International Journal of Advanced Technology in Engineering and Science
Vol. No.4, Issue No. 05, May 2016
www.ijates.com
ijates
ISSN 2348 - 7550

STRATEGY ALGORITHMS AND SERVICES WITH MOBILE COMPUTING

Sreedhar .M. Reddy

Scholar,Samskruti College of Engineering and Technology

(Affiliated JNTUH) Kondapur,Ghatkesar,Rangareddy (India)

ABSTRACT

Mobile Computing represents a new paradigm that aims to provide continuous network connectivity to users regardless of their location. Wide spectrums of portable, personal computing devices have recently been introduced in the market that range from laptop computers to handheld personal digital assistants. Coupled with the advent of wireless networking, this has given rise to a new style of computing wherein the computer can move with the user and yet maintain its network connections. The emergence of computer networking allowed isolated computers to share resources giving rise to distributed computing. Computer networking eliminated the physical isolation between computers, and enabled users connected to the system from one computer to use the resources available at another. This paved the way for distributed computing where a collection of networked computers cooperate to achieve a common goal. However, the access points to such a distributed system were still tethered machines and a user did not have the flexibility to move while accessing such a system. Mobile computing extends distributed computing in a direction where the services of such a system are available to an user regardless of location and more importantly, changes in location. Wireless networking eliminated the need to remain tethered to a wired, static infrastructure and yet avail of the services of a distributed system. We are initially proposing fundamental distributed algorithm: a logical ring with a token circulating amongst participants, and restructure it for servicing token requests from mobile hosts and tackle the problem of delivering a multicast message to mobile recipients from exactly-one location. Finally, we present a ensure pointing algorithm to evidence a reliable universal state of a distributed application executing on mobile hosts.

I. INTRODUCTION

Integration of mobile computers within existing data networks introduces new issues in the design of distributed algorithms and services. Location of a mobile host changes with time, and so the message count of a distributed algorithm should account for the "search" necessary to locate mobile participants. Further, mobile hosts are faced with resource constraints not commonly encountered by their tethered counterparts, viz. a low-bandwidth connection to the rest of the network, and tight restrictions on power consumption. This dissertation introduces a system model for networks with mobile hosts. To bridge the resource disparity between mobile and static hosts, we propose a two-tier principle for structuring distributed algorithms in this model. We also propose that location-management of mobile participants be integrated with algorithm design. A distributed system with mobile hosts thus consists of a wired infrastructure of static hosts (representing a conventional distributed system) that connects areas of wireless coverage (’cells’) to mobile computers. Mobility of computers is not

obtained for free. The ability to be mobile introduces a set of new constraints that are not associated with the desktop computers. The goal of mobile computing is therefore, to provide users with a comparable level of service as would be available to them from a distributed system of static computers, without compromising their ability to move. In systems with static hosts, connectivity of the underlying network does not change in the absence of link and/or host failures. 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 to prolong battery life. The communication between a mobile host and the rest of the network occurs via a wireless medium with a significantly lower bandwidth than wired links; additionally, message transmission and reception at a mobile host consumes battery power, which is a critical resource. Lastly, mobile computers utilize processors and storage systems that trade computing, communication speed and storage capacity for reduced power consumption and portability (weight and size). Thus, though a mobile computer can be taken anywhere, it delivers performance an order of magnitude less than their desktop counterparts. These aspects are characteristic of mobile computing and need to be addressed before distributed systems can be extended to include mobile hosts.

This considers how distributed algorithms and services should be structured for mobile hosts. A fundamental problem in distributed systems is that of providing mutually exclusive access to a shared resource to a group of competing entities such that there is no centralized permission granting authority. In the con-text of a mobile computing environment, the competing entities are mobile hosts. We show the drawbacks of applying existing solutions (designed implicitly for static hosts) to a distributed system with mobile hosts, and propose a two tier principle to structure distributed algorithms for mobile hosts, taking into consideration the unique resource constraints of this environment. .The first solution that we present allows a multicast message to be addressed to any arbitrary group of mobile hosts. We then refine this solution to allow multicasting only to pre-defined groups of mobile hosts; we present a protocol to track the location of group members and this aggregate location information determines which locations should receive a copy of a multicast message addressed to a specific group. The third section of this thesis focuses on the problem of recording a consistent global checkpoint of a distributed application executing on mobile hosts.

