Structuring Distributed Algorithms for Mobile Hosts

B. R. BADRINATH, Arup ACHARY A and Tomasz IMIELINSKI
Department of Computer Science, Rutgers University, New Brunswick, NJ 08903, USA

Distributed algorithms have hitherto been designed for networks with static hosts. Connectivity of such networks do not change unless hosts or communication links fail. With the introduction of mobile computing, this assumption is no longer valid. A mobile host (MH) can connect to the network from different locations at different times. This raises two fundamental issues: (1) a search is required to discover the current location of MH in order to deliver a message, and (2) a logical communication structure amongst MHs e.g., a logical ring, cannot be statically superimposed on the underlying physical network. Additionally, unlike their static counterparts, MHs are faced with severe resource constraints such as low bandwidth wireless connection to the rest of the network and tight restrictions on power consumption. This paper presents a operational system model for explicitly incorporating the effects of host mobility and proposes a general principle for structuring efficient distributed algorithms in this model. This principle is used to redesign two classical algorithms for distributed mutual exclusion for the mobile environment. We then consider a problem introduced solely by host mobility viz., location management for groups of MHs, and propose the concept of group location as an efficient approach to tackle the problem. Lastly, we present a framework which enables host mobility to be decoupled from the design of a distributed algorithm per se, to varying degrees.

1 Introduction

The design of distributed algorithms and protocols has traditionally been based on an underlying network architecture consisting of static hosts i.e., the location of a host within the network does not change. Consequently, in the absence of site and link failures, the connectivity amongst hosts in the network remains fixed. Distributed algorithms thus assume a model comprising of a set of processes, executing on static hosts, that communicate by messages over point-to-point logical channels. Each channel may span multiple physical links of the network; this set of links and the hosts at the endpoints of the channel does not change with time. Solutions to problems of synchronization and communication in distributed systems are based on this basic model. However, this model does not capture the features and constraints of a network with mobile hosts, and therefore, distributed algorithms based on this model, will need to be restructured for mobile computing.

Mobile computing requires integration of portable computers within existing data networks. A mobile host can connect to the network from different locations at different times. At the network layer, this has led to research on new addressing schemes and network protocols for routing messages to and from mobile hosts [6, 12, 19, 20]. Mobile computing also has significant implications for distributed data management [5, 10] and in particular, has motivated the design of distributed filesystems [14, 18] aimed at mobile users. The mobility of an intended recipient may cause it to receive the message at more than one location, or miss receiving it altogether: the problem of efficiently delivering a multicast message exactly-once to mobile recipients is considered in [1]. In [4], the characteristic features of a mobile computing environment are presented, how they affect distributed computations and briefly introduced a system model for incorporating host mobility.

Host mobility introduces new issues that were not present in distributed systems with static hosts. First, to deliver a message to a mobile host, it is necessary that the destination be first located within the network which we term as search. Second, as hosts move, the physical connectivity of the network changes. Hence, any logical structure, which many distributed algorithms exploit, cannot be statically mapped to a set of physical connections within the network. Third, mobile hosts have severe resource constraints in terms of limited battery life and often operate in a "doze mode" or entirely disconnect from the network. Disconnection in a mobile environment is distinct from a failure: disconnections are voluntary and so, a mobile host can inform the system of an impending disconnection prior to its occurrence. Lastly, communication between a mobile host and the rest of the network usually occurs via a wireless link, whose
bandwidth is an order of magnitude lower than wired links; further, transmission and reception of messages on the wireless link consumes power at a MH. These aspects are characteristic of mobile computing and need to be considered in the design of distributed algorithms.

This paper investigates how distributed algorithms should be structured for mobile hosts. It presents an operational system model for explicitly incorporating the effects of host mobility using cost parameters appropriate for the mobile computing environment. In this model, communication takes place only through exchange of messages between static and/or mobile hosts; hosts do not share memory or a common clock. All hosts and links are assumed to be free from failures. We propose the following principle for structuring efficient distributed algorithms for mobile hosts:

To the extent possible, computation and communication costs of an algorithm is borne by the static portion of the network. This attempts to avoid locating a mobile participant and lowers the “search cost” of the algorithm; additionally, the number of operations performed at the mobile hosts and thereby, consumption of battery power, which is a critical resource for mobile hosts, is kept to a minimum.

To illustrate how this simple principle can be used to meet the constraints of mobile computing, we consider distributed mutual exclusion for mobile hosts and two classical algorithms [16, 15] for this problem. We first show why both algorithms would be inefficient if executed directly on mobile hosts viz., why maintaining a replicated queue at the mobile hosts as in [15], or implementing a logical structure amongst mobile participants as in [16] will incur a high “cost”, with cost measures defined suitably to reflect the resource constraints of MHs. Then, using the above principle, we modify the two algorithms to derive solutions with a reduced “search cost” within the static network, which also meet the resource constraints of MHs viz., reduced power consumption and fewer messages over the low-bandwidth wireless links. Next, we consider the problem of efficient location management for groups of MHs, and propose the concept of “location view” to solve the problem. Finally, we present a framework for decoupling the effects of host mobility from the design of an algorithm to varying degrees, by associating a proxy on the static portion of the network for each mobile host.

