A Reliable Multicast Protocol for Distributed Mobile Systems: Design and Evaluation

Giuseppe Anastasi, Alberto Bartoli*, Francesco Spadoni

Dipartimento di Ingegneria dell'Informazione, Università di Pisa

Via Diotisalvi 2, 56126 Pisa, Italy

Fax: +39-050-568522, E-mail: anastasi@iet.unipi.it

** Dipartimento di Elettrotecnica, Elettronica ed Informatica, Università di Trieste*

Via Valerio 10, 34100 Trieste, Italy

Fax: +39-040-6763460, E-mail: bartolia@univ.trieste.it

Corresponding author

Giuseppe Anastasi

Università of Pisa - Dipartimento di Ingegneria dell'Informazione

Via Diotisalvi 2 - 56126 Pisa

Italy

Email: anastasi@iet.unipi.it

Tel.: +39 050 568 559

Fax: +39 050 568 522

A Reliable Multicast Protocol for Distributed Mobile Systems: Design and Evaluation

Giuseppe Anastasi, Alberto Bartoli*, Francesco Spadoni

Dipartimento di Ingegneria dell'Informazione, Università di Pisa, Via Diotisalvi 2, 56126 Pisa, Italy
Fax: +39-050-568522, E-mail: anastasi@iet.unipi.it

** Dipartimento di Elettrotecnica, Elettronica ed Informatica, Università di Trieste, Via Valerio 10, 34100 Trieste, Italy*
Fax: +39-040-6763460, E-mail: bartolia@univ.trieste.it

Abstract *Reliable multicast is a powerful communication primitive for structuring distributed programs in which multiple processes must closely cooperate together. In this paper we propose a protocol for supporting reliable multicast in a distributed system that includes mobile hosts and evaluate the performance of our proposal through simulation. We consider a scenario in which mobile hosts communicate with a wired infrastructure by means of wireless technology. Our proposal provides several novel features. The sender of each multicast may select among three increasingly strong delivery ordering guarantees: FIFO, Causal, Total. Movements do not trigger the transmission of any message in the wired network as no notion of hand-off is used. The set of senders and receivers (group) may be dynamic. Size of data structures at mobile hosts, size of message headers, number of messages in the wired network for each multicast, are all independent on the number of group members. The wireless network is assumed to provide only incomplete spatial coverage and message losses could occur even within cells. Movements are not negotiated and a mobile host that leaves a cell may enter any other cell, perhaps after a potentially long disconnection. The simulation results show that the proposed protocol has good performance and good scalability properties.*

Keywords: Mobile Computing, Reliable Multicast, Dynamic Membership, Causal Ordering, Total Ordering

1 Introduction

Recent technological trends have greatly stimulated distributed systems research and practice towards the support for *mobile* hosts. Laptops, cellular telephones, wireless networks, palmtop computers, wearable devices, allow users to carry computers to where their presence is actually needed. Not only this functionality may greatly extend the scope of existing applications of distributed systems, it may also enable novel application domains in such fields as inventory control, factory automation, on-site data collection, traffic monitoring and so on.

In this paper we propose a protocol for *reliable multicast* communication within a group that may include mobile hosts. By reliable multicast we mean, very informally, that all multicasts are delivered and there are no duplicates. Reliable multicast is a communication primitive that has proven its utility in the context of stationary distributed computing, in particular, where the application requirements imply a tight cooperation among a number of remote entities that must maintain some form of shared state. Our protocol extends the applications of this paradigm towards distributed mobile systems.

Mobile hosts typically need a special treatment, for a number of reasons. Traditional network protocols implicitly assume that hosts do not change their physical location over time **[BPT96]**. Mobile devices have severe resource constraints in terms of energy, processing and storage resources **[MDC93, FZ94]**. Wireless networks are characterized by limited bandwidths and high

error rates **[ES96]**. Furthermore, mobility introduces new issues at the algorithmic level. For example, a mobile host may miss messages simply because of its movements, even with perfectly reliable communication links and computers that never crash **[AB96]**.

Our scenario consists of a distributed system in which mobile hosts (MHs) communicate with a wired infrastructure through a number of spatially limited cells that define wireless links. Cells provide only *incomplete* spatial coverage and wireless communication is *unreliable*. Movements are unpredictable in the sense that a MH may leave a cell without prior negotiation and then it may re-enter *any* other cell, perhaps after remaining out of coverage for some time. We do not assume any support for routing messages to a MH, i.e., we do not rely on Mobile IP **[P96]**. The resulting scenario is quite general because it accommodates contemporary wireless LAN's, infra-red networks requiring line-of-sight connectivity, physical obstructions, picocellular wireless networks where cells coincide with rooms in a building **[GS94]**, disconnected modes of operation, long-range movements.

The proposed protocol has the following salient features:

- Multicast communication is reliable (all multicasts are delivered and there are no duplicates).
- The sender of each multicast may select among three increasingly strong delivery ordering guarantees **[HT93]**: *FIFO* (i.e., single-source ordering); *Causal* (i.e., deliveries occur in an order consistent with the "happened-before" ordering of transmissions **[L78,PRS97,AV97]**); *Total* (i.e., all group members deliver multicasts in the same order and this order is consistent with causal order).
- The set of senders and receivers (*group*) may be *dynamic*: a mobile host may join and leave such group at will of the application.
- Cell switchings of MHs do not trigger any message exchange in the wired network. This feature allows supporting efficiently a large number of MHs that frequently switch between cells **[MAO92,GS94].** The ability to accommodate neatly such a scenario is gaining importance due to the current trend toward smaller cells, which is motivated by their

advantages in terms of improved aggregate throughput and smaller power required for transmitting **[CP98]**.