1.1 Related Work on Mobile Computing

Given the relative infancy of mobile computing, it is not surprising that there has been little prior work on structuring distributed algorithms per se for mobile hosts. Research has primarily focussed on providing network-layer protocols to route data packets to mobile hosts regardless of location, and on data management issues for low-powered clients that access data stored within the static infrastructure using a low-bandwidth connection. Below, we summarize current research in this field: Extending IP protocols to handle host mobility Hosts in the Internet are associated with addresses, which determine the route a data packet takes to reach a particular destination. In effect, the address determines the “location” of a host vis-a-vis the rest of the network. However, with mobile hosts, this is no longer valid since the location of a mobile host changes. If the address associated with a mobile host remains the same regardless of its location, then this address can no longer be used to route a packet to it .On the other hand, if a mobile host is assigned a new address reflecting its current location after every move, then all other entities in correspondence with this host need to be informed of

changes in its address.Effects of host mobility on transport and higher layers one of the aims for providing network-layer support for mobile hosts is that transport-layer protocols do not need to be aware of host mobility, i.e. Provide functional trans-parency to TCP/IP. However, a recent study shows that active TCP connections, with Mobile IP at the network layer, show considerable degradation in perfor-mance: mobility of a host from one location to another increases delays and packet losses while the network learns how to route data to the host's new location. TCP interprets these events to be a result of network congestion and throttles further packet transmission, thereby leading to further drop in throughput. As a remedy, that TCP be made aware of host mobility. In addition to the trans-port layer, that host mobility also be made explicit at the application layer as well. File systems for mobile users.The primary motivation behind designing file systems for mobile users is to support disconnected operation. Prior to disconnection, a user caches files from static servers that (s) he needs to use while disconnected from the network. When the user reconnects, modifications to the files that were updated by the user in disconnected mode are propagated to the server. The three main issues here are: (1) hoarding, i.e. which files should be brought into the mobile client's local store prior to disconnection, (2) emulation, i.e. satisfying file requests from the local cache while disconnected, and what actions to be taken if such a request cannot be satisfied, and (3) reintegrating all modified files at the mobile client with the static server, on reconnection. Another motivation for designing file systems to handle mobility is to minimize synchronous operations and allow mobile clients to determine the level of consistency desired between the client copy and copies at the server(s). A third approach is the design of the storage subsystem at the mobile computer using flash memory to reduce power consumption .Operating systems Research in this area has been fueled by tight constraints on power consumption at mobile computers.It presents enhancements to the Unix kernel for power management and checkpointing system state (to protect the user from undesirable state loss when the battery is low). Guided by limitations of wireless bandwidth and power, [9] proposes an inter-process communication mechanism that provides (a) an agent within the static infrastructure that stores and filters messages on behalf of a mobile user (b) a hierarchical structure to messages, with immediate-delivery data at the top, and (c) the ability to automatically start/resume a server program at the mobile host on delivery of a message (to the server). Data Management Challenges to the database field due to the advent of mobile computing were namely controlling the flood of location updates arising out user mobility and the need for a new paradigm for answering queries where acquisition of location information is integrated with the query evaluation process. The issue of optimizing queries from a mobile computer to a database server on the static network is discussed in [5]: this involves optimization

II. IMPACT OF MOBILITY

We now investigate why it is inefficient to execute classical distributed algorithms at mobile hosts, that are neither aware of host mobility nor of the resource constraints associated with such hosts. Algorithm L-MH We first look at the drawbacks of executing Lamport's algorithm directly at N mobile hosts, which we refer to as algorithm L-MH. 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, waits for a reply message from each of them, and then sends a release message to all others after completing its access to the critical region. Each participant is required to a 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 (Nmh-1)(2Cwireless+Csearch). 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(Nmh-1). The energy consumed at an initiator is proportional to 3(Nmh- 1), while each of the other (Nmh-1) MHsconsume 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 algorithm L-MH, 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 each pair of MHs, regardless of their location in the network.