2 The system model

The term “mobile” implies able to move while retaining its network connections [12]. A host that can move while retaining its network connections is a mobile host (MH). The infrastructure machines that communicate directly with the mobile hosts are called mobile support stations (MSS). A cell is a logical or geographical coverage area under a MSS. All MHs that have identified themselves with a particular MSS, are considered to be local to the MSS. A MH can directly communicate with a MSS (and vice versa) only if the MH is physically located within the cell serviced by the MSS. At any given instant of time, a MH may (logically) belong to only one cell; its current cell defines a MH’s “location”. In this paper, we assume that all hosts and communication links are reliable. Further, for simplicity of presentation, we assume that all fixed hosts act as MSSs and use the terms MSS and “fixed host” interchangeably.

![Diagram of the system model](image.png)

The system model consists of two distinct sets of entities: a large number of mobile hosts and relatively fewer, but more powerful, fixed hosts (MSSs). The number of fixed hosts will be denoted by \( M \) and that of MHs by \( N \) with \( N >> M \). All fixed hosts and the communication paths between them constitute the static/fixed network. A MSS communicates with the MHs within its cell via a wireless medium. The overall network architecture thus consists of a “wired” network of fixed hosts that connect the otherwise isolated, low-bandwidth wireless networks, each comprising of a MSS and the MHs local to its cell. Host mobility manifests itself as a migration of a MH from one cell to another.

To send a message from a MH \( h1 \) to another MH \( h2 \), \( h1 \) first sends the message to its local MSS over the wireless network. This MSS then forwards the message to the local MSS of \( h2 \) which forwards it to \( h2 \) over its local wireless network. Since, the location of a MH within the network is neither fixed nor universally known in the network, i.e. its “current” cell changes with every move, the local MSS of \( h1 \) needs to first determine the
MSS that currently serves h2. This is essentially the problem that has been tackled through a variety of routing protocols at the network layer (and below) in [6, 12, 19, 20]. Thus, the cost incurred to route and deliver a message to a mobile host, varies with the specific routing protocol being used. Our system model is not tied to any particular routing scheme for delivering a message to a mobile host; instead, we will assume that any message destined for a mobile host incurs a fixed search cost.

The static network provides reliable, sequenced delivery of messages between any two MSSs, with arbitrary message latency. Similarly, the wireless network with a cell ensures fifo delivery of messages between a MSS and a local MH. The cost of a message depends on the type of channel on which it is transmitted:

- $C_{\text{fixed}}$ – cost of sending a point-to-point message between any two fixed hosts.
- $C_{\text{wireless}}$ – cost of sending a message from a MH to its local MSS over the wireless channel (and vice versa).
- $C_{\text{search}}$ – cost incurred to locate a MH and forward a message to its current local MSS, from a source MSS. Note that this cost is always greater than or equal to $C_{\text{fixed}},$ and in the worst case, require a source MSS to contact each of the other $M - 1$ MSSs to determine the MH’s current location.

Based on the above cost assignments, a message sent from a MH to another MH incurs a cost $2 \times C_{\text{wireless}} + C_{\text{search}},$ while a message sent from a MSS to a non-local MH incurs a cost $C_{\text{search}} + C_{\text{wireless}}.$

In our model, host mobility is asynchronous, i.e. there is no bound on the time interval between a MH leaving its current cell and entering a new one; however, a MH that leaves its current cell will eventually enter some cell in the system. A MH may leave its current cell at any time. However, it is assumed that a message destined for a MH will eventually be delivered to it (after incurring a search), regardless of the number of moves it makes.

As stated earlier, a fifo channel exists from a MH to its local MSS, and another fifo channel from the MSS to the MH. If a MH did not leave its cell, then every message sent to it from the local MSS will be received by it in the sequence in which they were sent. But, since a MH may leave its cell at any time, the sequence of messages received at the MH is a prefix of the sequence of messages sent from the MSS and eventual delivery of a message to the MH is not guaranteed, i.e. if $m_1, m_2, \ldots, m_s$ be the sequence of messages sent from a MSS to a given local MH, then the sequence of messages received at the MH is $m_1, m_2, \ldots, m_s,$ where $s \geq r.$ The model requires that a MH send a leave($r$) message on the MH-to-MSS channel supplying the sequence number of the last message, viz. $r,$ received on the MSS-to-MH channel. Once this message is sent, the MH neither sends nor receives any further message within the current cell. Each MSS maintains a list of ids of MHs that are local to its cell; on receipt of leave($r$) from a local MH, it is deleted from the list. When a MH enters a new cell, it sends a join(mh-id) to the new MSS; it is then added to the list of local MHs at the new MSS.

Some algorithms for mobile hosts [1] may utilise a handoff procedure: when a MH switches cells, MSSs of the two cells execute the handoff procedure. A MSS may maintain algorithm-specific data structures on behalf of a local MH. When a MH moves into a new cell, data structures from the previous MSS are transferred (“handed over”) to the new MSS. For this to be realized, it is necessary that the MH either inform the previous MSS of the id of its new MSS or vice versa. In Section 4 of this paper, management of location view will require that a MSS supply the id of its previous MSS after entering the new cell (with the join() message).