- Size of message headers, size of data structures at MHs, number of messages in the wired network for each multicast do not depend on the number of group members. These factors contribute to make the protocol highly scalable.

To the best of our knowledge, previous reliable multicast protocols for distributed mobile systems **[AB96,PRS97,AV97,YHH97,ARR97]** offered at most two of the following features, supported by our proposal: (i) incomplete spatial coverage; (ii) unreliable wireless communication; (iii) dynamic membership. None such protocol was able to accommodate cell switching without any message exchange in the wired network and none supported the three delivery guarantees FIFO, Causal, Total.

We have analyzed in detail the performance of the proposed protocol by simulation. We anticipate that the protocol has good performance and good scalability properties. For example, we shall see that the delivery latency with a fixed number of senders remains approximately constant up to a large number of receivers and that this latency is limited only by queuing times at "Mobile Support Stations", a factor that follows from the mismatch between the wired and wireless bandwidth and thus it is not a peculiarity of our protocol.

2 Overview and related work

2.1 Overview of the protocol

Our protocol is described in section 4. We present an overview here as a background for the next subsection where we discuss related work in this area. By *stationary host* (SH) we mean a host connected to the wired network.

In our protocol, new multicasts originated by MHs have to be processed by a small subset of SHs, the *coordinators*, that attach proper control information to these multicasts. The details of such processing and of the related control information depend on the delivery guarantee selected by the sender. Coordinators send the multicasts to *mobile support stations* (MSSs), i.e., SHs connected

to the wired network and covering a cell each. MSSs broadcast messages from coordinators in the respective cell.

MHs may miss multicasts, for any of the following reasons: (i) they roam out of coverage; or (ii) the wireless link looses the message; or (iii) they move at inopportune times, e.g., they switch cell before a multicast is broadcast in the old cell and after it has been broadcast in the new one. MHs detect missing multicasts and ask retransmissions to MSSs by means of a negative acknowledgment scheme. MSSs cache copies of past multicasts and, if a retransmission request cannot be satisfied with the cache content, they fetch the pertinent messages from coordinators, that store a copy of each past multicast that might have not been delivered by all group members. MHs piggyback acknowledgments for past multicasts into retransmission requests and MSSs forward such acknowledgments to coordinators, which enable coordinators to free storage resources as appropriate (see section 4.5 for additional methods for propagating stability information). Observe that MSSs do not interact between themselves upon cell switchings of MHs.

The proposed protocol is an extension of a protocol previously developed by one of the authors that was based on a single coordinator and provided only Total ordering **[B98]**. A simplified version of the proposed protocol, without dynamic membership, has been presented in **[ABS99b]**.

2.2 Related Work

Multicast to MHs could be performed by means of multicast protocols designed for wired networks and stationary hosts, provided that routing support for MHs is available. We decided not to rely on network-level mechanisms for routing messages to a MH, e.g., Mobile IP, for several reasons. First, tracking the location of individual MHs is costly, both in terms of latency and of additional traffic on the wired network, as each cell switching requires updating the distributed routing information. Second, such a strategy would make it more difficult to exploit the broadcast capabilities of the wireless medium when many MHs belong to the same cell. Third, cell switchings would generate traffic in the wired network even while no new multicasts are

generated. Fourth, we allow MHs to remain out of coverage for a time that is "long" if compared to the timeouts used at the transport level. Finally, the problem of recovering from missing messages would remain. Multicast protocols that rely on Mobile IP are generally targeted at different application domains and provide unreliable, best-effort, unsequenced delivery **[XP97, CWBM98]**. In particular, no messages are delivered during a cell switching and messages possibly lost will not be recovered in the new cell. In such framework, lost messages may be recovered by means of usual recovery schemes, e.g., NACK and retransmission **[PSLB97, PGM00]**. To reduce retransmissions Forward Error Correction (FEC) techniques can be used. The idea of FEC is to get transmission right the first time. A sender that multicasts a message transmits also some extra data (redundant data), either in the same message or in a separate one. At the receiver side the extra data are used to recover from possible communication errors and/or message losses. If extra data are not sufficient for recovery, the missing message is required by means of traditional schemes **[RV98, WT00, PGM00]**.

Rather than hide mobility and wireless links, we adopted an *indirect* approach, i.e., one in which: (i) the two portions, wired and wireless, of the communication path are treated differently; and (ii) MHs and SHs are not considered on par with each other. According to the indirect model, a MH wishing to reliably multicast a message will send the payload to the MSS of the cell where it happens to be located and the MSS will multicast the message on behalf of it. It has been shown that indirect protocols may offer better performance and more flexibility in mobile wireless environments **[BB97]**.

The first example of indirect protocol for reliable multicast was the protocol proposed in **[AB96]**. It supports *static* group membership and *FIFO-ordering* of deliveries (see also **[YHH97]** for clarifications about this ordering) and appears to assume *complete* spatial coverage. This protocol maintains on the wired network chains of pointers along which messages have to be forwarded to the current location of each group member. When a MH switches between cells, a dedicated *hand-off* procedure updates these pointers and moves the messages not yet delivered from the old

MSS to the new one.