-Doze and disconnected modes. Algorithm L-MH requires the participation of every MH in every execution of the algorithm. Consequently, it 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.

Algorithm R-MH Consider an execution of Le Lann's algorithm where each participant is a mobile host and the logical ring comprises of all the MHs in the system. We will refer to this algorithm as R-MH. Algorithm R-MH is the extreme case of executing an existing distributed algorithm at the mobile hosts. It shows that correctness of an existing algorithm, which did not explicitly consider host mobility, is not compromised when applied to the mobile environment. However, such an approach is not sensitive to the resource constraints specific to the mobile environment and consequently suffers from the following drawbacks:

- High search cost -Each message in the algorithm is addressed to a mobile host and therefore, incurs a search cost. Since the algorithm circulates the token amongst all MHs, the overall search cost incurred by the algorithm is proportional to Nmh.

- Excessive usage of wireless links Both sender and destination of each message is a MH; the message is thus transmitted over the wireless links between the respective local MSSs of both the sender and destination MHs. Therefore, the wireless cost component of every message in this algorithm is 2 wireless.The cost of each message (to pass the token between two adjacent MHs in the ring) is thus2Cwireless+Csearch and the overall cost of the algorithmis Nmh (2Cwireless + Csearch). Note that this cost is independent of thenumber of mutual exclusion requests satisfied in one traversal of the ring. - Power consumption at MHs The wireless component of the overall cost is indicative of the power consumed at MHs for transmission and reception of messages. Each message in this algorithm consumes power at both the sender (to transmit it to the local MSS) and the destination (to receive the message from its local MSS). Thus, an execution of the algorithm requires every MH to expend power for accessing the wireless link twice, and the cumulative power consumption for one execution of the algorithm is propor-tional to 2Nmh.Doze and disconnected modes. Algorithm R-MH requires the participationof every MH to maintain the logical ring and cannot therefore permit

any MH to disconnect. To allow disconnections, the logical ring will need to be reconfigured amongst the remaining participants. Secondly, a MH operating in doze mode, is forced to resume normal operation on receipt of the token from its predecessor in the ring. It may revert to doze mode after forwarding the token to its successor. Thus, algorithm R-MH does not allow a MH to operate in doze mode without interruption even though it does not need to access the shared resource (represented by the token); it must still receive and forward the token to enable other MHs to access the resource.

As an illustration of the above drawbacks, consider a logical ring among three mobile hosts h1, h2 and h3 shown in. In our system model, a single logical link in the ring, e.g. between h1 and h2, consists of: (1) the wireless connection between h1 and its ‘‘current’’ local MSS, MSS1, (2) a logical point-to-point connection between MSS1 and MSS2, the local MSS of h2 and (3) a wireless connection between h2 and MSS2. Now, if h1 changes its location and moves to the local cell under MSS2, then the logical links of the ring between the three mobile hosts need to be re-mapped. This is manifested as search cost when h3 forwards the token to h1. Next, assume h2 is presently operating in doze-mode; it is forced to resume its regular operating mode on receipt of the token from h1 and then forward the token to h3;h2 itself may not have any use for the token. In addition to doze-mode of operation, the logical ring is also affected by disconnections.

Specifically, the disconnection protocol should ensure that the logical structure can be re-established amongst the remaining participants.Prior to physically detaching itself, it can inform its immediate predecessor in the ring (h1) of its intent to disconnect; h1 can then permit h2 to disconnect after establishing a logical link directly to h3. The logical ring now consists of the remaining participants, viz. h1 and h3. Thus, disconnection causes deletion and addition of logical links from the ring and thereby a reconfiguration of the physical links; the set of participants changes as well. In contrast, a change in location of a mobile host causes only a reconfiguration of the physical links comprising the logical ring We remedy these drawbacks below by applying Lamport’s and Le Lann’s algorithm to the mobile environment in accordance with the two-tier principle.

