Distributed Algorithms for Mobile Environment

R. K. Ghosh

May, 2005

5.1 Introduction

At the fundamental level of human interactions with computers, mobile computing can is a new form interfacing with computers. A user may carry a portable device any where (aided by untethered wireless connectivity) and accessing computational resources for normal usage. Therefore, most of the issues concerning mobile computing arise out of wireless communication, portability and mobility of devices, and softwares that has to deal with characteristics of mobile devices as well as requirements of mobile applications.

From application prospectives, a mobile computing system is a distributed systems consisting of one more mobile devices and a set of static computers connected by a computer network. A major part of the research in mobile computing system is directed towards establishing and maintaining connectivity between mobile terminals with static computers through bridges between wireless with wired networks. Over the years, however, mobile computing has emerged as a distinct paradigm with problems that are characteristically different from those in normal distributed computing.

From an abstract point of view, a mobile computing system can be seen as graph that consists of a fixed core of static nodes and a dynamic set of mobile leaf nodes [6]. It is much like a cellular mobile telephone network. The mobile leaf nodes can be viewed as a set of persistent messages moving through graph. With this underlying graph model, traditional distributed algorithms can be implemented directly on mobile system. Unfortunately, a direct mapping of distributed algorithms to mobile environment is not practical due to limited bandwidth, fragility of wireless links and many other constraints connected with mobile nodes.

A commonsense driven approach to design a distributed algorithm for mobile system will be to leverage as much as possible the fixed part. It intrinsically formulates a logical two-tier approach to computing in a mobile distributed environment. By offloading most of the compute intensive tasks on the computers belonging to the fixed network, mobile devices can save its critical resources including battery power. Before we expand on the stated idea of two-tier approach, let us examine the difference between a traditional distributed system and a mobile computing system.

The mobility of a computing node brings up two important new issues in data delivery:

1. Locating a node for delivery of message, and

2. Transparent semantic routing of the message thereafter.

Consequently, any attempt to map existing distributed algorithms for execution in mobile computing environment in a simple way will unlikely to meet with much success. Nevertheless, observing the differences between mobile computing and distributed systems will help in recognizing the issues that may arise in design of distributed algorithms or restructuring existing distributed algorithms for execution on mobile computing systems.

5.2 Distributed Systems and Algorithms

A distributed system consists of a set of autonomous computers (nodes) which communicate through a wired network. A program which runs on distributed system is typically organized as a collection of processes distributed over different nodes of a distributed system. The computation performed such a program is called a distributed computation. A distributed computation consists of four iterative steps, (i) broadcasting, (ii) information gathering, (iii) executing some joint computation, and (iv) agreeing on coordinated actions by the participating processes. The last three steps are closely related. Information gathering in a distributed setup requires participants to exchange message amongst themselves. Similarly joint computation progresses with exchange partial results amongst participants. Likewise, a coordinated action requires information exchange. Since the processes are distributed over a set of autonomous computers, synchronization requirements a distributed algorithm can be met by either through a shared memory or through exchange of messages over the network among the nodes. A shared memory in a distributed system is mostly implemented at software level either transparently by extending the underlying virtual memory architecture or by explicitly through a set of library functions. So, typically message passing is the basic interface of exchanging information between compute nodes in a distributed system.

All distributed algorithms are designed with certain basic assumptions about the capabilities of the participating nodes.

1. The nodes are static and their locations (IP/MAC addresses) are known in advance. Therefore, no cost is incurred for locating a host.

2. The message setup cost is fixed and same for all the messages. Therefore, the transfer of message dominates messaging cost.

3. Sufficient bandwidth is available, transfer of large messages possibly may incur long network latencies. Therefore, it suffices to assume that a fixed cost is incurred for sending a message between two fixed nodes.

4. The participating are nodes resource rich, having enough computation power, memory.

5. The nodes are powered by continuous supply of power. Hence, do not have energy problem, and remain active during the full execution of the algorithm.

6. Their inability to receive a message due to a power failure is treated as a failure of the algorithm.

5.3 Mobile Systems and Algorithms

Before dealing with the design of distributed algorithms for mobile environment, there is need to understand how these algorithms can be evaluated to determine their efficiencies, because the evaluation criteria will influence the design of algorithms for mobile computing environment. As obvious from the characteristics of a mobile computing environment, the efficiency requirement of a distributed algorithm suited for such an environment should be to

- Minimize communication cost,
- Minimize bandwidth requirement,
- Meet the synchronization requirements, and
- Overcome resource constraints of mobile hosts.

