alpha=2, \beta=1$) distribution.

vii) Propagation Time (Tg): In real WANs with optical fiber and 10^3 km between routers the mean propagation time is about (3 msec). The parameter has a Gamma ($\alpha=3, \beta=1$) distribution.

vii) Processing Time (Tp): Assuming real processors with MIPS, the mean processing time, in case of direct transmission, is about (13 μsec) including the interrupt time. But in case of hop-by-hop transmission the mean processing time increases to approximately (1.6 msec). The parameter has a Gamma ($\alpha=13, \beta=1$) distribution in case of direct transmission and a Gamma ($\alpha=1.6, \beta=1$) in case of hop-by-hop transmission.

viii) Rdy_Join_Timer (T_j): This timer has a Gamma ($\alpha=7, \beta=1$) distribution with mean (7 msec).

4.3. Simulation parameters and results

The performance measures used in the simulation are:

- Number of messages required to recover from a failure.
- Total hop count required to recover from a failure.
- Time required to recover from a failure.

Note: In measuring the time factor we have assumed the parallelism whenever possible.

The simulation is done over the following parameters:

i- Network size (N): Simulation is done over a network size of 1500 node.

ii- Group size (G): Simulation is done over different group sizes. The group size is taken to be 50, 100, 150, 200, 250, 300, 350, and 400.

iii- Group structure (Gs): The simulation is run over different group structures for the same group size. Exactly 10 simulation runs with 10 different group structures are done for each group size (G) and the average is taken.

iv-Failure Degree (Fd): This parameter represents the degree of the failure. The parameter takes values 1, 2 and 4.

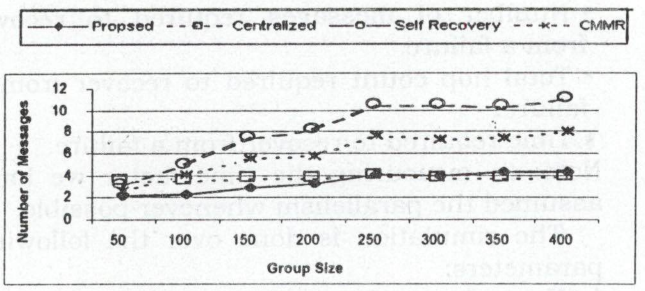
v- Failed Node (F): For each group structure 20 points are simulated and the average is taken.

The results of the simulation are shown in Figs. 7 through 9.

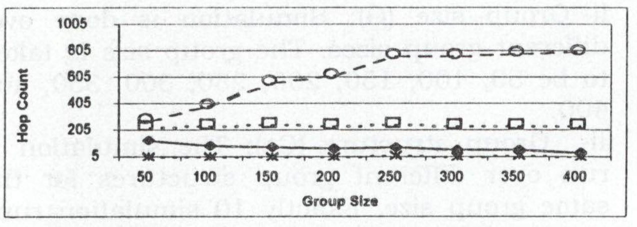
Figs. 7-a, b and c show the results in case of failure degree equals 1.

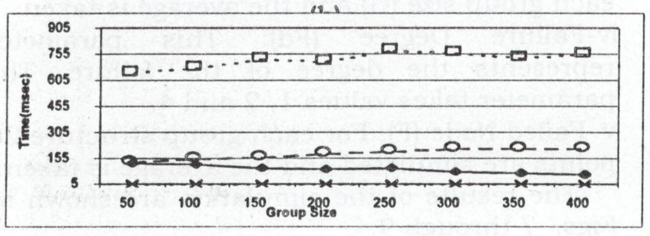
Fig. 7-a shows the number of messages required to recover from a single failure in the group. It is clear that the self-recovery algorithm has the largest number of messages due to the messages required to disjoin the complete subtrees and then every member in these subtrees has to rejoin again by itself. The overhead of the CMMR algorithm is mainly due to the messages sent to the entire subtrees twice, one for changing the mode of the members before the direct children of the faulty member send the "join-request" messages, and the other is after the rejoin is done.

The overhead of the proposed algorithm is little because the sending of messages to the entire subtree is done only once to change the LVL_ID values. The curves of the previous algorithms increase with the increase of the group size because the size of the subtrees increases, whereas the curve representing the



(a)





(c)

Fig. 7. The overhead in case of failure degree=1.

centralized algorithm is approximately fixed because the centralized algorithm does not send any messages to the isolated subtrees. The proposed algorithm converges with the centralized algorithm and may exceed it with the increase of the group size because the centralized algorithm encounters an overhead in handling the leaf members (this overhead does not exist in the other algorithms, as the detecting members are the child members). This overhead, in case of small groups, is large compared with sending one message to the isolated subtrees. But as the group size increases the overhead of sending a message to the subtrees increases.