III. STRUCTURING A TOKEN-BASED LOGICAL RING FOR MOBILE HOSTS

The two-tier principle suggests that the logical ring should be established within the fixed network. The logical ring now consists of all MSSs with the token visiting each MSS in a predefined sequence. A MH that wishes to access the token is required to submit a request to its local MSS. When the token visits this MSS, all pending requests are serially serviced. However, a MH may have changed its location since submitting its request, and therefore, the location of such migrant MHs need to be explicitly managed. Below, we present two location management strategies, viz. search and inform, for the case when all MSSs constitute the logical ring. An alternative method of structuring the ring is to partition the set of all MSSs into “areas” and associate a designated fixed host, called a proxy, with each area. The token now circulates only amongst the proxies, and each proxy is responsible for servicing token requests from MSSs within its area. A combination of search and inform strategies is used to manage the location of migrant MHs in this case. SEARCH Strategy Algorithm R-MSS:search maintains a unidirectional logical ring amongst the MSSs, and the token circulates in this ring. It consists of two distinct interactions: one, that is executed solely within the static segment to circulate the token amongst the MSSs, and the other takes place when a MH submits a request for the token to its local MSS. Actions executed by a MSS M On receipt of a request for the token from a local MH, M adds the request to the

rear of its request queue. When M receives the token from its predecessor in the logical ring, it executes the following steps:

a. Pending requests from M's request queue are moved to the grant queue.

b. Repeat

• Remove the request at the head of the grant queue

• If the MH making the request is currently local to M, then deliver the token to the MH over the wireless

Ensuring fair access to the token regardless of mobility In the three search strategies discussed so far, there are two entities whose location varies with time: (a) the token, and (b) MHs. This allows for a situation where a MH accesses the token at its current cell, moves to a cell under a MSS that is the next recipient of the token in the logical ring, and accesses the token again. Thus, a MH by virtue of its greater mobility (compared to a stationary MH) could access the token multiple times in one traversal of the ring by the token, while a stationary MH can access the token at most once. If "fairness" of access amongst all MHs, independent of mobility, is a desirable goal, then additional synchronization mechanisms need to be built into the algorithm to achieve this goal. Note also that algorithm R-MH, which maintains the logical ring amongst MHs, allows fair access to the token. Therefore, when we establish the logical ring within the fixed network, we need to also preserve the functionality of algorithm R-MH by providing a scheme that will ensure fair access to the token regardless of a MH's mobility. As an example, consider two MHs h1 and h2 initially located in the same cell under MSS M1. Let MSS M2 be the successor of M1 in the logical ring. If both MHs contend for the token at M1, then both requests will be satisfied when the token reaches M1. Now, assume h2 moves to the cell under M2 and contends for the token. If its request is received at M2 prior to the token reaching M2 from M1, then h2 can access the token for a second time. In contrast, if h1 continues to remain in M1's cell, it can again access the token only after the token revisits M1 after completing one traversal of the ring.

IV. SIGNIFICANT AND NON-SIGNIFICANT MOVES

The host-view membership protocol for HG is triggered whenever a MH h in G switches its cell. Let h move from the cell under MSS M to that under N. M necessarily belongs to HG, since h was earlier located in its cell. Depending on whether N belongs to HG, we distinguish between two types of moves:

- If N does not belong to HG, then it is a significant move, since h's entry to its cell will modify HG. Otherwise, it is a non-significant move.

The two types of moves are handled differently. For a non-significant move, by virtue of being a part of HG, N is already in a position to provide the multicast service to any MH in G that may enters its cell. However, for a significant move, N has to first transfer relevant "state" from other members of HG before it can provide a multicasting service to G. A non-significant move will therefore incur a substantially lower overhead in terms of data transferred within the static network, compared to a significant move.

