Distributed algorithms for mobile hosts

L.M. Patnaik
A.K. Ramakrishna
R. Muralidharan

Indexing terms: Mobile environment, Physical movement, Computer communications, Proxies, Static hosts

Abstract: In a mobile environment the physical movement of hosts causes changes in the physical topology of the network with time. Therefore, the direct execution of the existing distributed algorithms in a mobile environment renders them inefficient. Distributed algorithms are therefore modified by assigning proxies, which are static hosts in the network, to each of the mobile hosts. Proxies have been used at a level $d (d \geq 1)$ in the logical hierarchy of location servers. A mutual exclusion algorithm has been redesigned, and the results of simulations of the original and modified versions are presented. General purpose procedures have been developed which can be used in any distributed algorithm designed for a mobile environment and which uses static hosts as proxies.

1 Introduction

As computer networks such as the Internet have spread worldwide and computers have become smaller and smaller, a new environment has been created in which users carry portable computers and use them on the move to access networks. This is called 'mobile computing', which of late has become a buzzword in the area of computer communications.

Since mobile computers might not remain at a fixed location, a fixed wired physical connection cannot be maintained between a mobile computer and a communication network. So, communication from and to a mobile computer takes place through a wireless medium in the form of wireless messages.

Because of the mobility of hosts, the geographical distribution of mobile hosts keeps changing. This mobility of hosts makes distributed algorithms, which were designed assuming static hosts, of hosts, inefficient in a mobile environment. Thus, distributed protocols that run on mobile hosts might have to be modified so that they can be executed efficiently in a mobile environment using the available resources.

The only work that is directly related to redesigning distributed algorithms for mobile environments was reported by Badrinath *et al.* [1] . They raised some of the issues resulting from host mobility, and use level one proxies in the logical hierarchy of location servers to modify Lamport's mutual exclusion algorithm [2] and Lann's algorithm for mutual exclusion [3]. Badrinath *et al* discuss the creation of a logical hierarchy of location servers but use only level one proxies for modifying distributed algorithms.

If we use higher level proxies in the logical hierarchy of location servers, we expect that a reduced number of messages will be sent regarding the setting up of proxies compared to when lower level proxies are used. Thus, in our work we have used proxies at a level $d (d \geq 1)$ in the logical hierarchy of location servers. We have redesigned Raymond's tree-based algorithm for distributed mutual exclusion [4], and have simulated the modified algorithm and compared it with the original one in terms of communication costs incurred per critical section entry, average messages required per critical section entry and average response time.

2 System model and methodology

The entire geographical area is divided into wireless cells. There is a base station, called a hub or mobile support station (MSS) for each cell. These base stations are static. Each mobile host (MH) is part of one and only one cell at a time.

The hub is connected to an existing wired backbone transport network on one side and on the other side has a hub antenna with which it communicates, using radio waves or infrared waves, with the mobile hosts in its cell. Value added networks are connected to this wired backbone transport network. Other static hosts are also connected onto the wired network.

2.1 Message passing, terminology and assumptions

2.1.1 *Message passing:* An MH, to send a message to another host (mobile or static), first sends it through the wireless medium to the mobile support station. The MSS takes care of routing the message to its destination. Since the destination MH can be moving, locating the mobile host is a problem. Some routing protocols *[5, 6]* have been designed to address this problem.

2.1.2 *Terminology:* From the topology given above, we can see that the number of MHs, N, is far greater than the number of hubs (or MSSs), M, i.e. $N >> M$.

To make sure that performance measures are not dependent on the routing protocol used, a constant fixed search cost is assumed to be incurred whenever a

search for the MSS in whose cell the destination MH is present is done.

We use the terminology given in [1] for the purpose of computing the communication costs incurred per each critical section entry. The terminology used is as follows:

C_F: cost of sending a point-to-point message between any two fixed hosts

C_W: cost of sending a wireless message from an MH to its local MSS (and vice versa)

C_S: cost incurred in locating an MH and forwarding a message to its current local MSS, from a source MSS.

2.1.3 Assumptions: Some of the important assumptions made about the network are:

(a) The network is assumed to be fully connected.

(b) In a wireless cell, messages will be received in the order sent.

(c) An MH that leaves a cell will eventually enter some cell in the system after some time.

Each MSS has a list of identifiers of mobile hosts that are currently in its cell. This list is correspondingly modified when an MH sends a MOVING or ENTERED message to a relevant MSS.