Bandwidth is usually treated as a resource. Therefore, the impact of poor bandwidth can be examined along with the other resource constraints.

Synchronization is a key issue for the correct execution of any distributed algorithm. Unlike static clients, mobile clients can appear and disappear in any cell in a service area at any time. So, synchronization techniques should be adjusted to handle dynamically changing locations of the peers. Therefore, synchronization of mobile clients more complicated and far more expensive than that of clients in a static environment. There is, thus, a need to evolve of a new model for evaluating the cost of distributed algorithms in mobile environments. Apart from some easily identifiable cost criteria such as,

- cost of computation on a mobile node versus that on a static node,
- cost of relocating computation to static host, and
- cost of communication on wireless channels.

There are other cost related elements such as *disconnection* of a mobile host from the fixed network and its subsequent reconnection to the fixed network. Furthermore, the cost model applicable for mobile infrastructured network is not directly extendible to infrastructureless mobile ad hoc networks. So, separate cost models need to be developed for different mobile networks.

The above discussions indicate that the efficiency of algorithms for mobile environment can evaluated under following three broad criteria:

- How best a compution in a mobile environment can be modeled?
- How synchronization and contention problems can be resolved?
- How a cost model for messaging can be developed?

5.3.1 Placing computation

Whenever an operation is to be executed on a remote object it is achieved by sending messages to the node that hosts that object. The operation then gets performed by the remote node on behalf of the initiating host. It is convenient to think that the mobile host *logically* executing operations directly on the remote node. Sending a message to a mobile host itself is two step task:

1. First step is to locate the mobile host.
2. Next step is to actually send the message.

If the message is meant for a fixed host, the above two steps can be carried out by the base station (BS) of the mobile host within the fixed network. The BS being a part of fixed network would be able to forward the message to the destination node by leveraging the IP forwarding protocol. Sending messages to a fixed host is thus, lot less expensive than sending messages to a mobile host, because no location search is required for delivering messages to a fixed host. Thus, it is preferable to avoid sending messages to mobile hosts except the case when both sender and the receiver are under the same BS. As mobile hosts are required to avoid sending messages to one another, a mobile host should avoid executing operations on objects resident in another mobile host. So, the first design principle is:

Principle 1 *[1] To the extent possible, all remotely accessed objects should be resident on the fixed hosts.*

In our case an object is associated with the entity that requires computing power and bandwidth, thus, we disallow such things on the mobile host. The role of the mobile host comes up, as being involved in the thread of execution which executes the operations on the object (in most case the operation is staged by sending message to the fixed remote node holding the object).

5.3.2 Synchronization and contention

Whenever a particular resource is accessed from a number of remote locations there would be contention. We can treat this resource as an object. Each sequence of operations that are being initiated from a specific place (a mobile host) can be called a *thread* of execution. Thus, execution scenario is that of many concurrently running threads trying to operate on an object. For the moment, let us not make any assumptions about where this object is located. It may be either be resident on a fixed host or on a mobile host. The concurrent threads compete to gain access to the object. It is undesirable to have the object in an inconsistent state due to concurrent operations by the competing threads. In other words, the access to on the object by mutually competing threads should be synchronized.

Let us examine why mutual exclusion is an important issue in a distributed settings. The general pattern of all distributed algorithms can be visualised as comprising a *communication* component and *computation* component. The computation component is limited to the individual hosts and during execution of a computation, a host might need to communicate with its neighbours or other nodes for exchanging the results of partial computations. So further progress in computation may need synchronization. The synchronization requirements among other things may involve parameter initializations for the next phase of computation. Thus a distributed system of hosts exhibit bursts of communication in between periods of local computation.