Our protocol does not maintain any chain of forwarding pointers in the wired network and, consequently, it does not employ any notion of hand-off. This feature is highly desirable for several reasons. First, hand-off generates traffic on the wired network even while no new multicasts are generated. Second, given the current trend toward smaller cells **[CP98]** one should privilege strategies that accommodate frequent cell switching efficiently. In particular, hand-off does not appear to be suitable for picocellular networks because of the large amount of information that need to be forwarded **[GS94]**. Third, the traffic related to mobility management tends to be high itself **[MAO92]**, thus one should strive to avoid further traffic for higher-level protocols.

Absence of hand-off is desirable also from other points of view that we briefly mention in the following. More details about these claims can be found in the cited papers. First, the resulting protocol may be made tolerant to crashes of SHs much more easily **[ABS99b,LBA00]**. Essentially, hand-off implies that MSSs store critical state information and that this information has to travel across MSSs upon each cell switching, whereas in our approach MSSs merely *cache* state information and each MSS manages its cache *autonomously*, independently of cell switchings. Second, one can base correctness on more practical mobility assumptions. For example, **[AB96]** requires that MHs do not move while a certain portion of the protocol, related to hand-off, is in progress, and **[PRS97,AV97]** (appear to) require that the speed of users is always slower than the rate at which hand-off completes. In our case correctness does not depend on "when" a MH moves but it depends on *global* assumptions, i.e., of the form “a group member does not move very fast, *all the time*” **[B98]**. With our protocol, movements at inopportune times can only cause an occasional performance penalty and do not affect correctness. Furthermore, such penalty disappears when the MH stops moving “too fast”.

The protocol in **[PRS97]** extends the proposal in **[AB96]** to preserve causal order of deliveries. State information that MSSs maintain on behalf of MHs, and that must be transferred upon hand-

off, is augmented by a vector with one integer per MSS and a vector with one element per MH, where each element is a set of integer pairs. All arguments made about **[AB96]** apply to this protocol.

The protocols in **[AV97,YHH97]** support reliable causally-ordered message delivery in mobile distributed systems. The paper **[AV97]** actually presents a family of three protocols, all based on hand-off and all derived from the protocol for stationary hosts in **[RST91]**, whose crucial differences include the size of the message headers. The first algorithm requires headers of $O(N_M^2)$, where N_M is the number of MHs, the second requires headers of $O(N_G^2)$, where N_G is the number of MSSs, the third requires headers of $O(k^2 * N_G^2)$, where k is a positive integer. A variant to these protocols that requires $O(N_M * N_G)$ is presented in **[YHH97]**. Our protocol has message headers whose size depends only on the number of coordinators, that is typically much smaller than either N_M or N_G. On other hand, the protocols in **[AV97,YHH97]** are not restricted to multicast communication. As an aside, the protocols in **[AV97,YHH97]** are based on hand-off and, specifically, the protocol in **[AV97]** broadcasts a message to *all* MSSs upon *each* cell switching.

The protocol in **[ARR97]** supports *FIFO-ordering* of deliveries within a *static* group of MHs. It is meant to reside on top of a *location directory*, i.e., a service that indicates the MSS currently associated with a given MH. It requires a form of hand-off whose details depend on the guarantees offered by the underlying location directory. The paper evaluates by simulation the hand-off traffic on the wired network in the absence of new multicasts. The number of messages in the wired network per multicast grows with the number of receivers, e.g., around 5 and 7.5 with 2 and 5 receivers, respectively. Our protocol does not require a location directory, need not track the location of individual MHs, generates traffic on the wired network only when new multicasts are indeed generated, and imposes a load on the wired network that depends only on the number of MSSs, not on the number of MHs. With respect to such load, in the absence of message losses, FIFO and Causal ordering require a point-to-point message and a multicast to the

mobile support stations, whereas Total ordering requires an additional point-to-point message.

Reliable multicast protocols for mobile distributed systems generally insist on preventing the arrival of duplicates at MHs (the so-called *exactly-once* property) **[AB96,ARR97]**. Hand-off and *unicast* message transmission (i.e., a *single* transmission for *each* MH) are the main tools for achieving this property. In our protocol *broadcast* transmission is used inside wireless cells and a MH may occasionally receive duplicates of multicasts already delivered, due to the absence of hand-off and to the simple negative acknowledgment scheme used. Such duplicates will be discarded by the protocol layer at MH itself, i.e., without delivering them to the application. The main advantage of this approach is that it allows to fully exploit the broadcast nature of the wireless medium. The drawback is that some data structures need to be maintained at MHs, in order to detect duplicates. However, such data structures only require some few bytes of memory. As we shall see in Section 4, MHs have also a buffer for messages arrived out-of-order, but this is an optimization that is not necessary for correctness. Hence, an actual implementation may trade memory resources against delivery latency according to its own needs. Furthermore, full performance can be obtained even by reserving just a few kilobytes for buffering, as we have observed through simulation.