Fig. 7-b shows the total hop count required by all messages sent to recover from a failure. Since the number of messages in the self-recovery algorithm is very large and many of them are sent to the core to rejoin the members then the total hop count of the algorithm is also very large compared with the other algorithms. Although the required number of messages in the CMMR algorithm is also high, but most of these messages pass a single hop to change or return the mode of the isolated subtrees, and only the direct children try to rejoin with their grandparent which also encounters a very little overhead. The curve representing the CMMR increases slightly as the group size increases. In the centralized algorithm, since the number of messages is approximately fixed and most of these messages are sent to the centralized node which does not depend on the group size then the total hop count is approximately fixed also. In the proposed algorithm, although the number of messages increases as the group size increases, the total hop decreases as the group size increases. This is because the chance that a message finds a member with LVL_ID less than the detecting member LVL_ID increases as the group size increases, which reduces the hop count encountered by each message. And as the group size increases the proposed algorithm converges to the CMMR algorithm.

Fig. 7-c shows the total time required to recover from a failure.

In measuring the time factor the parallelism is assumed whenever possible. The time required by the centralized algorithm is the largest because most of the operations to recover from a failure are done serially, contradicting with the other algorithms which may include many messages and hops but their operations are done in parallel. For example, all members in the isolated subtrees may send their messages to rejoin in parallel, so the required time is calculating of a single message. The required time encountered by the proposed algorithm decreases as the group size increases for the same reason in case of the hop count. In the CMMR, since the direct children will join with their grandparent then the required time to recover from a failure is very small. The curve increases slightly as the group size increases.

Figs. 8-a, b and c show the results in case of failure degree equals 2.

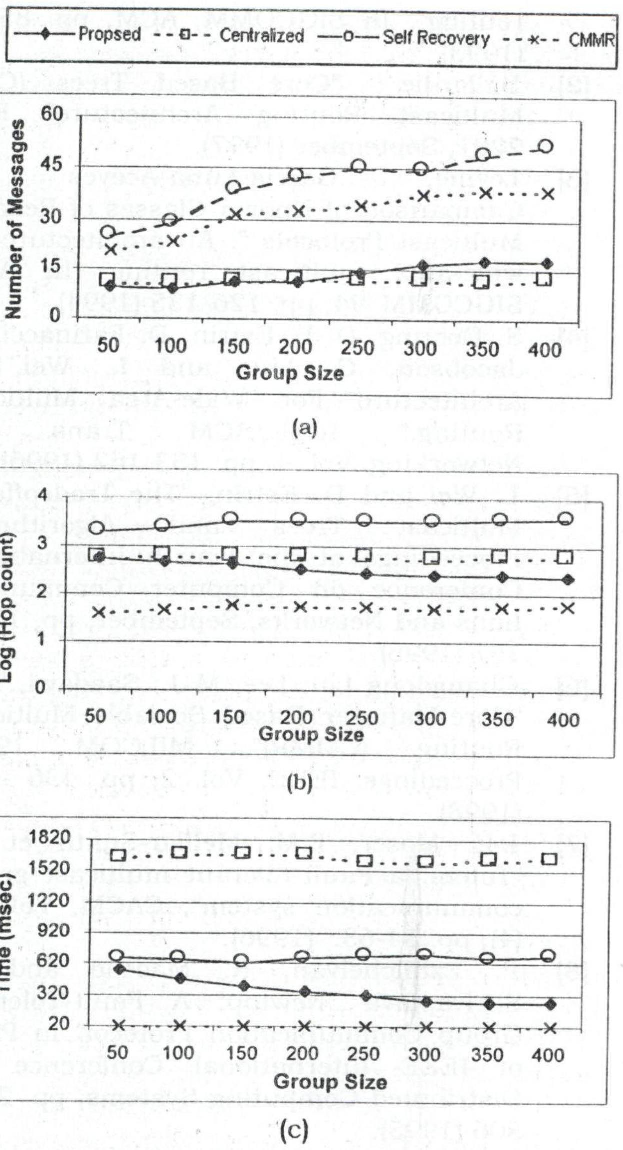


Fig. 8. The overhead in case of failure degree=2.

Fig. 8-a shows the required number of messages to recover from a failure degree equals 2. The relative location of the curves is approximately the same as in fig. 7-a, but the number of messages and the gap between the curves increases because the size of the isolated subtrees increases. Also the increase in the CMMR and the centralized algorithm is more rapidly than the other algorithms because the recovery in the former algorithms requires 2 steps not one as it is in the other algorithms.

Fig. 8-b shows the required hop count by all messages to recover from failure degree equals 2. Also the increase of the CMMR and the centralized algorithms is more rapidly than it is in the other algorithms.

The simulation results show that the performance of the proposed algorithm is better than the performance of the other algorithms which allow unlimited degree of failure (centralized and self-recovery algorithms). Also although the CMMR algorithm saves in the required time and the total hop count by specifying a special node for the members to rejoin with (grandparent), it pays a large cost of limiting the fault tolerance degree, and any failure above this degree can not be recovered and its requirements in the number of messages and the required storage is higher than those in the proposed algorithm.

Fig. 9-a and b show the required storage for the different algorithms.

Figs. 9-a and b show that the storage required by the CMMR is a linear function of the number of the supported fault tolerance degrees because the storage kept in every member depends on the supported degree. Whereas the required storage in case of the proposed algorithm and the centralized algorithm does not depend on the supported degree.

