A Message Efficient Group Membership Algorithm in Mobile Ad Hoc Distributed Systems

Yong Hwan Cho, Sung Hoon Park, Seon-Hyong Lee and Byeong Sun Hwang
School of Electrical and Computer Engineering, Chungbuk National Unvi. Cheongju ChungBuk 361-763
E-mail: [spark,yhcho@chungbuk.ac.kr, genexwave@hanmail.net]

Abstract

The Group membership paradigm can be used as a building block in many practical problems such as group communication, atomic commit and replicated data management where a group membership protocol might be useful. The problem has been widely studied in the research community since one reason for this wide interest is that many distributed protocols need a Group membership protocol. However, despite its usefulness, to our knowledge there is no work that has been devoted to this problem in a mobile ad hoc computing environment.

Mobile ad hoc systems are more prone to failures than conventional distributed systems. Solving group membership in such an environment requires from a set of mobile nodes to choose a unique node as a leader based on its priority despite failures or disconnections of mobile nodes. In this paper, we describe a solution to the Group membership problem from mobile ad hoc computing systems.

Key-words: Synchronous Distributed Systems, Group membership, Fault Tolerance, Mobile Ad Hoc Environment.

1. Introduction

Distributed systems consist of groups of processes that cooperate in order to complete specific tasks. A Group Membership Protocol is of particular use in such systems, providing processes in a group with a consistent view of the membership of that group. In this way, when a membership change occurs, processes can agree on which of them must complete a pending task or start a new task. The problem of reaching a consistent membership view is very similar to the one of achieving common knowledge in a distributed system, commonly referred to as the Consensus Problem [7].

The Group membership problem [1] requires that every node connected in a network has a consistent group membership view if all connected nodes are belong to one group. The problem has been widely studied in the research community [2,3,4,5,6] since one reason for this wide interest is that many distributed protocols need a Group membership protocol. However, despite its usefulness, to our knowledge there is no work that has been devoted to this problem in a mobile ad hoc computing environment.

When nodes are mobile, topologies can change and nodes may dynamically join/leave a network. In such networks, Group membership can be changed frequently, making it a particularly critical component of system operation. Mobile ad hoc systems are more often subject to environmental adversities that can cause loss of messages or data [8]. In particular, a mobile node can fail or disconnect from the rest of the network. Designing fault-tolerant distributed applications in such an environment is a complex endeavor [9,10].

The aim of this paper is to propose a solution to the group membership problem in a specific ad hoc mobile computing environment. This solution is based on the termination detection algorithm that is a classical one for synchronous distributed systems.

The rest of this paper is organized as follows. Section 2 describes the mobile system model we use. In Section 3, a solution to the Group membership problem in a conventional synchronous system is presented. A protocol to solve the group membership problem in a mobile ad hoc computing system is presented in Section 4. We conclude in Section 5.

2. System Model, Constraints and Assumptions

Before developing a Group membership algorithm for ad-hoc computing environments, we first define our system model based upon assumptions and goals. We model an ad hoc network as an undirected graph, i.e., \( G = (V, E) \), where vertices \( V \) correspond to set of mobile nodes \( \{1, 2, \ldots, n\} \) \( (n > 1) \) with unique identifiers and edges \( E \) between a pair of nodes represent the fact that the two nodes are within each other’s transmission radii and, hence, can directly communicate with one another that changes over time as nodes move. Each process \( i \) has a variable \( N_i \), which indicates the neighboring nodes, with that \( i \) can directly communicate with the neighboring nodes. We assume that every communication channel is bidirectional; \( j \in N_i \) iff \( i \in N_j \). More precisely, in the network \( G = (V, E) \), we can define \( E \) such that for all \( i \in V \), \((i, j) \in E \) if and only if \( i \in N_j \). The graph can
become disconnected if the network is partitioned due to node movement. Because the nodes may change their location, \( N \), may be dynamically changed and so may \( G \) accordingly. We make the following assumptions about the nodes and system architecture:

- All nodes have unique identifiers. They are used to identify participants during the Group membership detection process.
- Links are bidirectional and FIFO, i.e. messages are delivered in order over a link between two neighbors.
- Node mobility may result in arbitrary topology changes including network partitioning and merging. Furthermore, nodes can crash arbitrarily at any time and can come back up again at any time.
- A message delivery is guaranteed only when the sender and the receiver remain connected (not partitioned) for the entire duration of message transfer. Each node has a sufficiently large receive buffer to avoid buffer overflow at any point in its lifetime.

The objective of our Group membership algorithm is to ensure that after a finite number of topology changes, eventually each node \( i \) has a consistent view of group membership of the group to which \( i \) belongs.

### 3. Group Membership Specification

We now define a specification, consisting of four properties, for a group membership algorithm. We assume the system to be initialized to a start state where the sequences are the same at all processes and the last nonempty views in their sequences are the ones reported by all failure detectors.

Property 1: Agreement. At any point in time, all processes have a consistent history.

Property 2: Termination. If there are no more changes in the local views of the processes, they eventually reach their quiescent states.

Property 3: Validity. If all processes in a view \( v^* \) perceive view \( v^* \) as their local view and they have reached their quiescent states, then the last nonempty elements of their sequences of global views are all at position \( j \) and must be equal to \( v^* \).

Property 4: Safety. Once a view is “committed” in the sequence of global views, it cannot be changed. The first property expresses agreement. Consistent history must be an invariant for any program that satisfies the specification. The second property expresses termination. When the inputs of all processes are stable, the processes are eventually going to stop changing their output sequences. The third property rules out trivial solutions where protocols never decide on any new view or always decide on the same view. It ensures that a protocol that satisfies the specification does something useful, by stating that when all processes in a set agree on such set, they must commit this common view at the same position \( j \) in their sequences of global views.