We did not strive to achieve exactly-once on the ground that it does not really imply substantial battery savings: given the broadcast nature of the wireless medium, the lower protocol layers of *all* MHs in a cell have to handle each message directed to any of them — i.e., a MH has to handle many unnecessary messages anyway. Furthermore, as confirmed by our simulation results, duplicates do not consume wireless bandwidth significantly. Important battery savings could instead be obtained by minimizing *transmissions* operated by MHs. This feature is not exploited in **[AB96]**, where each multicast has to be explicitly and individually acknowledged by the MH. In contrast, we piggyback acknowledgments into other protocol messages that are needed anyway, i.e., for recovering from missing messages.

The protocol in **[AB96]** is such that a MH has to acknowledge every received multicast before the

MSS can send another multicast to it. This strategy consumes scarce wireless bandwidth and increases contention for the wireless link, thus contributing to increase the average end-to-end delivery latency. Our protocol does not need explicit acknowledgments and does not delay the transmission of new multicasts operated by MSSs. From this point of view, the protocol in **[ARR97]** takes an approach similar to ours.

3 System model

MHs and SHs communicate solely through message-passing. Communication in the wired network is reliable and FIFO-ordered. An MSS may broadcast messages to all MHs in its cell and send messages to a specific MH in its cell, whereas a MH may only send messages to the MSS of the cell where it happens to be located. Communication within a cell is FIFO-ordered (the wireless link is basically a point-to-point link) but messages may be lost. MHs may roam in areas not covered by any cell. Host failures are beyond the scope of this paper.

Each MSS maintains a data structure, called local, that identifies the set of MHs in its cell. A MSS becomes aware of which MHs are in its cell by means of a *beaconing* protocol. Periodically, the MSS broadcasts a "beacon" message within its cell. Upon receiving a beacon, a MH announces its presence in the cell by sending back a response "greeting" message that contains its host identifier. We shall omit the details of this protocol for sake of brevity.

We point out what follows: (I) When a MH switches between cells, the related MSSs do not exchange any state information about this computer; they simply update autonomously the respective local. (II) We allow messages exchanged between a MH and an MSS to be lost even while MH continues to belong to local at that MSS; for instance, there could be a physical obstruction in the related cell or MH could leave and re-enter this cell so quickly that its movement is not tracked by the beaconing protocol. (III) A MH could (temporarily) belong to local at multiple MSSs; for instance, because the execution of the beaconing protocol triggered by a cell switching cannot complete simultaneously at both MSSs involved. In short, location information need not reflect "instantaneously" the actual cell compositions and may be

(temporarily) inaccurate. We interpret local only as a "hint" about the actual cell composition and use it only for allocating and deallocating data structures at MSSs. The content of local is never critical for correctness.

4 The protocol

4.1 Interface

A MH becomes a group member by executing the Join() primitive and stops being a group member by executing the Leave() primitive (we consider only one group for ease of presentation). Each group member may send multicasts to the other members, through the following primitives: F-Cast(), that provides FIFO-ordered delivery; C-Cast(), that provides causally-ordered delivery **[L78,PRS97,AV97]**; and T-Cast(), that provides a total order on multicast delivery that is consistent with the causal order of transmissions. More in detail, consider the following properties **[HT93]**:

- (**FIFO Order**) If a group member sends a message m2 after sending a message m1, then any group member that delivers both messages delivers first m1 and then m2.
- (**Causal Order**) If the transmission of a message m2 causally follows the transmission of a message m1, then any group member that delivers both messages delivers first m1 and then m2.
- (**Total Order**) Any two group members that deliver a pair of messages m1 and m2, deliver them in the same order, e.g. either they both deliver m1 before m2, or they both deliver m2 before m1.

FIFO Order is guaranteed for any pair of messages. Causal Order is guaranteed for any pair of messages in which m2 is sent via either C-Cast() or T-Cast(). Total Order applies to any pair of multicasts sent via T-Cast().

The fact that group members indeed deliver messages is guaranteed under reasonable assumptions on their physical movements, i.e., "a group member does not move very fast, all the time" **[B98]**. In particular, a group member M stops delivering messages if: (i) M starts entering and leaving

cells so quickly that its messages never arrive to any MSS or the matching acknowledgements are systematically lost; and (ii) this pattern of movements persists forever. Similarly, a host wishing to be a group member indeed manages to become a group member unless it moves according to (i) and (ii).

Some of the messages delivered by group members consist of *membership changes* that are triggered by Join() and Leave() primitives. Such messages are marked with a special flag and contain the current group membership. Property Total Order apply to membership changes.

4.2 Implementation overview

We say that a host C *receives* a message m when m arrives at the protocol at C. We say that C *delivers* m when the protocol forwards m up to the application. For ease of presentation we shall assume that all group members run on MHs and that there is at most one group member at each host.

A statically defined set of SHs act as *coordinators*. A statically defined coordinator is the *boss*, indicated CB. Each group member is associated with a coordinator. Initially we shall assume that: (I) the group membership is static; (II) it is known to all coordinators; (III) each group member knows the identity of its own coordinator. We shall remove these assumptions in Section 4.4, where we shall discuss group joining and leaving.

In this section we shall provide an overview of the implementation, whereas in the next sections we shall detail the actions performed at each host. The overview is based on Figure 1, that shows simple examples of the message pattern resulting when a group member, say p, issues a multicast. The examples assume 3 cells, 4 coordinators, 7 MHs (s is out of coverage). Letters next to coordinators indicate the associations with MHs (the boss CB is not associated with any MH). The first field of each message is a value of an enumerated type called *tag* and indicated in SMALLCAPS. Each group member has a unique *member identifier* selected by the group member upon joining the group (see Section 4.4).