The data structure that is closely tied with the current incarnation of HG is h_RECDG[ ]. In case of a non-significant move, some incarnation of HG is present at both M and N; let the respective current incarnation numbers be m and n. In this case, M transfers h_RECDGm[ ] to N during handoff. However, since n may be different from m, h_RECDGm[ ] may need to be suitably modified so that it corresponds to the nth incarnation. For a significant move, both HGm and h_RECDGm[ ] needs to be transferred from M to N since N does not

presently belong to HG. More importantly, a significant move requires participation of other MSSs in HGm as well:

- M sends an update message to all MSSs in HGm to include N in the host- view (resulting in HGm+ 1).
- Each MSS is then required to transfer a copy of the pending multicast messages initiated by it, to N. The motivation is that once N has been added to HG (by incurring the overhead of a significant move), further moves by MHs in G to its cell, can then be handled as a non-significant move.

V. HOST VIEW MEMBERSHIP PROTOCOL

For each multicast group in the system, a separate host-view membership protocol (HVMP) is executed to maintain their respective host-views. The protocol is described below with reference to a multicast group G.

5.1 Role of the coordinator

The only function of the coordinator CG is to serialize concurrent changes to G's hostview so that the local copies of HG progress in the same sequence. It maintains an incarnation counter for G, and when requested by a MSS for a new incarnation number, it returns the current value after incrementing the counter.

5.2 Responding to changes in host-view

Let L be a MSS which belongs to G's host-view, i.e. G is listed in L_groups (the list of all groups whose host-views include L), and let its current incarnation of HG be l. L is informed of changes to HGl through view_change() messages sent from other MSSs. A view_change() message contains the group-id G, the incarnation number k of the change, and the MSS that is added or deleted from HG. A view_change(G, k, <change>) message is processed as follows:

1. If l < (k - 1), i.e. L has not yet received view_change() messages with incar- nation numbers in the range l + 1, . . ., k - 1, then delivery of the message

5.3 Departing from the host-view

A MSS N departs from HG by first obtaining an incarnation number k from CG. When its current incarnation number rises to k - 1, it sends a view_change(G, k, delete(N)) to all MSSs in HGk-1, deletes all data structures associated with G, and is no longer deemed to belong to HG. However, till the incarnation number at all MSSs in G's host-view rises to k, they will continue to send multicast and HVMP messages to N; these messages are simply discarded by N, once it has sent the view_change() message. Should N rejoin the host-view in the meantime due to a (significant) move by a MH h, N will begin to buffer these messages after sending the transfer(h, G, ?) message. If N rejoins HG in the nth incarnation (on reception of handoff(h, G, n-1, _ , _)), then N will retain a view_change() message for later delivery only if its associated incarnation number is greater than n. For each MSS in HGn, any multicast-related message (i.e, multicast() or delete()) received by N prior to receiving the pending multicast messages (and a ack_add() message) from the MSS, is deleted.

N may depart HG when the following conditions are satisfied:

1. No MH local to N's cell is a member of G and the handoff process associated with the departure of any such MH has completed. This also implies, that ack_list(m) associated with each message m in N_BufferG, is empty i.e., the list of MHs that have acknowledged receipt of m, has been forwarded to m's initiator3.

2. N_mcastG is empty i.e., delivery of all multicast messages to G, that were initiated by N, has been completed.

3When the last MH that belongs to G, leaves the cell, HVMP will notify the HANDOFF module; it is the HANDOFF module that is actually responsible for forwarding all non-empty ack_lists (step GD1 in section 4.4.2). It is possible that, after requesting CG for a new incarnation number k and before receiving this number, a MH belonging to G enters N's cell. In this case, N will rescind its decision to leave HG by sending a dummy message view_change(G, k, ?) to all MSSs in HGk-1. Although N is allowed to depart HG when the above two conditions are satisfied, correctness of HVMP is not sacrificed if the departure of N is deferred for a time period Tdepart. Recall that when a MSS joins HG, it receives a copy of the pending multicast messages initiated by every other MSS in HG. Therefore, this overhead is avoided if any MH that belongs to G enters N's cell during this interval Tdepart, since N continues to be a part of HG and the MH's move to N will be treated as a non-significant move. On the flip side, if no MH enters N's cell during Tdepart, then N will needlessly be sent a copy of every MCAST and view_change() message. Thus, a low value of Tdepart could lead to N leaving HG and then joining it again, while a high value of Tdepart could lead to useless propagation of multicast and view_change() messages. To determine a balanced estimate of Tdepart from experimental studies, mobility patterns of a multicast group needs to be investigated.