When hosts become mobile, one additional cost parameters, namely, *cost of location lookup* is introduced. Furthermore, due to resource poorness of mobile hosts, the cost assignment criteria of computational resources at mobile hosts become drastically different from that at fixed host. The communication cost also needs to distinguish between costs incurred for messaging over wired and wireless links. A simple technique, to avoid the high resource cost at mobile hosts is to relocate compute intensive parts of the algorithm, as much as possible, to the fixed hosts. To address location lookup cost, three different strategies, namely, the *search*, *inform* and *proxy* are proposed by Badrinath, Acharya and Imielinski [1] in the context of restructuring a logical token ring network. These techniques are generic in nature and, therefore, can be used in various other classes of distributed algorithms. We will examine these strategies

5.3.3 Messaging cost

As obvious, one of the important complexity measures of distributed algorithms is the communication cost. It is dependent on the number of messages exchanged during one execution of the algorithm. But when hosts are allowed to become mobile, the communication complexity should also include the cost of location search. Location search includes the messages exchanged to find the location of a mobile host. We know that mobile hosts have severe power constraints. They can send messages only on wireless link which require them to expend power. So, the communication cost over a wireless link must be considered as higher in several orders of magnitude than the communication cost over the conventional wired link. So, Badrinath, Acharya and Imielinski proposed three cost measures for counting number of messages exchanged during execution of a distributed algorithms in a mobile computing environment. The cost model is as follows:

- $C_w \Rightarrow$ cost of sending a message from MH to BS over wireless channel (and vice versa)
- $C_f \Rightarrow$ cost of sending a message from a point to another point linked by the wired N/W.
- $C_s \Rightarrow$ cost of searching/locating a MH and forwarding a message to its current BS from a source BS.

A simplest search strategy to locate a mobile host is to let the searching base station query all other base stations and determine if a mobile host belongs to a cell under one of the set of queried base stations. The base station responding to the query is the one which has the mobile host in the cell under it. The querying base station can then forward the message meant for the mobile host to the responding base station. So, message exchanged for location search will be as under.

1. In the first round all base stations except one receive message from the querying base station. So total of $(N_{BS} - 1) \times C_f$ messages are exchanged.

2. The base station having the searched mobile host responds. This incurs a cost of C_f.

3. Finally querying base station forwards a message (data packet) to the responding base station. This incurs a cost of C_f.

Adding all three costs, the worst case cost for search is equal to $(N_{BS}+1)\times C_f$.

The cost of sending message from a mobile host MH to another mobile host MH′ involves following actions:

1. MH sends the message to its own base station BS. The cost incurred is C_w

2. BS then searches for the destination MH′ to find the base station BS′ under whose cell area MH′ is currently located and deliver the message to BS′. The cost incurred for this action is C_s.

3. BS′ sends the data packet to final destination MH′. An action incurring a cost of C_w.

Counting all the messages, we find that the cost of sending a message from a mobile host MH to another mobile host MH′ can be at most $2C_w + C_s$.

5.4 Two Tier Structure

As apparent from the cost model, a distributed system is viewed as a two tier structure, namely,

(i) a collection of mobile nodes, each of which may operate in connected or disconnected or doze mode, and a low bandwidth wireless connects it to the fixed network.

(ii) a network of fixed nodes each having more resources in terms of storage, CPU speed, power and bandwidth,

The guiding principle for structuring distributed algorithms for mobile applications is that communication and computing demands of an algorithm should be satisfied within static segment of the network to as much extent as possible which is already mentioned under principle 1. Some important justifications for this principle are:

- The message sent from a base station to a non local mobile host will incur a search cost. The same is true for exchange of messages between mobile hosts located in different cells. In order to reduce overall cost, the search cost must be reduced. So it is desirable to limit the communication between a mobile node and a fixed node locally within the same cell.

- Mobile nodes operate with battery power. It has to expend power for disk accesses, cpu operations and also for sending and receiving message over wireless links. On the top of this wireless channels offer significantly low bandwidth compared to wired links, the quality of these channels also depend on vagaries of weather and channel interferences of various types. So, transmission over wireless interface will involve packet loss, retransmissions and consequently large latency. Therefore, the number of wireless messages exchanged during execution of any algorithm should kept at a minimum even at the cost of a higher number of message exchanges over the wired network.