Disconnection of a MH is handled similar to a MH switching cells. However, there is an important difference between the two. When a MH leaves a cell, it will eventually show up in some cell. On the other hand, when a MH disconnects, there is no guarantee that it will reconnect to the system at a later time. A MH disconnects by sending a disconnect($r$) message to its local MSS, where $r$ is the sequence number of the message last received from the MSS (similar to a leave($r$) message). The local MSS deletes the MH from its list of local MHs; however, it sets a “disconnected” flag for the particular MH-id. If and when, the MH reconnects at some MSS with a reconnect(mh-id, previous mss-id) message supplying the location where it had previously disconnected, the “disconnected” flag is unset as part of the handoff procedure. The MH may not always be able to supply the id of its previous MSS with the reconnect() message; in that case, the new MSS may have to query each fixed host to determine the previous location of the MH and then execute a handoff procedure. If some MSS attempts to search for a MH that has disconnected, the local MSS of the cell where the MH disconnected informs it of the disconnected status of the MH.

3 Structuring distributed algorithms

Motivation

We cast distributed systems with mobile hosts into a two-tier structure: (1) a network of fixed hosts with more resources in terms of storage, computing and communication, and (2) mobile hosts, which may operate in a disconnected or doze mode, connected by a low-bandwidth
wireless connection to this network. The guiding principle for structuring distributed algorithms for MHs in this model is that the computation and communication demands of a algorithm should be satisfied within the static segment of the system to the extent possible. Below, we present justifications for this choice:

- A message sent from a MSS to a non-local mobile host incurs a search cost. The same is also true for a message exchanged between two mobile hosts in different cells. To reduce the search component of the overall execution cost, it is desirable that communication between a fixed host and a mobile host occur locally within the same cell.

- The ability of a MH to operate while on the move, requires a stand-alone source of power viz., batteries. Given the limited life of batteries, power consumption is a serious practical consideration at a MH[2, 8, 11]. In addition to disk accesses and cpu operations, a MH has to expend its limited power resources to send and receive wireless messages; such a constraint is not faced by messages exchanged between fixed hosts over the wired network. Additionally, wireless channels have a significantly lower bandwidth than those within the fixed network. Thus, the number of wireless messages exchanged in any algorithm execution should be minimal, possibly at the expense of a higher number of messages exchanged within the fixed network.

The two points mentioned above suggest that the data structures encapsulating the “state” of an algorithm execution, should mostly reside at the fixed hosts; thus, messages generated to update these data structures will be addressed to fixed hosts. Communication necessary to execute an algorithm may be split into three components: global, local and search. The global component consists of messages whose source and destination are both fixed hosts, e.g. to update appropriate data structures, and mostly represents the communication necessary for the progress of an algorithm execution. The local component refers to communication within a single wireless cell between a MH and its local MSS, and will often be used to initiate an algorithm execution from a MH or to communicate the final result of an execution from a MSS to a local MH. The search component consists of messages that the fixed hosts exchange to determine the current location of a MH so that a message addressed to this MH, may be forwarded to the appropriate MSS. Thus, our approach suggests that the global component dominate the overall communication.

- The two unique modes of operation of mobile hosts viz., disconnected and “doze-mode”, provide compelling arguments against executing an algorithm directly on MHs. When operating in a doze-mode, the MH shuts/slow down most of its system functions to reduce power consumption, and only listens for incoming messages. Like disconnection, this is a voluntary operation. However, the implications are different. In doze mode, a mobile host is reachable from the rest of the system and thus, can be induced by the system to resume its normal operating mode, if required. In contrast, disconnection and subsequent reconnection is initiated from the mobile host; it is cut off from the system in the intervening period.

- A distributed algorithm designed for the mobile computing environment, should not require each MH to participate in every execution of the algorithm. Otherwise, it prevents those MHs from operating in a doze-mode that neither initiated the computation nor is the result of an execution significant to them and consequently, attempts at conserving power by operating in doze-mode are completely thwarted. Thus, by downloading most of the communication and computation requirements to the fixed segment of the network, the static hosts are responsible for the progress of an algorithm execution and a mobile host will not be required to intervene unless it is interested in the outcome of the execution.

- Algorithms that directly execute at the mobile hosts need to consider the possibility that one or more of the participants may disconnect while an execution of the algorithm is in progress. In this case, the disconnecting mobile host may be required to inform every other mobile participant of its impending disconnection: this involves a search overhead (to locate the remaining mobile participants) and additional communication cost. On the other hand, by making the static hosts responsible for the progress of an algorithm execution, the effects of a MH’s disconnection can possibly be localised within the static segment of the system.

- Many distributed algorithms rely on an underlying logical structure such as a ring, tree or grid, amongst the participants to carry out the needed communication. The main purpose of such a structure is to provide a certain degree of order and predictability to the communication amongst the participants; messages exchanged within such structures follow only selected logical paths. Consider now, the effects of the different operational modes of the mobile hosts on a logical structure comprising of MHs.

- Disconnection of a mobile node may require that the logical structure be reconfigured, resulting in addi-
tional message traffic and possible search overhead.

– Further, a logical structure predefines the sequence of nodes that a message should traverse starting from a given sender to its destination; thus, the intermediate nodes if operating in doze-mode, are forced to resume normal operation to forward such messages.

Thus, the cost of maintaining a logical structure amongst the mobile hosts may override the benefits of using such an underlying structure for algorithm design. Instead, it may be possible to obtain similar benefits by maintaining the logical structure amongst the fixed hosts without experiencing the disadvantages associated with that of mobile hosts.

