A New Group Membership Protocol in Synchronous Distributed Systems

Sung-Hoon Park¹ and Yeong-Mok Kim²
¹,²Department of Computer Engineering, Chungbuk National Unvi., Cheongju, ChungBuk, Korea

Abstract - In distributed systems, a group of computer should continue to do cooperation in order to finish some jobs. In such a system, a group membership protocol is especially practical and important elements to provide processes in a group with a consistent common knowledge about the membership of the group. Whenever a membership change occurs, processes should agree on which of them should do to accomplish an unfinished job or begins a new job. The problem of knowing a stable membership view is very same with the agreeing common predicate in a distributed system such as the consensus problem. Based on the termination detection protocol that is traditional one in asynchronous distributed systems, we present the new group membership protocol in arbitrary wired networks.

Key-words: Synchronous Distributed Systems; Group membership; Fault Tolerance; Wired Arbitrary Network Environment

1. Introduction

In distributed systems, a group of computer should continue to do cooperation in order to finish some jobs. A group membership protocol is especially helpful tools to allocate processes in a same group with a same view of the membership of the group. Whenever a membership change occurs, processes can consent to which of them should do to finish a waiting job or begin a new job. The problem of getting a stable membership view is very same with the one of getting common knowledge in a synchronous distributed system such as the consensus problem [1].

The Group membership protocol [2] is that every process connected in a network requires getting a stable same group membership view if all connected process are belong to just one group. The problem was widely discussed at the study community. The reason for this great study is that many distributed systems need a group membership protocol [3,4,5,6,7]. In spite of such practically usefulness, to our knowledge there is only a few research that have been committed to this problem in a wired arbitrary connected computing environment.

Depending on process failure and recover, network topologies is changed and process may dynamically connect and disconnect over a wired network. In such wired networks, group membership can be changed so much, making it a special critical module of system software part. In wired arbitrary network systems, a lot of environmental adversities are more common than the static wired network systems such as that can cause loss of messages or data [8]. In particular, a process can easily get to fault by hardware or software problem and disconnect from the wired network. Implementing fault-tolerant distributed applications in such an environment is a complex and difficult behavior [9,10].

In this paper, we propose a new protocol to the group membership protocol in a specific wired distributed computing system. Based on the termination detection protocol that is traditional one in asynchronous distributed systems, we address the new group membership protocol. We make up of the rest of this paper as follows. In Section 2 we address the system model we use. In Section 3, we describe a specification to the group membership problem in a traditional synchronous distributed system. We also address a new protocol to solve the group membership problem in a wired arbitrary computing system in Section 4. In Section 5, we address conclude.

2. Computing System Model, Definition and Assumptions

In this section, we describe our models for capturing behavior of distributed systems. We use these models foe reasoning about correctness of our protocol as well as for analysis of distributed computations. Our model for distributed systems is based on message passing, and all of protocol is around that concept. Many of these kinds of protocol have analogs in the shared memory computing system but will not be addressed in this paper.

First, we define our system model based on some assumptions and after that we address our goals. We model a distributed system as a loosely coupled messaging-passing system without shared memory and a global clock. Our distributed computation model for a wired network is made up of as an undirected graph. That is, the undirected graph is described as \( G = (V, E) \), in which vertices \( F \) facing each other with set of process \{1, 2, ..., n\} ( \( n > 1 \) ) with unique identifiers and edges \( E \) between a pair of process correspond the fact that the two process are in each other's transmission radii. Hence, our distributed system has a channel to directly communicate with each other which changes over time when processes move.

Every process \( i \) has a variable \( N_i \), which denotes the neighboring processes, with that \( i \) can directly communicate the neighboring processes. Every process communicates with a channel that is
4. Group Membership Algorithm in Wired 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.

4.1 A Group Membership in a Wired Networks

We first address our group membership protocol in the wired network settings. In which we assume that process and channels have no faults.

The protocol is made up of three phases running at the process that starts the group membership protocol.
1) The first phase that is a diffusing phase and it works by first diffusing the “who” messages.
2) The second phase that is a searching phase and it runs by then accumulating the id of every process that is consist of the wired networks. We represent this computation starting processes as the start process.
3) The third phase is a closing phase that is managed by deciding the same view and announcing it as a stable new view to all process.

The start process will have the information enough to decide a uniform group membership view after taking all process’ ids completely and the start process will then broadcast it to the rest of the process in the network. The three kinds of message, Who, Ack and View are used to manipulate the operations.

As the first phase is diffusing computing phase, Who message is used to make a start of the group membership protocol by diffusing the Who message.

1) The first Phase: When group membership protocol is launched at a start process s, the start process makes a replying queue w and a accepted queue r. The start process makes a scattering computation by forwarding a Who message to all of its immediate neighboring processes. At the starting point, the replying queue makes up of only its most close neighboring process’s ids and the accepted queue has nothing.

When process i receives a Who message from the neighboring process for the first time, it immediately sends the Ack message to the start process and propagates the Who message to all its neighboring process except the process from which it first accepted an Who message.

The Ack message sent by process i to the start process contains the ids of all its neighboring process that are needed for the start process to decide the stable view of the process connected with a distributed network. After that, any Who message accepted by other neighboring process will be ignored.

2) The Second Phase: Searching phase. When the start process receives the Ack message was taken out from the process j, it takes j out from the replying...
queue and gets j into the accepted queue and as soon as possible it detects sequentially the each process’s id included in the Ack message. In the case that the start process accepts the Ack messages, however the replying queue is continually increasing by accepting the Ack messages. But the replying queue at the end could have no element and the replying queue could insert all ids of processes connected to the wired networks whenever the start process accepted the Ack messages from all other processes. Therefore the start process eventually has much information enough to decide the stable view of the group based on the replying queue. That is because the replying queue could be eventually unoccupied and it means that the start process has accepted the Ack messages from all the process.

3) The Third Phase: Once the start process has accepted Ack from all other process, it decides the stable view based on the replying queue and forwards a View message to all other process to let know the current view of the group. We show some sample running protocol as the protocol execution to explain more specific features. We address the protocol in a stable view based on the replying queue and forwards it, that is the View message displayed in Figure 1(d).

Each of these Ack messages includes the ids of the neighbor. All the time, the start A accepts all acknowledgments from all of other process except itself in Figure 1(d) and then determines the stable view between the group and forwards it, that is the View message displayed in Figure 1(d).

Figure 1: An example of group membership protocol execution on the process search protocol.

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.
6. References