A group member mid issuing either F-Cast() or C-Cast() (Figure 1-left) sends a NEW message

containing the payload to the local MSS (1). The MSS replies with an acknowledgement (2) and forwards the message to the coordinator associated with mid, say C(mid) (3). C(mid) changes the tag to NORMAL, appends a sequence number and multicasts the resulting message to MSSs (4). MSSs then broadcast NORMAL messages to group members in the respective cells (5).

A group member mid issuing a T-Cast() (Figure 1-right) sends a NEW message as above and C(mid) constructs a NORMAL message as above. However, in this case, C(mid) forwards the NORMAL message to the boss CB (4) rather than multicasting it to MSSs. The boss appends a further sequence number and then multicasts the resulting message to MSSs and C(mid) (5).

Incomplete coverage, unreliable wireless links, unpredictable movements of MHs make possible that NEW messages be lost. For this reason, mid retransmits each NEW message until receiving the matching acknowledgment from a MSS.

Retransmissions and unpredictable movements of MHs make possible the arrival of duplicate and out-of-order NEW messages at coordinators. To cope with this, mid attaches a locally-generated sequence number to each NEW message enabling C(mid) to discard duplicates and to process NEW messages in the order in which they were generated.

Incomplete coverage, unreliable wireless links, unpredictable movements of MHs, make possible the loss of NORMAL messages at group members as well as the arrival of duplicate and out-of-order NORMAL messages. Such events are detected by group members based on the sequence numbers attached by coordinators. Missing messages provoke a retransmission request in the form of a NACK message sent to the local MSS (see below). Duplicates are discarded while out-of-order messages are buffered until they can be delivered without violating the ordering specified by the sender.

A MSS that receives a NACK from group member mid retrieves a copy of the multicasts indicated in the NACK and sends them in sequence to mid. Such copies are extracted from a local cache maintained by the MSS (i.e., when the MSS receives a NORMAL message m from a coordinator, it broadcasts m in the cell and stores m in the cache). If a message is not cached, then MSS fetches

a copy from the coordinator that originated that message.

To make sure that any fetch requests from MSSs can be satisfied, each coordinator C stores a copy of each NORMAL message m sent. C discards m when it knows that m has been delivered by all group members, i.e., that m has become *stable* **[BSS91]**. Each group member indicates which multicasts it has delivered within NACK messages. MSSs extract this information and forward it to coordinators as part of the processing of retransmission requests.

4.3 Actions at each host

In this section we describe the specific actions performed by group members, MSSs and coordinators, respectively, for managing messages. Full details can be found in the companion report **[ABS99c]**.

Actions at Group Members

A group member mid may receive either a request to issue a multicast from the application or a NORMAL message from the local MSS. The related actions are as follows.

- When mid wishes to issue a multicast, it sends a NEW message to the local MSS. This message includes (i) the payload, (ii) the identifier of the coordinator associated with mid, C(mid), (iii) a flag, order, specifying the desired ordering, (iv) a locally-generated sequence number, and (v) a description of the messages delivered by mid so far (i.e., a copy of the delivered array, see below). The group member periodically resends this message until receiving an acknowledgment from a MSS. If mid roams in an uncovered region before receiving the acknowledgment, retransmissions are postponed until entering a cell again.
- Upon receiving a NORMAL message m (see the related flowchart in Figure 2), mid checks whether m is a duplicate. To this end, mid maintains an array of sequence numbers, delivered, with one entry per coordinator and initialized upon joining the group (Section 4.4). The entry associated with coordinator Ck has this interpretation: mid has delivered all NORMAL messages sent by Ck with sequence number $s \leq$ delivered[Ck]. By comparing the sequence number in the received message m to the pertinent entry of delivered, mid

determines whether m is a duplicate, in which case mid discards m. Then mid checks whether m satisfies the *delivery condition* for the ordering specified by the originating group member. This condition depends on (**i**) the messages delivered by the originating group member upon sending m (m includes a description of these messages); (**ii**) the sequence number of m; and (**iii**) the array delivered at mid (see **[ABS99c]** for details). If the condition evaluates to false, then m cannot yet be delivered otherwise the desired ordering would be violated, i.e., mid has not yet received one or more messages that must be delivered before m. In this case, mid buffers m and requests retransmissions of these messages. It does so by sending to the local MSS a NACK message describing the missing messages and the delivered messages. This NACK is resent periodically because it might be lost. On the other hand, if m satisfies its delivery condition then mid delivers m, updates delivered accordingly and process the buffered NORMAL messages whose delivery condition has become true (due to the delivery of m).

Actions at MSSs

A MSS may receive NEW and NACK messages from group members and NORMAL messages from coordinators. The corresponding actions are as follows.

- Upon receiving a NEW message from a group member mid, MSS sends an acknowledgment to mid and forwards the message to the coordinator C(mid) specified in the message itself.
- Upon receiving a NORMAL message, MSS stores a copy of the message in the local cache and broadcasts the message in the cell.
- Upon receiving a NACK message from a group member mid, MSS extracts the description of messages already delivered by mid and forwards this description to coordinators by means of a STABINFO message. Then, it extracts the description of the missing messages and relays these messages to mid, through a sequence of NORMAL messages. These messages are obtained from the local cache or, in case of a miss, from the coordinator that originated the message. If MSS detects through beaconing that mid has left the cell, then MSS stops the

transmission and considers the processing of the retransmission request as completed. If the NACK is received while MSS is processing a previous NACK by the same mid, then MSS determines which NACK is more recent and discards the other, e.g., MSS might abort the sequence of messages being relayed and start a new one.