5.4 Forming a multicast group

The initial creation of a multicast group is not a part of HVMP per se. It assumes that the initial incarnation of G is available at the MSSs comprising HG0, using a separate protocol; the function of HVMP is to update HG in a consistent manner. Below, we present some approaches for creating a group G:

1. A MSS M, initiates creation of G, possibly on behalf of a local MH, by sending a init(G) to all MSSs within a specified scope, e.g. a campus. A MSS, on receiving this message, queries its local MHs if they wish to join G; ids of all interested MHs are sent back to M which forms EG. The set of MSSs that each have a local MH in EG, forms HG0.

2. A MSS executes the at least-once delivery algorithm described earlier, to multicast a invite(G) message to a set of specified MHs. A recipient MH either sends a yes or no reply to its local MSS. Instead of ack_list(), the local MSS sends back a list of yes/no answers to the initiator MSS. All MHs that

5.5 Message delivery to a multicast group G

We now describe the protocol to deliver multicast messages to G. As in the exactlyonce protocol of section 5.2.5, there are three modules: WIRED, HANDOFF and WIRELESS module. The WIRELESS module plays the same role, viz. maintaining fifo links between a MH and its local MSS, and will not be described again. Most of the responsibility of the handoff process is now borne by HVMP; therefore, the HANDOFF module is considerably simplified.

VI. WIRED MODULE

To multicast m to G, a MSS M initiates the protocol as follows:

GA1.M maintains a local counter for every group G in M_groups. The counter for G is incremented and assigned as m_seq. M then sends multicast(G, m) to all MSSs in HGk, where k is its current incarnation of G's host-view. GA2.m is also added to M_mcastG, and dests(m) is initialized to EG.

GA3.Recipient MSSs reply with respective ack_list(m) messages. M deletes a MH from dests(m) if it is included in a ack_list() message. When dests(m) becomes empty, a delete(G, m) message is sent to all MSSs in M's current incarnation of HG, and m deleted from M_mcastG. On receiving multicast(G, m), a MSS N executes the steps below. Let the current incarnation of HG at N be n.

GB1.m is appended to N_BufferG; a list dests(m), initialized to EG along with another empty list, ack_list(m) are associated with m.

GB2.for each MH h in N_LocalG if (h 2 N_Local) and N_Transmit does not contain an entry <G, m', h>

/* there is no message m' addressed to G, awaiting delivery to h, and h has not left the cell*/ then if (h_RECDGn[M] < m_seq) then insert <G, m, h> in N_Transmit else delete h from dests(m) .GB3.When N receives delete(G, m), m is deleted from N_BufferG. The WIRELESS module delivers messages from N_Transmit to the target MH. When h acks receipt of m to the WIRELESS module, the WIRED module at N is notified and it executes as follows:

GC1.h_RECDGn[M] := m_seq

GC2.h is added to ack_list(m), and deleted from dests(m). GC3.check_ack_list(m, G)4

GC4.if (h 2 N_LocalG) and (h 2 N_Local)

then /*h has not left the cell; so deliver the next applicable message in N_BufferG to h*/

/* Let the sequence of messages in N_BufferG from the front to the rear, be m1, m2,. . .mg, where g is the number of messages in N_BufferG; let m occupy the kth position in this sequence.

for i := k+1 to g

if (h_RECD[init_mi] < mi_seq)