**Distributed mutual exclusion**

The problem of distributed mutual exclusion will be used to illustrate our approach to designing distributed algorithms: a set of processes compete for access to a shared resource (“critical region”) such that no more than one process should be able to simultaneously access that resource. A distributed solution requires that there be no privileged process and communication take place solely by passing messages.

**Lamport’s algorithm**

We first look at the drawbacks of executing Lamport’s algorithm [15] directly at the N mobile hosts, while we will refer to as algorithm L1. Without delving into the details, consider only the communication pattern and data structures required by the algorithm. To secure mutual exclusion, a participant sends a request message to all other participants, wait for a reply message from each of them, and then send a release message to all others after completing its access to the critical region. Each participant is required to maintain a request queue of pending requests, which is updated on receipt of request and release messages. This approach has the following drawbacks:

– **High search cost.** Each message in the algorithm is addressed to a mobile host and therefore, incurs a search cost. The overall cost of one execution of the algorithm is \(3 \times (N - 1) \times (2 \times C_{\text{wireless}} + C_{\text{search}})\). Note that the search overhead is proportional to \(N\), the number of MHs in the system.

– **Battery consumption at MHs.** Updates to the request queue and message transmission and reception over the wireless links, consumes power at the MHs. The source and destination of each message in this scheme is a MH, and therefore, consumes battery power at both the sender (to transmit it to the local MSS) and the destination (to receive the message from its local MSS).

The overall energy consumption for one execution of the algorithm is thus proportional to \(6 \times (N - 1)\). The energy consumed at an initiator is proportional to \(3 \times (N - 1)\), while each of the other \((N - 1)\) MHs consume energy to receive two messages (request and release) and send one (reply) message each.

– **Fifo channels between MHs.** Correctness of the algorithm requires that messages are delivered in sequence (fifo) at a destination. Since in L1, the source and destination of every message is a MH, this requirement places an additional burden on the underlying network protocols to maintain a logical fifo channel between any pair of MHs, regardless of their location in the network.

– **Doze and disconnected modes.** Algorithm L1 requires the participation of every MH in every execution of the algorithm, and consequently does not permit any MH to disconnect or to operate in a doze mode without interruption even for the time interval during which it does not attempt to access the critical region.

We adapt Lamport’s algorithm to the mobile computing environment, by shifting the communication and computation requirements of the algorithm to the static segment. Instead of the \(N\) MHs executing the algorithm, it is the MSSs that maintain the necessary data structures viz., the request queue, and exchange request, reply and release messages amongst each other within the static network to secure mutual exclusion on behalf of a requesting MH. The necessary modifications to Lamport’s algorithm are as follows:

– Only messages exchanged between MSSs follow the timestamping rules of [15]; messages between a MH and a MSS are not timestamped.

– A MH \(h_l\) initiates L2 by sending a message, init(h_l), to its local MSS \(m_l\). \(m_l\) now executes Lamport’s algorithm with other MSSs, on behalf of \(h_l\): the request, reply and release messages are tagged with the initiating MH’s id \(h_l\).

– When \(m_l\) secures mutual exclusion for \(h_l\) (following the rules of Lamport’s algorithm), it sends a grant_request message to \(h_l\). Since \(h_l\) may have changed its cell in the meantime, this requires a search cost to locate \(h_l\).

– Having completed its access to the critical region, \(h_l\) sends a release_resource message to \(m_l\) (relayed via \(h_l\’s\) current local MSS). \(m_l\) then deletes \(h_l\’s\) request from its queue and sends a release(h_l) message to every other MSS (as required by Lamport’s algorithm).

– If \(h_l\) disconnects prior to receiving the grant_request message from \(m_l\) then, on receiving the grant_request message from \(m_l\) (to be forwarded to \(h_l\)), its current
local MSS will note that the “disconnected” flag is set for \( h1 \) and in response, notify \( m1 \) of \( h1 \)’s disconnected status. Since \( h1 \) is unreachable, its request will not be satisfied and \( m1 \) sends a release message to all other MSSs. If \( h1 \) disconnects after receiving the grant_request message but without sending release_resource, then L2 requires that it reconnect to send the release_resource message. Disconnection of \( h1 \) at any other time does not affect the progress of L2.

Correctness Lamport’s algorithm ensures that if the timestamp of a request R1 is less than that of another request R2, then R1 will be satisfied before R2. In algorithm L2, a request from a MH is timestamped when the init() message is received by its local MSS, i.e. though MHs do not maintain logical clocks, the timestamp assigned to request(h1) by \( m1 \) can be considered as the timestamp of \( h1 \)’s request for mutual exclusion. Since the MSSs execute Lamport’s algorithm without any modifications to the algorithm per se, a grant_request message will be sent to \( h1 \) before another MH \( h2 \) if the timestamp assigned to request(h1) is less than that of request(h2).

Communication costs First, the init message costs \( C_{\text{wireless}} \). Delivery of grant_request requires a search for the current local MSS of \( h1 \) followed by a wireless transmission, i.e. \( C_{\text{wireless}} + C_{\text{search}} \). Delivery of release_resource incurs a cost \( C_{\text{wireless}} + C_{\text{fixed}} \). Thus, the overall cost in this algorithm is:

\[
(3 \, C_{\text{wireless}} + C_{\text{fixed}} + C_{\text{search}}) + (3 \times (M - 1) \times C_{\text{fixed}})
\]

Comparison of algorithms L1 and L2. It can be clearly observed from the cost structures of algorithms L1 and L2, that L2 eliminates the drawbacks of L1 by explicitly acknowledging mobility of hosts and shifting the required data structures onto the static portion of the system: the participation of the mobile hosts was kept to a minimum.

