Role and Responsibility of Membership Service in Group Communication

Rajeev Ranjan Kumar Tripathi[1],Rajesh Kumar Singh[2], Shrawan Kumar Pandey[3], Rakesh Kumar Pandey[4]
Institute of Technology and Management GIDA, Gorakhpur1,3
KIPM-College of Engineering, GIDA, Gorakhpur2,4

Abstract— **There are various circumstances when a common message is to be delivered to many recipients by an application. In this scenario, instead of passing this message to one by one receiving processes, we generally create a group of recipients. In Distributed System terminology, it is referred as Group Communication. An application which is supporting the concept of Group Communication may allow creating various groups. One process may belong to many groups. In an application, size of group may be static or dynamic. Group may be dynamically created as it appears now a day in social networking sites and application. A process can leave or join a group at any time. To reduce the degree of coupling existing in between sender of message and receivers we deploy an additional layer which is referred as membership service. This layer maintains an address pool of groups existing in the application along with the information that which processes are belonging to which group. This membership service assigns unique identifier to the groups. This paper is shedding light on various issues of membership service.**

Keywords—distributed system,group communication, static size, dynamic size, coupling

I. INTRODUCTION

Whenever a common message is to be delivered among various processes, it is always better to maintain a group of those processes. In Distributed System this group is sometimes also referred as Object Group or Vector of Objects. This group is a non homogenous collection of processes. Homogeneity exists only in terms of common messages that are to be received by these processes. When a group is created, instead of referring individual processes we can directly refer to the group. To refer a group uniquely, a unique group identifier is assigned to the group. It is to be noted that this uniqueness is per application basis. Size of group may be static or dynamic. In static size scheme, group size remains constant throughout the entire life cycle of the application. If size is dynamic, at any time a process can leave the group or a new process can join the group. There is no effect of size on the group identifier. It means processes of a group do not participate in formation of group identifier if application is supporting/ using only dynamic group size. However if application is

AP, Computer Science and Engineering

supporting/using only static group size, the residing processes of group may participate in formation of group identifier. Generally when we talk about a group, we talk about group head. For example if we want to convey a message in a class, we have three ways: (1) displaying the message on notice board (2) announcement in the class and (3) through class representative. Class representative may further announce the message in front of class or convey the message to each and every student personally or informs to some students and those students further convey to rest of the students[1,2,3,5,8,9]. The last option used by class representative is referred as gossip messages in Distributed System. In gossip messages, integrity of message depends on the process which is playing the role of sender in the chain. The malfunction behavior of sender process can compromise the integrity of message. Class representative knows how many students are there in the class and to whom this message is to deliver. If we are using group head concept in the group communication, message is delivered to the group head and group head may opt any scheme to convey the message as described above in case of class representative. Distributed application may discard the concept of group head also. An analytical model of group communication is given below. Membership layer is in between applications and groups. This layer is a core part of group communication and it is reducing the degree of coupling existing between application and groups.

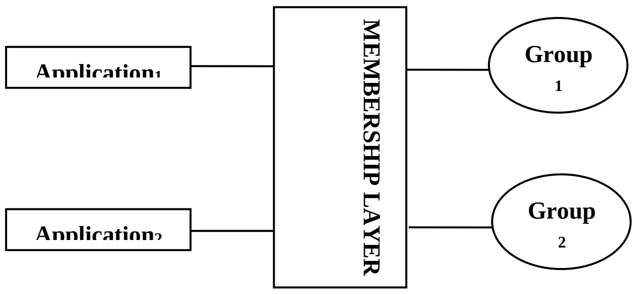

Fig1. Analytical model of Group Communication

Next two sections of this paper are explaining the pros and cons of the schemes.