4 The procedure check_ack_list(m,G) is defined as:

if (dests(m) T N_LocalG = ;) and ack_list(m) = ; then send a copy of ack_list(m) to init_m

ack_list(m) := ;

then

add <G, mi, h> to Transmit queue; — exit for loop

else — h is deleted from dests(mi) — check_ack_list(mi, G)

VII. CONCLUSIONS

"Location" of a MH has been identified with its current "cell", e.g. HVMP maintained a group's host-view as a set of cells where at least one member of the group was located. However, both for the exactly-once protocol and HVMP, location of a MH could be maintained at a coarser granularity, e.g. a set of adjoining cells and the entire host-view maintained in a hierarchical manner. In addition to reducing location updates, it could also reduce the transmission overhead on a initiator (to send a copy of a multicast message and its corresponding delete() message to all MSSs in the host-view); with a hierarchical scheme, the transmission overhead is distributed amongst multiple MSSs at the expense of a higher latency for each message to traverse the multiple levels of the hierarchy. Alternatively, instead of using point-to-point message delivery between MSSs, a (best-

effort delivery) multicasting service provided at the network layer [15, 28, 56] could be used to efficiently route copies of multicast, view_change() or delete() messages amongst the MSSs comprising a host-view, with additional transport-layer mechanisms to ensure reliable and sequenced delivery. The scope of this work was restricted to handling host-view changes due to mobility of group members. However, addition and/or deletion of members (MHs) could also result in a change of host-view. To incorporate these changes into HVMP would require propagating changes to EG to all MSSs in HG as well

BIBLIOGRAPHY

[1] Acha, A.,. An efficient protocol for ordering broadcast messages in distributed systems. In The 3rd IEEE Symposium on Parallel and Distributed Processing (1991).

[2] Badrinath, B. R. Delivering multicast messages in networks with mobile hosts. In Proc. of the 13th Intl. Conf. on Distributed Computing Systems (May 1993).

[3] Agrawal, D., and Abbadi, A. E. An efficient and fault-tolerant solution for distributed mutual exclusion. ACM Transactions on Computer Systems 9, 1 (Feb 1992).

[4] Venkatesan, S. Tolerating mobile support station failures. Tech. rep., University of Texas at Dallas, November 1993.

[5] Ganguly, S. Query optimization for energy efficiency in mobile environments. In Proc. of the 1993 Intl. Workshop on Foundations of Models and Languages for Data and Objects.

[6] Alonso, R., and Korth, H. Database system issues in nomadic computing. Tech. Rep. TR-36-92, MITL, December 1992.

[7] Korth, H. Database system issues in nomadic computing. In Proc. of the ACM SIGMOD Intl. Conference on Management of Data (June 1993).

[8] Moser, L. E., Melliar-Smith, P. M., Agarwal, D. A., and Ciarfella, P. Fast message ordering and membership using a logical token-passing ring. In Proc. of the 13th Intl. Conf. on Distributed Computing Systems (May 1993).

[9] Duchamp, D. Agent-mediated message passing for constrained environments. In USENIX Symposium on Mobile and Location-Independent Computing (Aug. 1993).

[10] Peleg, D. Concurrent online tracking of mobile users. In Proc. ACM SIGCOMM Symposium on Communication, Architectures and Protocols (September 1991).

[11] Imielinski, T. Impact of mobility on distributed computations. ACM Operating Systems Review 27, 2 (April [12] Imielinski, T., and Marantz, R. Handling mobile clients: A case for indirect interaction. In Fourth Workshop on Workstation Operating Systems (WWOS-IV) (Oct. 1993).

[13] Badrinath, B. R., and Imielinski, T. Replication and mobility. In Proc. of the 2nd workshop on the management of replicated data (1992), pp. 9-12.

[14] Virmani, A. Locating strategies for personal communication networks. IEEE Globecom 92 Workshop on networking of personal communications applications (1992).

[15], Francis, P., and Crowcroft, J. Core Based Trees(CBT) an architecture for scalable inter-domain multicast routing. Submitted for publication.