- L1 incurs a search overhead proportional to \( N \), while L2 incurs only a constant search cost per execution.
- Also, since \( C_{\text{search}} > C_{\text{fixed}} \), and \( N \gg M \), the overall message cost is lower for L2 than L1.
- L2 requires only constant number of wireless messages, viz. \( 3 \, C_{\text{wireless}} \), and does not store the request queues at MHs; L2 is thus more energy efficient than L1 in terms of consuming battery power at the MHs.
- L2 does not require fifo communication channels with mobile endpoints.
- L1 does not provide for the disconnection of any MH. L2 is not affected by disconnection of a MH unless it has a pending request for mutual exclusion, in which case it tackles it efficiently.

Distributed mutual exclusion using a logical ring

Many distributed algorithms are based on a logical structure amongst participants; messages exchanged within such structures follow only selected logical paths. We next consider a classic solution for distributed mutual exclusion [16] that utilizes a logical ring, and show that in order to meet the constraints of mobile hosts, the logical ring should be comprised of the MSSs instead of the MHs.

In the algorithm of [16], all participants are logically arranged in an unidirectional ring and a token circulates in this ring. Each participant executes as follows:

- wait receipt of token from its predecessor in the ring;
- enter <critical region>, if desired;
- send token to its successor in the ring.

All communication in this algorithm, thus occurs only along the channels that define the logical ring. Algorithm R1 Consider an execution of the algorithm directly on the MHs wherein the \( N \) MHs form the logical ring. Now, the sender and recipient of every message is a MH and incurs a cost \( 2 \, C_{\text{wireless}} + C_{\text{search}} \). Thus, the total communication cost for the token to traverse the ring once, is \( N \times (2 \, C_{\text{wireless}} + C_{\text{search}}) \). Note that this cost is independent of \( K \), the number of mutual exclusion requests satisfied.

Inefficiency in maintaining a logical structure amongst mobile hosts stems from the fact that, unlike fixed hosts, the physical connectivity amongst mobile hosts is redefined on every move; this manifests itself through an increase in the search component of the overall communication cost of the algorithm.

Algorithm R2 Algorithm R2 maintains a logical structure amongst the MSSs: a token circulates amongst the \( M \) MSSs logically arranged in a unidirectional ring. Each MSS maintains a request queue. A MH that needs to access the critical region sends its request to its local MSS, which then inserts it at the tail of its request queue. When the token arrives at a MSS, all pending requests from the request queue are moved to a grant queue. The MSS (holding the token) then sequentially services each entry in the grant queue: it deletes the request at the head of the queue, sends the token to the MH that made the request and awaits return of the token from the same MH. This is repeated till grant queue is empty. The MSS then transfers the token to the next MSS in the ring. A MH, on receipt of the token, accesses the critical region and then returns the token to the MSS that sent the token.

Algorithm R2 illustrates an interesting interplay between mobility of hosts and the movement of the token.
amongst the static MSSs: after a MH’s request is satisfied at the current MSS, it is possible that it moves to a new cell under a MSS which is the next recipient of the token in the logical ring. The MH may then submit a new request at this MSS, prior to the arrival of the token. Thus, multiple requests from the same MH may be satisfied (at different MSSs) in a single traversal of the ring, and the total number of requests that may be satisfied is thus $N \times M$.

To prevent a MH from accessing the token more than once in one traversal of the ring, the token is associated with a counter (token_val) which is incremented every time it completes one traversal. Each MH maintains a local counter access_count whose current value is sent along with the MH’s request for the token to the local MSS. A pending request is moved from the request queue to the grant queue at the MSS holding the token, only if the request’s access_count is less than the token’s current token_val. When a MH receives the token, it assigns the current value of token_val to its copy of access_count. We will refer to this variation of R2 as R2’. The choice of using R2 or R2’ is governed by the following trade-off: R2 sacrifices “fairness” at the expense of satisfying more number of requests in one traversal of the ring, while R2’ ensures atmost one access to the token by a MH; both incur the same fixed cost to circulate the token amongst the MSSs.

**Communication costs.** The cost incurred by the token for one traversal of the ring is $M \times C_{\text{fixed}}$. A request message from a MH to its MSS requires a wireless transmission, and the MSS first needs to search a MH followed by a wireless transmission to transfer the token to the MH. To return the token to the MSS, the MH first transmits it over the wireless network to its local MSS, which then forwards it to the intended MSS over the fixed network. Thus, the cost of satisfying a single request for mutual exclusion costs $3 \times C_{\text{wireless}} + C_{\text{fixed}} + C_{\text{search}}$. The total cost of satisfying $K$ requests in one traversal of the ring by the token is thus,

$$K \times (3 \times C_{\text{wireless}} + C_{\text{fixed}} + C_{\text{search}}) + M \times C_{\text{fixed}}$$

where $0 \leq K \leq M \times N$ for R2, and $0 \leq K \leq N$ for R2’.