In the proposed algorithm every member keeps 3 bytes only as a LVL_ID value, and in the centralized algorithm the complete structure is kept in a centralized node. The self-recovery algorithm does not need any extra storage to support the fault tolerance issue.

Figs. 9-a and b show that the required storage by the proposed algorithm is very small compared with the other algorithms.

5. Summary and conclusions

In this paper a new fault tolerance algorithm over multicast groups is presented. The algorithm provides unlimited degree of failure recovery with a small amount of the required storage. Several algorithms have been designed to handle the issue of the fault tolerance over multicast groups such as the centralized algorithm, the self-recovery algorithm and the CMMR algorithm, but there is a major drawback in every one of these

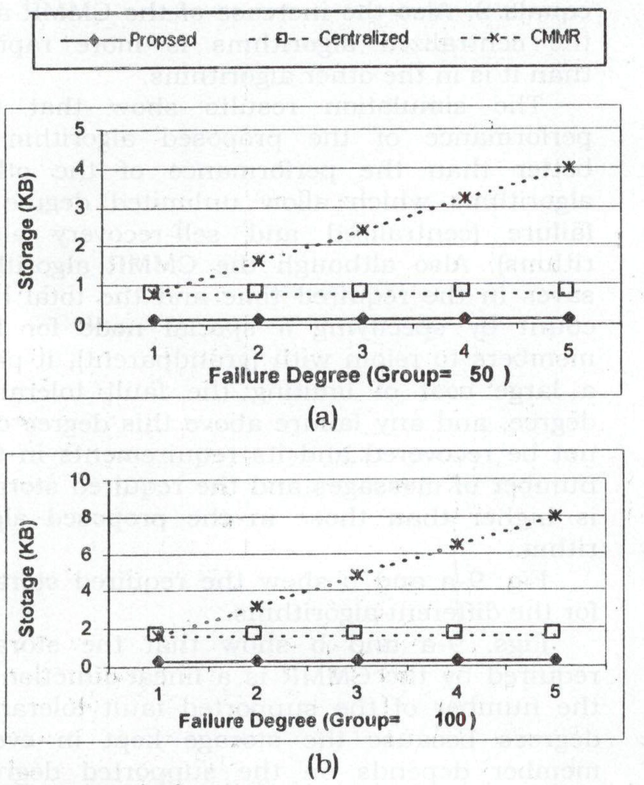


Fig. 9. Required storage.

algorithms which makes it inefficient specially in case of large groups. In the centralized algorithm, the centralization and the massive storage are the main drawbacks. In the self-recovery algorithm, the huge number of control messages is the main drawback. In the CMMR algorithm, the limitation of the supported failure degree and the huge storage are the main drawbacks. The proposed algorithm assigns every member in a group a "LVL_ID" identifier which represents its height in the group, when a failure occurs this identifier is used to specify which members can rejoin the isolated subtrees. The proposed algorithm offers a better performance, scalability and saves in storage compared with the other algorithms.

References

[1] T. Ballardie, P. Francis, and J. Crowcroft. "Core based trees (CBT): an architecture for scalable inter domain multicast routing". In SIGCOMM, ACM, pp. 85-95 (1993).

[2] Ballardie "Core Based Trees (CBT) Multicast Routing Architecture" RFC 2201, September (1997).

[3] Levine, Garcia-Luna-Aceves "A Comparison of Known Classes of Reliable Multicast Protocols ". An architecture for wide-area multicast routing. In ACM SIGCOMM '94, pp. 126-135 (1994).

[4] S. Deering, D. L. Estrin, D. Farinacci, V. Jacobson, C. Liu, and L. Wei,"PIM Architecture For Wide-Area Multicast Routing," IEEE/ACM Trans. on Networking, Vol. 4, pp. 153-162 (1996).

[5] L. Wei and D. Estrin, "The Tradeoffs of Multicast Trees and Algorithms," Proceedings of the Fourth International Conference on Computer Communications and Networks, September, pp. 150 - 157 (1995).

[6] Changdong Liu; Lee, M.J.; Saadawi, T.N. "Core-Manager Based Scalable Multicast Routing (CMMR)", MILCOM 1998. Proceedings. IEEE, Vol. 2, pp. 436 -441 (1998).

[7] L.E. Moser, P.M. Melliar-Smith et al, "Totem: a Fault-tolerant multicast group communication system", CACM, Vol. 39 (4), pp. 54-63. (1996).

[8] P. Ezhilchelvan, R. Macedo and S. Shrivastava "Newtop: A Fault-Tolerant Group Communication Protocol" in Proc. of IEEE International Conference on Distributed Computing Systems, pp. 296-306 (1995).

[9] Papadopoulos C., Parulkar G., and Varghese G. "An Error Control Scheme for Large-Scale Multicast Applications". Proc. of IEEE Infocom, pp. 1188-1196 (1998).

[10] H. Omar, T. Saadawi and M. Lee, "Supporting Reduced Location Management Overhead and Fault Tolerance in Mobile-IP Systems", ISCC, 6 - 8 July Egypt (1999).

Received March 2, 2002
Accepted June 11, 2002