Finally, when MSS receive no NORMAL messages from coordinators, it periodically retransmits in the cell the sequence number of the last NORMAL message sent (or includes this sequence number within beacons). This allows group members to detect possible NORMAL messages which got lost.

Actions at Coordinators

A coordinator C may receive NEW messages from group members associated with it and STABINFO messages from MSSs (we omit the obvious handling of requests sent by MSSs as a result of a cache miss). If the coordinator is acting as the boss it may also receive NORMAL messages from other coordinators. The corresponding actions are as follows.

- Upon receiving a NEW message m (see the related flowchart in Figure 3), if the sending group member mid is not included in the current membership C discards m[1]. Otherwise, C checks whether m is a duplicate, in which case it discards m (recall that NEW messages carry a sequence number selected by the sender). Then, C checks whether m arrived out-of-order, in which case it buffers m. In case m is not a duplicate and arrived in-order, C changes the tag to NORMAL, appends a locally-generated sequence number and stores a copy of the resulting message m1 into norm-buffer. This is a data structure containing each NORMAL message sent by C that might have not been delivered yet by all group members. Then, if the ordering specified in m is T-Cast(), C sends m1 to CB, otherwise it multicasts m1 to MSSs. Finally, it processes NEW messages sent by mid that are buffered and have become in-order.
- Upon receiving a STABINFO message, the coordinator C extracts the description of the messages sent by C itself and delivered by the specified group member mid (this description

[1] This check would not be necessary in a static group. It is necessary in order to support dynamic membership as shown in the next section.

is a copy of the delivered array at mid, i.e., a sequence number for each coordinator). Based on this description, C updates norm-buffer to record messages delivered by mid and removes from norm-buffer messages delivered by all group members.

- Upon receiving a NORMAL message (this event may occur only at CB), C appends a locally-generated sequence number, stores a copy of the resulting message in norm-buffer and multicasts this message to the MSSs and to the coordinator that sent the message.

4.4 Dynamic membership

Group joining and leaving occur as follows (see also **[ABS99c]**). Initially, there are no group members. Membership changes are propagated by means of totally-ordered NORMAL messages carrying only a description of the membership change.

- A MH wishing to become a group member constructs a JOIN message and sends this message to CB. The message includes the unique host identifier of MH and a join-id, i.e., a bit pattern selected by MH so that it is different from any other join-id that MH selected in the past (e.g., a 32-bit random pattern **[L93]**). The pair (MH, join-id) will be the member identifier mid of the new group member. Then MH starts listening to NORMAL messages sent by CB until receiving the message with the membership change notifying about its inclusion in the group[2]. This message informs MH about the identity of its own coordinator and about the initial values for the delivered array. MH will deliver messages with greater sequence numbers.
- A group member wishing to leave the group sends a NEW message with a special indication to CB. Upon receiving the acknowledgment from a MSS, the group member stops participating in the protocol.

The actions of the boss CB with respect to dynamic membership are as follows.

- CB maintains a member-cache with the identifiers of past group members that have left the group. An entry is purged from the member-cache after a time sufficiently long to ensure

[2] Actually, before sending the JOIN, MH listened to NORMAL messages sent by CB in order to initialize delivered[CB], which is necessary for

that no messages related to that group member are still in transit (assumptions of this kind are practically reasonable and are often necessary in distributed computing **[L78,HR94]**).

- Upon receiving a JOIN message, CB checks whether the enclosed mid is either in the current membership or in member-cache. If so, it discards the message. Otherwise, it instructs all coordinators that mid is joining the group and they have to provide their respective sequence number. Having received all such numbers, CB constructs a NORMAL message m containing these sequence numbers and describing the membership change, stores m in norm-buffer and multicasts m to MSSs.
- Upon receiving a NEW message telling that the sending mid wishes to leave the group, CB checks whether mid is in member-cache. If so, it discards the message. Otherwise, it instructs all coordinators that mid is leaving the group and waits for a response. Having received all such responses, CB constructs a NORMAL message describing the membership change, stores this message in norm-buffer and multicasts it to MSSs.

Finally, a MSS inspects NORMAL messages to detect membership changes and to stop relaying missing messages to MHs that, meanwhile, have left the group.

4.5 Observations

We make the following observations about the protocol.

Retransmissions of NEW messages and acknowledgments are mandatory in our system model. These features imply a cost in terms of power consumption and wireless bandwidth usage but such cost cannot be avoided in our model. Of course, one could build a protocol where communication between MH and MSSs does not need retransmissions and acknowledgments, but in a different system model, e.g., the protocol in **[AB96]** assumes no message loss within cells and absence of movements during hand-off.

A MH can move at *arbitrary* times. For instance, it could send a NEW message m while in a cell, then leave the cell before receiving the related acknowledgement, re-send m in a new cell and so

---

requesting retransmissions of missing messages from CB.

on, until receiving the expected acknowledgement.

Upon a cell switching, no message need be exchanged on the wired network. In particular, a MSS need not interact with any other MSS.