II.WITH GROUP HEAD

Group head (GH) is a process which is elected from the residing processes of the group. Election algorithm is used in selection of group head. Group head has a list of all the processes residing in the group called member list (ML). When a message comes to the group head through membership layer, group head checks the ML and delivers the message to the individual process. Whenever a new process joins the group, group head adds this process to ML. When a process leaves a group the group head deletes the entry of this process from the ML. In this scheme, group head is single point of failure. If the group head crashes, new head is to elect by using some election algorithm and the ML of the group must be prepared again. It is to note that every process has an identifier called process id (PID). This PID is added and deleted from the list when a new process joins or leaves the group. When a new PID is added to the ML it is referred as address expansion. Election of the group head and maintenance of the list requires more message exchanges. For example, if Bully Algorithm is used to select a group head, in worst case n^2 messages are required. Fault tolerance policy is also ensured by the group head [9, 10, 11, 12]. Process of a group may become unreachable either because of network partition or because of crash. It becomes very difficult to decide whether to remove the process from the list or not. As size of group increases offered load on group head also increases and quality of service (QoS) decreases. Designers of the system have to pay special attention to this issue. Group head delivers view of group to the membership layer also. This view is periodically returned to the application by the membership layer. Next two subsections are describing the role of membership layer and message format.

A. *Membership Layer*

Membership layer is an intermediate layer in between applications and groups. Membership layer maintains a list of group heads and the resident processes of the groups. We see that list of members of a group is replicated at two points first at group head and second at membership layer. This list of group member (ML) can be opted by the new group head (in case of failure of a group head) to save both messages and time in reformation of the list. Only one update is required and it is the marking of new elected process as the group head. Message of an application come to the membership layer and the membership layer transfers the message to the appropriate group head. This selection is based on ID of group head which is assigned by the membership layer. Applications attached to other side of membership layer know the IDs of group head.

B. *Message Format*

Application sends message in the format Message (*SID*,*msg* ,*id*). Here,*SID* represents the id of sending application, *msg* represents the message and *id* represents the group head (GH) id. The membership layer delivers the *msg* to the appropriate group which is based on the id of GH attached with the message. Application along with the membership layer knows the status of group. Status of group means knowledge about the processes residing in the group. On requirement, application sends request to add/remove a process in/from a group. Addition of a new process requires a registration message having format REGISTER (*SID*, *pid*, *id*). Here *pid* refers to the *id* of process to be added and id is the GH id. Exclusion of a process from a group requires deregistration message having format DEREGISTER (*SID*,*pid, id*). Here *pid* and *id* have the same meaning. Membership layer acknowledges the registration and deregistration of a process to appropriate application.

III. WITHOUT GROUP HEAD

Here, there is no notion of group head (GH). In this scenario, membership layer keeps records of groups long with the residing processes. Every group is assigned a unique group identifier (gid).

A. Membership Layer

Membership layer maintains a list of *gids* along with its resident processes. Whenever a new process is added to this group membership layer adds its entry into the corresponding record of the group (ML). On removal of a process from a group also requires update in this list. Membership layer uses broadcast service to deliver the message. It is to be noted that processes may be local to a machine or they may be at remote machines.

B. Message Format

Application layer sends message with the format Message (*SID,msg,gid*). Here, *SID* represents sender identifier, *msg* represents the message and gid is group identifier. Request of registration and deregistration of processes terminates at membership layer. Here registration message has the format REGISTER (*SID*,*pid*, g*id*). Here *pid* refers to the g*id* of process to be added and id is the group identifier. Exclusion of a process from a group requires deregistration message having format DEREGISTER (*SID*,*pid, gid*). Here *pid* and g*id* have the same meaning. Membership layer acknowledges the registration and deregistration of a process to the application.

IV. ROLES OF GROUP MEMBERSHIP SERVICE

Major responsibilities of a membership services are as:

A. *GroupMembershipChanging Service*

A process can leave a group and can enter into another group at any time. Group membership service must detect this case very efficiently. A process may belong to several groups simultaneously. Let us consider following scenario: p_1∈ ◌̑ G_1 and p_1∈ ◌̑ G_2 . Process p_1 belongs to the group G_1 and G_2. If G_1 is destroyed, all the processes of G_1 are killed except p_1. Process p_1 is killed when all the groups are cancelled to whom it belongs.