- A mobile node can operate in *disconnection* or *doze* mode. When a node operates in doze mode it shuts or slows down most of its system functions in order to reduce power consumption. It can listen to incoming messages in doze mode. In a disconnection mode, a mobile host operates autonomously and has no connectivity with the network. The major difference between two modes of operations is that a mobile host in doze mode is reachable from the rest of the system whereas in a mobile host in disconnection mode is unreachable. A node in doze mode may be forced into reconnection through paging and wakeup. As opposed to this a disconnection and subsequent reconnection are always initiated by the mobile host. However, if mobile hosts were to participate in execution of distributed algorithm then these nodes should operate in connected mode. Even a participating mobile host which neither initiated the computation nor requires the result is prevented from operating in doze or disconnection mode.

- Most distributed system employ a regular logical structure such a tree, ring or a grid among the participants to carry out the required communication. The purpose of providing a systematic structure of communication is to make it ordered and predictable among the participants. Messages exchanged will then follow only preselected logical paths. But if a participating mobile host decides to operate in disconnection mode during execution of a algorithm, then the communication structure has to be reconfigured resulting in additional overhead in terms of messages traffic and possibly search. A selected logical path for communication predefines a sequence of nodes through the messages would traverse from a given source to a given destination. So, all intermediate nodes on the path are prevented from operation in doze or disconnected mode.

Message Model, Mobility and Disconnection

It is assumed that a message will eventually reach destination mobile host MH possibly after search. There is no limitation on the number of moves that a mobile node can make. Let us assume that a FIFO channel exists from MH to local BS and another from BS to MH. A MH may leave a cell at any time. If $m_1, m_2, \ldots, m_s$ is the sequence of messages sent from BS to a MH, then the sequence of messages received at MH may be $m_1, m_2, \ldots, m_r$, where $r < s$. It implies that the messages received by MH from BS is only a prefix of the entire message sequence.

Mobility is exhibited by a host leaving the area of an old cell to entering into the area of a new cell. Each base station maintains a list of active mobile hosts. When a mobile host MH moves away from a cell under a base station BS to a cell under a new base station BS$'$, it sends *leave*(r) to BS, where r is the sequence number of the last message MH received from BS before leaving its cell. On receipt of *leave(r)* BS neither sends on BS-to-MH channel nor receives any from MH-to-BS channel. A MH on entering the cell under BS$'$

sends a *join*(MH, BS) to BS′. BS′ can then update its list of mobile hosts. Technically a *handoff* process is to executed for maintaining connectivity for a mobile host leaving an old cell and entering a new cell.

Disconnection, when voluntary, is same as leaving a cell. This is because MH operates autonomously in disconnection mode. The difference is that a MH leaving a cell will eventually become active in some adjoining cell. But a disconnected mobile host may not even surface. A MH deciding to operate in disconnected mode sends *disconnect(r)* to BS. BS deletes the MH but puts a disconnect flag on. When MH connects back it sends *reconnect*(*MH*, *BS*). Disconnection flag is removed as part of hand-off. MH may not always supply id of the base station at the time of reconnection, in that case a query must be executed to find the local base station. Query involving a disconnected MH is satisfied by a base station BS where the disconnect flag is set.

5.5 Mutual Exclusion

Consider the case of executing an algorithm over a set of participating mobile hosts. They need to cooperate and communicate for the progress in execution of the algorithm. As explained earlier, mutual exclusion ensures certain activity to be carried out only by one participant at a time. Therefore, mutual exclusion is fundamental to cooperation of processes in a distributed system, specially on the use of common or critical resources.

The mutual exclusion in a distributed system can be solved as follows.

- A participating node which wishes to use a critical resource, sends a request message to all other nodes.
- Waits for a reply from each.
- If the critical resource is not used presently by any other node, then use the resource.
- Sends a release to all after completing the access to the critical resource.

Suppose the above mutual exclusion to be executed on a set of N_{MH} mobile hosts. The cost of execution can be found by counting messages that are exchanged.

1. In the first step a mobile host sends $(N_{MH} - 1)$ request messages to its local base station for $N_{MH} - 1$ other mobile hosts. So the message cost for this step is $(N_{MH} - 1) \times C_w$

2. The receiving base station has to search and deliver the message to the local base stations of the respective mobile hosts. The cost for locating one particular mobile and delivering a message to it is $C_s + C_w$. So, the total cost $N_{MH} - 1$ mobile hosts is $(N_{MH} - 1) \times (C_s + C_w)$.