When a group member mid remains unreachable "for a while" and then enters a cell, the MSS will use its local cache to bring mid up-to-date and, in case of "long" disconnections, it will perhaps fetch "old" multicasts from coordinators.

The details required for implementing reliable FIFO-multicast in the wired network are irrelevant to our discussion. For example, one could add a layer above the (best-effort) IP multicast if available **[DC90]**.

Stability information may flow to coordinators in a variety of ways, not only with the simple method described, i.e., within NACK and STABINFO messages. For example, group members could piggyback acknowledgments also in NEW messages. Coordinators could even solicit somehow explicit acknowledgments every now and then, for example when a group member does not miss any message for a long time. The key issue is that acknowledgements and deliveries proceed asynchronously.

When a group member receives a multicast m, it decides whether to deliver, buffer or discard m. Buffering improves the performance of the protocol but it is not necessary for correctness, e.g., a received message m that cannot be buffered for lack of buffering space can be discarded (one could even discard a buffered message to make room for m). Intuitively, since the protocol is able to recover from lost messages, the reason for the loss does not matter. Notice that each MH may decide autonomously how much memory to allocate for buffering.

When the number of MSSs covering the area of interest may be much larger than the number of group members, then sending a NORMAL message to all MSSs is clearly an unnecessary cost. Depending on the operating environment, it might be useful to involve in the protocol only those MSSs whose cells actually contain group members. The simple extension presented in the next section, and independent of all what discussed so far, can be used to this purpose.

4.6 Involving only MSSs of non-empty cells

Coordinators maintain a set containing the identifiers of all MSSs covering the area of interest and send multicasts only to MSSs in this set, that we call subscribed. The actions at MSSs are modified as follows: (I) a MSSs that is not currently "subscribed" asks coordinators to include it in the subscribed set whenever its cell becomes non-empty; (II) a subscribed MSS whose cell has remained empty for a while may "unsubscribe". We do not give all details for sake of brevity and refer the reader to **[B98]**, that considers the case of a single coordinator. Essentially, subscription and unsubscription consist of a remote procedure call from the MSSs to the boss CB. The processing of this call consists in updating the subscribed set at all coordinators, which can be done with one multicast on the wired network, from CB to coordinators.

We observe what follows:

- When a group member enters a cell of a MSS that is not subscribed, it may perceive an increased latency of message delivery. However, this delay is of the order of time necessary for processing the related remote procedure call, that is small compared to the speed of users' movements. Correctness is not affected because, intuitively, the cell of the MSS being subscribed is equivalent to an uncovered area.
- Any reliable multicast protocol that does not *always* send *all* multicasts to *all* MSSs must exhibit a sort of start-up period when a group member enters an empty cell. Furthermore, other protocols offer similar functionality only by means of schemes that are more complex and more costly than that outlined here. For example, the extended form of the protocol in **[AB96]** uses a data structure, called *location view*, that is the set of MSSs whose cells contain group members. This location view is replicated at *all* MSSs in the view. Updates to the location view must be performed at *each* replica and must be serialized. Hand-off must be properly synchronized with location view management.
- Beaconing enables an MSS to know the identifiers of MHs in its cell, but in practice not all MHs will be group members. It follows that a MSS cannot tell whether its cell is "empty" or

not. The optimization in this section requires, for example, that a dedicated field of a "greeting" message (the MH's response to a "beacon" message from the MSS) may contain information of the form "I am a group member ".

5 Simulation Environment

To analyze the performance of the protocol we developed a discrete event simulation model and implemented it in C++ language. The simulation model is fully described in **[ABS99c]**. Group members generate messages according to a Poisson process, i.e., time intervals between consecutive messages are random variables exponentially distributed. A group member remains in a cell for a random time interval. The length of this interval is exponentially distributed and its average is a parameter called *average cell permanency time (T_{cell})*. Then the group member either switches to another cell or enters the uncovered area, based on a parameter called *out-of-coverage probability (P_{out})*. The new cell is selected independently of the current position, i.e., all cells have the same probability. A group member not in coverage remains so for a random time interval. The length of this interval is exponentially distributed and its average is a parameter called *average out-of-coverage permanency time(T_{out})*.

The simulator supports a single multicast group with static composition. We did not implement dynamic membership as it would have increased the execution times of the simulations without providing significant insights - we are mainly interested in estimating performance indices about the multicast protocol rather than about the membership protocol[3]. Furthermore, one can obtain raw estimates of the membership performance indirectly, e.g., the time it takes for a MH to become a group member is roughly the time for delivering a few messages.

To better understand the simulation results, it is important to consider the components of the end-to-end delay (or *latency*), i.e., the time experienced by a message along its way from a sending group member to a receiving group member. These components are shown in Figure 4 in case no

[3] With static membership all events of a simulation run contribute to the overall statistics. On the other hand, with dynamic membership a simulation experiment with *k* group members would require the *k* MHs to join one by one an initially empty group. In order to collect statistically significant results, the simulation should then run for a time sufficiently long to allow neglecting the events that occurred during start-up, when

message loss occurs. When a message is lost the latency includes a further component, the retransmission delay, which is the time necessary to detect and recover the lost message. Processing corresponds to protocol actions, while buffering occurs when the message has to be delayed because it arrived out-of-order (e.g., buffered at group members). A message is queued for transmission when its processing is completed but the pertinent network interface is busy. Transmission delays are the times necessary to actually put bits on the wireless/wired medium (determined by the available bandwidth) while propagation delay is the time it takes to these bits for reaching the intended destination.