Disconnection of an MH is handled similarly to MH switching cells. When an MH disconnects, there is **no** guarantee that it will eventually reconnect. This is not the case when an MH is moving off from a cell. An MH disconnects by sending a DISCONNECT message and can reconnect by sending a RECONNECT message.

2.2 Methodology

A logical hierarchy of location servers (LHLSs) in the form of a tree is created on the network to which hosts are connected. Location servers act as proxies for mobile hosts during execution of the distributed algorithm.

Whenever a logical structure has to be imposed on the mobile hosts, we instead impose the logical structure **only** on the proxies which are at level d in the LHLS. Each distributed algorithm is designed to work for a particular d level but can be modified to work with different d level proxies by imposing the logical structure on a set of proxies at the different d levels.

Since a proxy for an MH is at level d in the LHLS, whenever an MH moves into an an area with a different d level proxy, a new d level proxy will be assigned to the MH. The relationship between a MH and a d level proxy remains invariant otherwise. When $d = 1$ the mobile support stations act as proxies for mobile hosts. So, the association between an MH and its proxy keeps changing whenever the MH moves into a new cell. When d = maximum levels in the LHLS, there will be only one static host which acts as a proxy for all the MHs in the network.

3 Tree-based algorithm for distributed mutual exclusion

In this Section we modify the tree-based algorithm for distributed mutual exclusion [4]. We use the system model described in Section 2 and give general purpose procedures and general purpose data structures which can be used in **any** distributed algorithm that is designed using proxies. The proxies used for modified algorithm at a level d in the LHLS.

Informal description of algorithm

This algorithm is token based, with only the holder of the token being able to execute critical section code.

The nodes in the distributed system are arranged in an unrooted tree structure. All messages are sent along the undirected edges of the tree. The tree may either be a minimal spanning tree of the actual network topology, or merely a logical structure imposed on the network assumed. Each node knows only of the existence of its neighbours. Every time the token passes between nodes, the tree becomes directed with the new holder of the token being the root. Each edge is made directed and points towards the root. Each node maintains a requests queue into which requests from the neighbouring nodes and requests from itself are queued. When a node obtains privilege it is sent to the first node in its request queue.

3.1 Modified algorithm

In this algorithm a STATIC is assigned as a proxy for each MOBILE host.The proxy assigned is at level d ($d \geq 1$). The proxy for an MH remains the same as long as the MH moves between locations having the same d level proxy. Every MSS keeps information about the d level proxy in the logical hierarchy of location servers (which is at $d - 1$ levels above the current MSS).

In the original algorithm [4], a node needs to have PRIVILEGE (token) to execute the critical section. A REQUEST message indicates that the sender is nonprivileged and wants PRIVILEGE (in order to execute the critical section) either for itself or one of its immediate neighbours. A PRIVILEGE message indicates the transfer of PRIVILEGE from the sender to receiver.

One node is chosen as the initial holder of PRIVILEGE. A variable holder at node i indicates the holder of PRIVILEGE according to node i. The initial holder of PRIVILEGE informs all the other nodes that it is holding the PRIVILEGE, by sending an INITIALISE message to all its neighbouring nodes. When node i receives an INITIALISE message from the node j, holder *(i)* is set to j and the INITIALISE message is sent to all its neighbours other thanj.

In the modified algorithm we introduce three new messages in addition to the INITIALISE, REQUEST and PRIVILEGE messages. The new messages introduced are:

(a) PASSPRIV(i). This message, sent by node i, indicates something similar to granting of PRIVILEGE, when the receiving node is MOBILE, but is not exactly the same. If the receiving node is STATIC, this means that PRIVILEGE is passed back from an MH to its proxy.

(b) MYPROXY (i, CurrProxy, d). When an MH, MH_i moves from one cell to another, it sends a MYPROXY(i, CurrProxy, d) message to the MSS of the new cell after sending an ENTERED message. By sending this message, MH_I indicates to the MSS that CurrProxy is the proxy of MH_I at level d.

(c) SETPROXY (SavedProxy). When an MH, MH_I, sends a MYPROXY (i, Currproxy, d) message to an MSS, the MSS checks to see if CurrProxy is same as the proxy at level d for this MSS. If not, the MSS sends a SETPROXY (SavedProxy) message to MH_I. Here, SavedProxy is the proxy at level d for this MSS. The