3. In the next step each of $N_{MH} - 1$ participants replies back to requesting mobile host. These participants collectively send $(N_{MH} - 1)$ messages through their local base stations. Each message requiring a search and delivery through local base station of the requesting mobile host. So the cost of this step is $(N_{MH} - 1) \times (2C_w + C_s)$.

4. Finally, after use of critical resource the initiating mobile host send release messages to $N_{MH} - 1$ other mobile hosts. The cost for sending this message is again $(N_{MH} - 1) \times (2C_w + C_s)$.

Adding all the costs together, the cost of satisfying a mutual exclusion request is

$$3(N_{MH} - 1) \times (2C_w + C_s)$$

Note that each request requires exchange of $6(N_{MH} - 1)$ messages on wireless channel. So the requirement of energy consumption at participating nodes is proportional to $6(N_{MH} - 1)$. The energy requirement at requesting host alone is $3(N_{MH} - 1)$. In addition to prohibitive cost, there are technical difficulties like reconfiguring set of active participants to permit some participants to operate in disconnected mode.

We can improve it by assuming that mutual exclusion is achieve by passing a token in a logical ring among the participating mobile hosts. The mutual exclusion is satisfied by having only mobile host that has the token, can access the critical resource. The token is passed from one mobile host to its successor in the logical ring after the former has used it. With this improvement the cost is reduced considerably. The analysis of cost of the above token ring algorithm, which we will refer to as TR-MH, is as follows.

> Both sender and recipient are mobile hosts. The message is sent first from sender to its local base station then from there to base station under which the recipient is found. So the cost of messages exchanged

on wireless links is $2C_w$. We know that the cost of locating a mobile host and subsequently sending a message to its current base station is C_s. Therefore the cost of passing of token between two adjacent MHs in the logical is $2C_w + C_s$.

Assuming the ring to consist of N_{MH} mobile hosts, overall cost of circulating token on the logical ring is $N_{MH}(2C_w + C_s)$. Note that the above cost is independent of the mutual exclusion requests satisfied by one complete circulation of the token in the logical ring.

Each exchange of message requires power both at the recipient and the sender. Every MH accesses wireless twice (once for acquiring and once for releasing) for one circulation of the token through it. So, energy requirement for executing this algorithm is proportional to $2N_{MH}$.

5.5.1 Logical Ring Using Two-Tier Principle

A logical structure of ring is maintained among the fixed hosts, i.e, base stations. A token is circulates over this ring in a predefined sequence. A mobile host MH wishing to access the token submits the request to its current BS. When the token becomes available, it is sent to MH at its current base station, say BS′. After using the token MH returns it to BS′ which in turn returns the same back to BS.

The cost of servicing token will depend also on the way location of a MH is managed. Badrinath, Acharya and Imielinski [1] had explored the token ring problem under two location management strategies, namely *search* and *inform*. They also investigated an variant of inform method where the token circulates among a small number of fixed hosts calle *proxies*. In this method the set of all base stations are partitioned into regions and associated a designated fixed host called *proxy* with each region. The number of proxies is much less than the number of base stations. The delivery of token to a MH will, therefore, require a search but within a small region under the current proxy of the MH. Essentially, a combination of search and inform is used to manage the location of a MH.

5.5.2 Search Strategy

It involves searching the entire area of coverage to find a migrant MH. The algorithm can be best understood by fragmenting the actions performed by the component devices, namely, base station and mobile hosts. Each base station maintaines two separate queues, request queue Q_{req} and Q_{grant}. Token access requests by mobile hosts at a basestation are queued up in Q_{req}. When the token is received by a BS from its predecessor all pending requests are moved into a different queue called grant queue Q_{grant}. Then all requests are serviced from Q_{grant} while new requests get added to Q_{req}. After all requests are serviced, the token is passed on to successor basestation in the ring. So, the actions of BS are as follows:

```
on receipt of a request for access of the token from $MH$
    add $MH$'s request to the rear of $Q_{req}$;
on receipt of the token from the predecessor in the ring {
    move all pending requests from $Q_{grant}$;
    **repeat** {
        remove request at head of $Q_{grant}$;
        **if** ($MH$ which made the request is local to BS)
            deliver the token to $MH$ over wireless link;
        **else**
            search and deliver token to $MH$ at its current cell;
        await return of token from the $MH$;
    } **until** ($Q_{grant}$ == empty);
    forward token to BS's successor in the logical ring;
}
```