**Comparison of algorithms R1 and R2**

- **Disconnection and doze mode** Algorithm R1 is vulnerable to disconnection of any MH and requires the logical ring to be re-established amongst the remaining MHs when one or more MHs disconnect. However, with R2, disconnection of a MH that has not submitted a request for mutual exclusion, does not affect the rest of the system at all. In addition, it is also easy to handle disconnection of a MH with a pending request in R2: when the token is received by the local MSS of such a MH (after searching for it), it observes that a “disconnected” flag is set for the particular MH and returns the token back to the sending MSS. R1 also interrupts a MH operating in doze mode if it happens to be the next recipient of the token in the logical ring, irrespective of whether it made a request for mutual exclusion or not. In contrast, R2 interrupts a MH in doze mode only to satisfy a prior request from that MH.

- **Battery consumption** Algorithm R1 consumes battery power on every MH to first receive the token from its predecessor in the ring, and then transmit it to the successor even though it may have no use for the token itself. R2 avoids consuming battery power at every MH: only those MHs that request the token expend battery power to access the wireless network thrice, i.e. to transmit the request, receive the token and then return it back.

## 4 Location Management

A widely used abstraction for building distributed applications is that of *process groups*. A collection of processes or processors are grouped together to provide a desired service. In implementing a process group, there are two related issues to be considered: group membership and group communication. Group membership defines the set of processes currently constituting the group in the face of member failures and new members joining the group, while group communication defines the desired semantics of message delivery amongst group members, e.g. reliability, message ordering and time of delivery. Both issues have been extensively studied in the current literature, e.g. [7, 9, 13, 17]. However, the impact of host mobility on process groups has not been considered to date.

It is our thesis that when groups consists of mobile hosts, *mobility of members* introduce a fundamentally new aspect to process groups, namely *group location*. Group location is defined as the set of current locations, one per group member. Whenever a member moves to another cell, its current location and therefore, group location changes: thus, given its dynamic nature, there is a need to efficiently manage group location for MHs.
In static systems, location management for groups is not necessary since location of a host does not change for the period of its membership in the group; group membership automatically defines group location. For groups consisting of MHs, group location cannot be inferred from group membership; the problem is to efficiently maintain the location of group members even after assuming that group membership does not change.

The utility of maintaining group location lies in cutting down the search cost necessary to send messages to the entire group. On the other hand, maintaining consistent group location requires propagating location updates to members, i.e., inform cost. As related work, [3] presents a theoretical approach to tackle this trade-off between search and inform costs for individual mobile users. At a more practical level, network-layer protocols [6, 12, 19, 20] also implicitly consider this problem of locating MHs in order to route a message; here too, the scope of the problem is restricted to locating single mobile destinations. We consider the trade-off in the context of efficient management of group location.

In the following discussion, we use the term “group message” to refer to a message sent to all group members as part of some underlying group communication. Let \( G \) be a group, and \( |G| \) be the number of MHs in the group. To compare different strategies, we consider the cost incurred by each strategy for a duration of time wherein the total number of moves made by members of \( G \) is \( MOB \) and the total number of group messages sent is \( MSG \); group messages sent to update locations are not counted in \( MSG \).

**Pure search strategy**

In this approach, a MH only maintains the list of MHs that comprise \( G \). After a move, a MH does not inform any other group member of its new location. To send a group message to \( G \), a MH sends a point-to-point message separately to each MH in \( G \). Each message incurs a search cost and the overall cost of a group message is:

\[
C_{group} = (|G| - 1) \times (2 \times C_{wireless} + C_{fixed} + C_{search})
\]

The overall cost of this scheme is \( MSG \times C_{group} \), and is independent of \( MOB \).

This approach of “search on demand” is similar to that of the network-layer routing protocol of [12] for individual MHs in the sense that no location information of individual MHs is maintained on a permanent basis (it may be cached temporarily at a MSS in [12]). Our approach essentially extends the idea to groups of MHs.

**Always-inform strategy**

A MH maintains a location directory \( LD(G) \), that is a list of \(<h, \text{location of } h>\) pairs for every MH \( h \) that belongs to \( G \).

- To send a group message, a MH consults \( LD(G) \) and sends a copy to the current location (MSS) of every member, which forwards it to the MH. The cost of a group message is:

\[
(|G| - 1) \times (2 \times C_{wireless} + C_{fixed})
\]

- After a move, a MH sends a location update message to the current location of each group member (obtained from \( LD(G) \)) informing them of its new location; the cost of sending a location update is same as the cost above for a group message. Each recipient updates the entry \(<h, \text{location of } h>\) in its \( LD(G) \) on receiving a location update from \( h \).

- The total cost\(^2\) incurred by this scheme is

\[
(MOB + MSG) \times (|G| - 1) \times (2 \times C_{wireless} + C_{fixed})
\]

A more meaningful figure of merit is obtained by distributing the total cost incurred, over the \( MSG \) number of group messages: the effective cost of a group message is therefore:

\[
C_{group} = (MOB_{MSG}) + 1 \times (|G| - 1) \times (2 \times C_{wireless} + C_{fixed})
\]

The key point to note here is that the mobility-to-message ratio\(^3\) \( \frac{MOB}{MSG} \) determines the efficiency of this scheme, i.e., mobility of group members is reflected in \( C_{group} \).

We note that the network-layer routing protocol of [6] maintains the individual locations of all MHs in a location directory. Our strategy essentially applies this idea to groups of MHs.