B. *Implementation of Failure Detector*∉

A process may become unreachable at any time because of crash or network partitioning. Membership layer deploys fault tolerance service and it assigns status to other processes as *Suspected* and *Unsuspected*. A process may become unreachable because of network partitioning but it should not be marked as *Suspected* process by the fault tolerance service. Process is marked *Suspected* when it becomes unreachable (non responding) because of crash only. It is applicable for group head. Membership layer acknowledges the application about the unreachable process. Application then decides whether removal of this process is required or not. So fault tolerance service must be deployed wisely.

C. *Notifying Group Membership Status*

Whenever a new process joins a group or leaves a group, application is acknowledged by the membership layer. This process is referred as view delivery. Next section is shedding light on view delivery.

D. *Performing Group Address Expansion*

When a process joins a group or leaves a group its membership must be registered or deregistered. Delivery of message to a process depends on registration and deregistration of the process.

V.VIEW DELIVERY

View delivery means providing information about the changes taking place in the group. Changes occur when a process joins a group or leaves a group. Let us consider a process p and p_1, a group G and consider the views V_i as

- $V_0(G)=\{p\}$, this view says that process p joins group G. Before addition of process p, group G was an empty group.

- $V_1(G)=\{p,p_1\}$, this views says that another process p_1 has joined this group.
- $V_2(G)=\{p\}-p_1$, this view says that process p1 has left the group G.

An event is said to be occurred in a view V(G) at a process P if P has delivered a view at the time of occurrence of event and next view $V^1(G)$ is not delivered yet. Some requirements exist in view delivery as:

A. Order

If a process P delivers a view V(G) and then $V^1(G)$ then no any other process Q delivers $V^1(G)$ before V(G).

B. Integrity

If a process P delivers a view V(G) then p is residing process of group G.

C. Non-triviality

If P and Q are two members of a group and they are reachable to each then Q exists in the view delivered by process P and P exists in the view delivered by the process Q.

VI.PROPERTIES OF GROUP COMMUNICATION

Group communication must be reliable in the sense that when a message is sent to a group, all the resident processes of the group must receive this sent message. Reliable group communication depends on the parameters like *full delivery* and *correctness*. Various issues are associated with correctness as *message ordering*, *atomicity* and *survivability*. Coming sub sections are describing these parameters.

A.Full Delivery

Full delivery ensures that a message sent by sender is delivered to all members of the group provided that membership layer is not crashed.

B.Correctness

Correctness depends on following factors.

- *Order*

If a sender is sending messages as m_1 & m_2 to same group such as: *send*(m_1)->*send*(m_2) then *rec*(m_1)->*rec*(m_2). Here -> is representing happened before relationship. Here *send* and *rec* are sending and receiving events of messages. If m_1 is sent before m_2 then all members of that group will receive m_1 before receiving m_2. If two different senders are sending two different messages m_1 and m_2 to same group then all members must receive either m_1 before m_2 or m_2 before m_1.

- *Atomicity*

Atomicity ensures that either all members of a group will receive a message or none.

- *Survivability*

This property ensures that system is in operating mode in presence of failures also[2,3,7,8,9,12].

VII.INTER PROCESS COMMUNICATION

Processes residing in a group may communicate with each other by message passing. This choice is based on the requirement of application. If inter process communication occurs, every member comes to know about the status of group. This inter process communication facilitates in selection of new GH if election is required. If inter process communication is allowed store and forward mechanism of message sending may be used by the GH to pass the received message [1, 3, 5].

VIII.NON REPUDIATION OF MESSAGES

Application sending messages cannot deny for any sent message. To ensure this unique *SID* is attached with all the messages. Membership layer maintains a history of message exchange. Consistency and inconsistency of message can be concluded from this history. Membership layer when receives a message from an application to a group, it updates the history by adding this message to it with its timestamp and then it forwards the message to the corresponding group (this selection is based on *gid/id* of group/GH). We refer this history maintained by membership layer here as local state of a group. Consistency of message is given by:

send(msg) ∈ ◌̂ LocalState(gid) ^ rec(msg)∉LocalState(P) .1
send(msg) ∈ ◌̂ LocalState(gid) ^ rec(msg) ∈LocalState(P) .2

Equation(1) is representing that message is forwarded to the group by the membership layer but it is not received by the process. Here P is a resident process of group G. LocalState (P) represents the local state of process P. Here message is in *transit* mode and channels are not empty. Equation(2) says that message is sent by membership layer and it is received by the process. Here, message is in *transitless* state and channel is empty. Both the above cases are representing that consistency of message is preserved. Inconsistency occurred when
send(msg) ∉◌̂ LocalState(gid) ^ rec(msg)∈LocalState(P) .3
Message is not sent but received by a process.