Note that this requirement is weak because a new membership is created only if the local views of the different processes in the membership reach agreement. The fourth property also rules out trivial solutions, requiring processes not to change old views in their sequences.

#### 3. Group Membership Algorithm in an Ad Hoc Network

In this section, we describe a Group membership algorithm based on the termination detection algorithm, simply TDA, by diffusing computations. In later sections, we will discuss in detail how this algorithm can be adapted to a mobile setting.

#### 3.1 A Group Membership based on TDA

We first describe our group membership algorithm in the environment of a static network, where we assume that nodes and links never fail. The algorithm consists of three phases operated at the node that initiates the group membership algorithm. 1) Scattering phase - it operates by first scattering the “who” message and 2) Gathering phase - it operates by then gathering the id of each node that is connected to the static networks. We refer to this computation-initiating node as the source node. 3) Completing phase – it operates by deciding the consistent view and announcing it as a consistent new view to all nodes.

As we will see, after gathering all nodes’ ids completely, the source node will have the information enough to determine a consistent group membership view and will then broadcast it to the rest of the nodes in the network. The algorithm uses three messages, i.e., Who, Ack and View.

1) **Scattering phase.** Who message is used to initiate the group membership protocol by “scattering” the Who message. When group membership protocol is triggered at a source node \( s \), the source node makes a waiting list \( w_l \) and a received list \( r_l \) and begins a diffusing computation by sending an Who message to all of its immediate neighbors. Initially the waiting list consists of only its immediate neighboring node’s ids and the received list is empty.

When node \( i \) receives a Who message from the neighboring node for the first time, it immediately sends the Ack message to the source node and propagates the Who message to all its neighboring nodes except the node from which it first received an Who message.

The Ack message sent by node \( i \) to the source node contains the ids of all its neighboring nodes that are needed for the source node to decide the consistent view of the nodes connected with a distributed network. After that, any Who message received by other neighboring nodes will be ignored.
2) Gathering phase. When the source node receives the Ack message from the node \( j \), it removes \( j \) from the waiting list and puts \( j \) into the received list and immediately checks one by one the every node’s id contained in the Ack message. If there is the any id in the Ack which has already been acknowledged, i.e. that means it is in the received list, it is discarded. Otherwise, it is put into the waiting list of source node and the source node waits the Ack message from it.

The waiting list is growing and shrinking repeatedly based on the received Ack messages, but the received list is steadily growing by receiving the Ack messages. But the waiting list eventually could be empty and the received list could include all ids of nodes connected to the networks when the source node received the Ack messages from all other nodes. Hence the source node eventually has sufficient information to determine the consistent view of the group based on the received list, because the waiting list could be eventually empty and it means that the source node has received the Ack messages from all the nodes.

3) Completing Phase. Once the source node has received Ack's from all other nodes, it determines the consistent view based on the received list and broadcasts a View message to all other nodes announcing the current view of the group. We illustrate a sample execution of the algorithm. We describe the algorithm in a somewhat synchronous manner even though all the activities are in fact asynchronous. Consider the network shown in Figure 1(a). In this figure, and for the rest of the paper, thin arrows indicate the direction of flow of Who message sending out Ack messages (denoted as “E” in the figure) to its immediate neighbors, viz. nodes B and C, shown in Figure 1(a).

As indicated in Figure 1(b), nodes B and C in turn propagate the Who message to its immediate neighbors only except the source node and send the Ack message with neighboring node list to the source node A. Hence B and C also send Who message s to one another. But the Who messages are not acknowledged to the source node since nodes B and C have already received Who messages from the source node respectively. The information about neighboring node is piggybacked upon the Ack message sent by each node. Upon received Ack messages from B and C, node A updates \( wla = \{ B, C \} \), \( rlb = \{ A \} \) with the neighboring node information piggybacked on the Ack messages.

5. Concluding Remarks

In this paper, we proposed an asynchronous, distributed group membership algorithm for mobile, ad hoc networks and showed it to be correct. We formally specified the property of our group membership algorithm using temporal logic. We have assumed the ad-hoc network topology is dynamically changing and nodes are frequently connected and disconnected over the networks. With this approach, the group membership specification states explicitly that progress and safety cannot always be guaranteed. In practice, our requirement for progress is that there exists a constant \( c \) such that if connection or disconnections occur for a period of at least \( c \), then by end of that period, the system reaches a state satisfying a consistent view. Furthermore, the system remains in that state as long as no failures or disconnections occur. In fact, if the rate of perceived a node failures in the system is lower than the time it takes the protocol to make progress and accept a new consistent view, then it is possible for the algorithm to make progress every time there is a node failure in the system.

In real world systems, where process crashes actually lead a connected cluster of processes to share the same connectivity view of the network, convergence on a new consistent view can be easily reached in practice. However, the algorithm should work correctly even in the case of unidirectional links, provided that there is symmetric connectivity between nodes. We are currently working on the proof of correctness in the case of unidirectional links. We are also investigating on how our group membership algorithm can be adapted to perform clustering in wireless, ad hoc networks.
Figure 1: An execution of group membership algorithm based on the node detection algorithm. Arrows on the edges indicate transmitted Who messages, while dotted arrows parallel to the edges indicate Ack messages. In Figure 1(c), the node D and F also send the Ack messages to the sources node when they received the Who message s from the B and C respectively. Each of these Ack messages contains the identities of the neighbor. Eventually, the source A hears all acknowledgments from all of other nodes except itself in Figure 1(d) and then decides the consistent view among the group and broadcasts it, via the View message shown in Figure 1(d).

6. References