**Location view**

In the two earlier approaches, group location is either determined on demand (pure search) or explicitly stored (always inform) on a per MH basis. This results in either a high search cost to send a group message or a high inform cost when a member changes its location. Instead of maintaining locations of each individual member, we introduce the concept of location view: for a group \( G \), the location view \( LV(G) \) is the set of MSSs each of which has a member of \( G \) located in its cell. It can be expected that \( |LV(G)| \) will be significantly smaller than

---

\(^2\) It is possible that the location of a group member changes while a group message is in transit to its previous location. In this case, the sender will need to send a second copy of the message to its new location. We disregard this possibility in the above calculations.

\(^3\) A similar ratio, call to mobility ratio, has been proposed for tracking locations of individual mobile users in [10].
|G|, especially for groups whose members are localised in a few cells.

- Each MSS in $LV(G)$ maintains a copy of $LV(G)$, and the local MHs in its cell that belong to $G$.
- $LV(G)$ changes only when a MH in $G$ moves to a cell that does not currently belong to $LV(G)$, or when the only member of $G$ to be located in a given cell in $LV(G)$ moves out of that cell: we will refer to such a move as a significant move. No change to $LV(G)$ occurs when a member of $G$ moves between cells within $LV(G)$.
- Since, $LV(G)$ may be updated due to concurrent significant moves, it becomes necessary to serialise changes to $LV(G)$ so that all copies of $LV(G)$ are updated in the same sequence. One simple way of achieving this is to assign a fixed MSS as a coordinator for $G$ which also maintains a copy of $LV(G)$. All changes to $LV(G)$ are first sent to this MSS which then forwards it to the MSSs in $LV(G)$ and also updates its own copy. Since the static network guarantees delivery, copies of $LV(G)$ at different MSSs will receive updates in the same sequence.
- If a MH in $G$ moves to a cell under MSS $M$ that is outside $LV(G)$ then, as part of handoff, the MH first supplies the id of the MSS $M'$ of its previous cell to $M$, along with the join message. $M$ requests $M'$ to notify the group coordinator to include $M$ in $LV(G)$. The coordinator then sends the latest copy of $LV(G)$ to $M$, and sends an incremental update to $LV(G)$ (i.e., to add $M$) to all other MSSs in $LV(G)$. Conversely, when the sole member of $G$ moves out of the cell under $M'$, it requests the coordinator to be deleted from $LV(G)$. If both the cases apply i.e., the only member of $G$ in the cell under $M'$ moves to $M$ (which does not belong to $LV(G)$), then $M'$ sends a combined request (to add $M$ and to delete $M'$ from $LV(G)$) to the coordinator. Thus, the cost of updating $LV(G)$ is atmost $\left(\|LV(G)\| + 3\right) \times C_{\text{fixed}}$

- A MH sends a group message $m$ by transmitting it to its local MSS. The local MSS consults $LV(G)$ and sends $m$ to every MSS listed in $LV(G)$; each recipient MSS forwards $m$ to its local group members. The cost incurred to send a group message in this scheme is $\left(\|LV(G)\| - 1\right) \times C_{\text{fixed}} + |G| \times C_{\text{wireless}}$

Similar to the always-inform strategy, we assume here that $LV(G)$ dose not change while a group message is in transit.

We now calculate the total cost incurred under this approach. Let $f$ be the fraction of moves that are significant, i.e. lead to a change in $LV(G)$. Then, starting from an initial view $LV(G)^0$, the location view progresses as $LV(G)^1, LV(G)^2, \ldots LV(G)^k$ where $k = f \times MOB$ is the number of significant moves. In this sequence, let $LV(G)^{max}$ be the location view with the maximum number of elements (MSSs); since, each significant move can increase or decrease the size of $LV(G)$ by atmost 1, $|LV^{max}(G)| \leq |LV^i(G)| + k$. Also, the cost of a group message depends on the size of $LV(G)$ at the time of sending the message; assume that the $j^{th}$ group message is sent in the $l(j)^{th}$ location view. Then the overall cost incurred by this scheme is:

$$C_{\text{group}} \leq \left( f \times \frac{MOB_{\text{MSG}}}{MOB_{\text{MSG}}} + 1 \right) \times |LV(G)^{max}| \times C_{\text{fixed}} + (3f \times \frac{MOB_{\text{MSG}}}{MOB_{\text{MSG}}} - 1) \times C_{\text{fixed}} + |G| \times C_{\text{wireless}}$$

Observe that $C_{\text{group}}$ now depends only on the significant fraction of the mobility-to-message ratio.

**Comparison of three approaches:** The effective cost of sending a group message in the pure search strategy is independent of the mobility of members, while it depends on the mobility-to-message ratio for the always inform strategy. By introducing the concept of location view, this cost becomes proportional only to the significant fraction of the mobility-to-message ratio. In addition, the number of messages sent over the static network is proportional to $|G|$ in both pure search and always inform strategies, while it is proportional to $|LV(G)^{max}|$ using location view; $|LV(G)^{max}|$ will be significantly less than $|G|$ when members of $G$ are concentrated in a few cells. Finally, by using the concept of location view, the onus of location updates has been shifted to the static segment of the network: this conserves battery power at the MHs and reduces the number of wireless messages (as shown by the $C_{\text{wireless}}$ product terms in $C_{\text{group}}$ computed for the above strategies), and also allows MHs to disconnect without adversely affecting the process of location management.