As far as a mobile MH is concerned, it can request for token servicing at any time to its current basestation. Once token is received it uses the critical resource and returns token after doing so. So, the actions performed by a MH are as follows:

```
on requirement for an access of the token
    submit request to current local BS;
on receipt of token from local BS {
    hold the token and use the critical resource;
    return the token to local BS;
}
```

From the sketch of the algorithms, it is clear that at any time only one MH holds the token. Therefore, mutual exclusion is trivially guaranteed. The maximum number of requests that can be serviced at any base station is bounded by the size of the Q_{grant} at that base station when the token arrives. No fresh request can go to Q_{grant} as they are added only to Q_{req} So a token acquired by a base station will be returned after a finite time by servicing at most all the requests which were made before the arrival of the token. This number can not exceed N_{MH}, the total number of mobile hosts in the system. This means the token would eventually visit each base station in the ring.

The communication cost can be analyzed as follows:

1. Cost for one complete traversal of the logical ring is equal to $N_{BS} \times C_f$, where N_{BS} is the number of base stations.
2. Cost for submission of a request from a MH to a base station is C_w.
3. If the requesting MH is local to a BS receiving the token then the cost of servicing a request is C_w. But if the requesting MH has migrated to different base station BS$'$ before the token reaches BS where the request was initially made, then a location search will be required. So the worstcase cost of delivering token to the requesting MH will be $C_s + C_w$.
4. The worstcase cost of returning the token to BS delivering the token to MH is $C_w + C_f$. C_f component in cost comes from the fact that MH may have subsequently migrated from BS where it made request to the cell under a different base station BS$'$.

Adding all the cost components, the worstcase cost of submitting a single request and satisfying is equal to $3C_w + C_f + C_s$. If K requests are met in a single traversal of the ring then the cost will be

$$K \times (3C_w + C_s + C_f) + N_{BS} \times C_f$$

Since, the number of mobile host requesting a service is much less than total number of mobiles in the system. So $K \ll N_{MH}$.

In order to evaluate the benefit of relocating computation to fixed network we have to compare it with token ring algorithm TR-MH described earlier in this section where the token circulates among mobile hosts.

- *Power consumption.* The power consumption in present algorithm is proportional to $3K$ as only $3K$ messages are exchanged over wireless links. In TR-MH algorithm it was $2N_{MH}$. As $K \ll N_{MH}$ it is expected that $\frac{3K}{2N_{MH}} < 1$

- *Search cost.* For the present algorithm it is $K \times C_s$ whereas in the previous algorithm the cost is $N_{MH} \times C_s$ which is considerably more.

5.5.3 Inform Strategy

Inform strategy reduces the search cost. It is based on simple idea that the search becomes faster if enough footprints of the search object is available. In other words, search is less expensive if more information is available about the possible location of the mobile host being searched for. Essentially, the cost of search is proportional to the amount updated location information. For example, the search cost will be 0 if a MH to notifies the BS (where it submitted its request) after every change in its location till it has received the token. So, the difference between purely search based algorithm to inform based algorithm is that Q_{req} maintains the location area information along with request made by a mobile host, and every mobile host with a pending request informs change in location to the basestation where the request is made. The algorithm specifying the actions of a BS is provide below.

```
on receipt of a request from a local $MH$:
    add the request $< MH, BS >$ to the rear of $Q_{req}$;
on receipt of a inform($MH, BS'$) message
    replace $< MH, BS >$ in $Q_{req}$ by $< MH, BS' >$;
on receipt of the token from the predecessor of BS in logical ring {
    move $Q_{req}$ entries to the $Q_{grant}$;
    repeat
        remove the request $< MH, BS' >$ at the head of $Q_{grant}$;
        if ($BS' == BS$)
            deliver the token to $MH$ over the local wireless link;
        else
            forward token to $BS'$ for delivery to MH;
        await return of the token from $MH$;
    until ($Q_{grant}$ == empty);
    forward token to BS's successor in logical ring;
}
```

The actions to be executed by a mobile host MH are provided below.

```
on requirement of $MH$ needing an access to the token:
    submit request to its current local BS
```