IX.OPERATIONS PERFORMED ON GROUPS

System designers provide a set of APIs to perform various operations on groups as:

- Creating a Group
- Registering a Group
- Joining a Group
- Leaving a Group
- Merging Groups
- Cancellation of Group

Creating a group provides facility to create a group dynamically. Registering a group means registration of group at membership layer. Joining a group enables a process to join a group. Leaving a group means a process not leaving a specified group. On requirements various groups can be merged through merging groups operation. In last a group can be cancelled or destroyed by using the Cancellation of group. Membership layer exposes these APIs to both ends.

X.CONCLUSION

This paper is describing roles and responsibility of membership layer in group communication. Membership layer reduces degree of coupling existing in between applications and processes. Designers deploy set of APIs to the membership layer which is furthered exposed to the applications and groups to perform operations on groups. Instead of passing same message to individual processes, message broadcast is generally used. Fault tolerance policy can use any scheme to detect network partitioning and process crash scenario. Non repudiation of message cannot take place in the proposed architecture as sent message carries identities of senders. Group can be maintained with or without group head.

REFERENCES

[1]. Chung Kei Wong, Mohamed Gouda, and Simon S.Lam., " Secure Group Communications Using Key Graphs," ACM Sigcomm 98, 1998. University of Texas at Austin.

[2] X. Rex Xu, Andrew C. Myers, Hui Zhang, and Raj Yavatkar," Resilient Multicast Support for Continuous-Media Applications," NOSSDAV,1997.

[3].R. Housley, W. Ford, W. Polk, and D. Solo. ”Internet X.509 Public Key Infrastructure, Certificate and CRLProfile”. Internet Engineering Task Force, June 1998.

[4].H. Harney and C. Muckenhirn. ”Group Key Management Protocol Architecture,” Internet Engineering Task Force, July 1997. RFC 2094.

[5].M. Hiltunen and R. Schlichting. ”A Configurable Membership Service”. IEEE Transactions on Computers, 47(5):573–586, May 1998.
[6].Moser, L.E., Melliar-Smith, P.M., Agarwal, D.A., Budhia, R.K.,and Lingley-Papadopoulos, C.A. Totem, " A fault-tolerant multicast group communication system," Commun.ACM 39,Apr.1996.

[7].Fischer, M., Lynch N., and Paterson, M., "Impossibility of Distributed Consensus with One Faulty Process", J. ACM,32, April 1985, pp 374-382.

[8].Mishra, S., Peterson L., and Schlichting, R., "Consul: a Communications Substrate for Fault-Tolerant Distributed Programs", Distributed Systems Engineering, 1 (1993), pp. 87-103.

[9].Birman, K., Schiper, A. and Stephenson, P.,"Lightweight Causal and Atomic Group Multicast", ACM Transactions On Computer Systems, Vol. 9, No. 3, August 1991, pp. 272-314.

[10].Chandra, C.T. and S.Toueg, “Unreliable Failure Detectors for Asynchronous Systems” Proc. of 10th ACM Symp. On Principles of Dist. Comp., Montreal, , August 1991.

[11].Mostefaoui, A., Raynal, M., "Causal Multicasts in Overlapping Groups: Towards a Low Cost Approach", Research Report, IRISA Campus de Beaulieu -35042 RENNES, France.

[12].Schiper, A., and Ricciardi, A.M., "Virtually Synchronous Communication based on a Weak Failure Suspector", Digest of Papers, FTCS-23, Toulouse, pp. 534543,June 1993.