5 Separating mobility from algorithm design

This paper cast distributed algorithms for a mobile
computing environment into a two-tier structure: (1) a network of static hosts, and (2) mobile hosts that connect to different locations in this static network at different times. A key element of this structure is the association between a MH and a MSS on the static network. We term the MSS currently responsible for communicating with a MH as its proxy. Different algorithms based on the two-tier structure can then be characterised by the relationship they specify between a MH and its proxy. The association between a MH and its proxy is governed by the following parameters:

- **Scope of a proxy** determines which mobile hosts are associated with a given proxy. In both algorithms L2 and R2, the scope of a proxy is based on the location of a MH: the proxy associated with a MH is always its local MSS.

- **Obligations of a proxy** specify how a proxy should behave when a local MH leaves its cell without waiting for a computation to terminate, that was initiated by the MH at the proxy. For example, in L2, if a MH submits a request for mutual exclusion at a MSS and switches its cell prior to receiving a corresponding request grant, then the MSS (proxy) is obligated to search for the MH when the MH’s request is at the head of its queue.

The concept of using a proxy to handle mobility appears to be an appealing one: it has been used to filter and/or delay delivery of messages to a MH in [2], and to present a fixed location on behalf of a mobile client to applications running on a static host [21]. Here, we apply the concept to **decouple the effects of mobility from the design of a distributed algorithm**.

A distributed algorithm for static hosts can be extended to cover MHs in a uniform manner as follows: a proxy is associated with a MH for the duration of the MH’s lifetime; a proxy will be informed about the location of its MH(s) whenever such a MH changes its location, and will be responsible for receiving and sending messages to the MHs associated with it. Now, the distributed algorithm can be extended to the mobile environment by **executing the algorithms at the proxies of the participating mobile hosts**. This provides a two-layer structure in which one layer executes the algorithm over the set of static hosts (proxies) and the other layer handles host mobility, i.e., the interaction between a proxy and the MHs “under” it. In this case, with a fixed association between a MH and its proxy, a **total separation** of mobility from the algorithm is achieved. However, this is not always a desirable solution because a proxy has to be informed of every move by a MH; in case of “wide area moves” and for MHs that frequently change their cell, this leads to a high message traffic from the MH to its proxy and may be infeasible from a practical standpoint. Thus, we need to look for less static solutions in which the association between the MHs and proxies change, depending on the mobility of hosts.

One way of flexibly associating a proxy with a MH is to explicitly incorporate the mobility of hosts into the structure of the algorithm itself. The algorithms L2 and R2 presented in this paper reflect this approach. The association between a MH and its proxy changes on every move: the local MSS acts as a proxy to the MHs in its cell. Thus, a MH need not inform any non-local MSS of its current location. However, the obligations of a proxy may involve searching for a MH (as is the case for L1 and R2), and needs to be carefully defined for correctness of the algorithm.

An intermediate approach between a lifelong association of a proxy with a MH and assigning a new proxy on every move, is to consider a logical hierarchy of fixed hosts, called “location servers”, such that the MHs are the leaves that attach to this hierarchy: MSSs are at level 1 in the hierarchy. The proxy for a MH is now defined to be a location-server that is d levels higher than a leaf. In this case, the association between a MH and its proxy remains invariant along as the MH moves between locations that have a common d-level ancestor. A distributed algorithm designed for this setup will need to explicitly handle the effects of mobility, since the association between a MH and its proxy could change; however, compared to the approach where the association between a MH and its proxy is redefined on every move, the association does not change on every move in this approach. When the value of d is 1, then this approach is equivalent to associating the local MSS as the proxy of a MH, with the association being redefined on every move; when d equals the number of levels in the hierarchy, then there is only one proxy for all MHs, i.e. a centralized and static solution.

### 6 Conclusions

The design of algorithms for distributed systems and their communication costs have been based on the assumptions that the location of hosts in the network do not change and the connectivity amongst the hosts is static in the absence of failures. However, with the emergence of mobile computing, these assumptions are no longer valid. Additionally, mobile hosts have severe constraints on energy consumption, computing power and size of available memory, compared to fixed hosts. This paper first presents a new system model for the mobile computing environment and then describes a general principle for structuring distributed algorithms in this model.

The basis of our approach to designing efficient distributed algorithms in this model was to localize the communication and data structures necessary for an algorithm
within the static portion of the network to the extent possible; the core objective of the algorithm is achieved through a distributed execution amongst the fixed hosts while performing only those operations at the mobile hosts that are necessary for the desired overall functionality. Power consumption at the mobile hosts is thus kept to a minimum, and since updates to the data structures are performed at the fixed hosts, the overall search cost is reduced. A fundamental problem in distributed systems viz., mutual exclusion, served as an illustrative example. Next, we considered different strategies to efficiently maintain location information for groups of MHs. Here also, we see that it is advantageous to maintain and update group location as a set of MSSs under which group members reside, rather than associating location information with individual members. Finally, we introduced the notion of a proxy on the static network, that is associated with each mobile host: the effects of host mobility on the structure of a distributed algorithm can be isolated to a desired degree by appropriately defining the scope and obligations associated with a proxy.

Bibliography