Time for processing, transmission and propagation is determined by simulation parameters. Buffering and queuing times cannot be predicted in advance as they depend on a number of factors, e.g., message generation rate, number of sending group members, etc. One of the valuable results of our simulations are estimates of buffering times and queuing times. We anticipate that *queuing delays at MSSs*, and specifically at the wireless interface between MSSs and group members, tend to be predominant as the message rate increases. This result is particularly significant because this delay component is due to the mismatch between the wired bandwidth and the wireless bandwidth, hence MSSs are likely to be the ultimate bottleneck of any multicast protocol for distributed mobile systems.

We shall denote by *minimum latency* the latency if queuing delays, buffering delays and retransmission delays were null or negligible, i.e., if latency was determined only by processing, transmission and propagation.

6 Results

We present results[4] that refer to messages issued with T-Cast(). We also performed experiments with all messages requiring Causal or Fifo order and such results were similar to those reported below (see Section 6.5 and **[ABS99a, AB00]** for details). Unless stated otherwise, we set the main parameters as follows (see **[ABS99c]** for more details and section 6.5 for other scenarios).

---

the number of members progressively grew from 0 to *k*.

We considered an operating environment similar to a small campus or building. Therefore, we considered 40 MSSs and 100 group members, of which, 10 are sending group members. Each of them sends, on the average, 8 messages/sec of 512 bytes each. The wired network has 10 Mbps bandwidth and propagation delay uniformly distributed within the range [0.5 - 2.5] msec. The wireless network has 1 Mbps bandwidth and negligible propagation delays, as wireless cells are supposed to be very small, e.g., ten meters. The value considered for the wireless bandwidth is typical for wireless LANs **[IEE97, AL00].** The probability of message loss in a wireless cell is 0.001, but the results are similar for values in the range [0, 0.02] **[ABS99c]** which is typical for an in-building environment without sources of interference **[XP99, ES96].**

We characterized mobility with T_{cell}=5sec and P_{out}=0, i.e., each group member remains in a cell, on the average, for 5 seconds and then enters another cell ($P_{out}\neq 0$ is considered in section 6.4). We assumed that each group members allocates 512 KB for buffering. We adopted a large buffer because we stressed the protocol in highly critical conditions, i.e., high message loss rates in the wireless cell and long disconnection periods. With more realistic values, a size of 32 KB has proven sufficient to obtain full performance **[ABS99c]**. We used 3 coordinators: C_0, C_1 and CB. We decided to consider a boss not associated with any group member in order to estimate the worst-case average latency for messages requiring Total ordering.

6.1 Scalability

Figure 5-left shows the average latency with varying number of *receivers*. It can be seen that the protocol scales very well. Several features of the proposed protocol contribute to this performance, including: a single wireless transmission for broadcasting within each cell (unlike **[AB96]**); absence of hand-off upon cell switching; cache of past messages at MSSs. Protocols in **[ARR97]** and **[AB96]** definitely do not exhibit this feature, as their latency quickly increases with the number of receivers (see Section 6.5 and **[AB00]** for comparison with **[AB96]**). It can be seen that latency is slightly greater than the minimum latency, i.e., buffering, queuing and

[4] Results have been estimated by using the independent replication method and assuming a confidence level of 90% **[LK82].**

retransmission delays are almost negligible with the workloads considered here.

Figure 5-right shows the average latency with varying number of *senders*. It can be seen that the average latency remains close to the minimum latency while the workload is less than 80% of the wireless capacity and grows significantly for higher workloads.

6.2 Potential Bottlenecks

The simulator has been instrumented so as to identify the various components of the latency. This allowed us to understand the bottlenecks of the protocol at high workloads. At first glance, one might expect that the boss CB is the main bottleneck as it has to process all multicasts. However, we found that this intuition is wrong as the major bottleneck is the wireless interface at MSSs.

Table1reports the overall latency for increasing number of senders up to 224 messages/sec (918 Kbps, i.e., 92% of wireless bandwidth utilization). Each row reports the average delay experienced at MSS, coordinator and boss and the part of this delay that is due to queuing. It can be seen that: (I) the average total delays at the coordinator and at the boss are practically negligible in comparison to that at MSS; (II) the total delay at MSS (and, specifically, the queuing component) tends to become the predominant factor of the overall latency.

Simulation results have also shown that this delay is mostly experienced in the transmission queue. This result is clearly due to the mismatch between the bandwidth of the wired and wireless networks.

N_s	Aggregate Message Rate (mess/sec)	Aggr. Bit Rate (Kbps)	Average Latency	Destination MSS		Coordinator		Boss	
				Average total delay	Average queuing delay	Average Total Delay	Average queuing delay	Average total delay	Average Queuing Delay
15	120	480	17.61	6.27	1.88	0.44	0.00	0.45	0.01
20	160	640	20.49	8.10	3.80	0.44	0.00	0.45	0.01
25	200	800	28.11	12.99	8.60	0.44	0.00	0.46	0.02
26	208	852	31.77	15.17	10.78	0.44	0.00	0.46	0.02
27	216	884	37.81	18.46	14.07	0.44	0.00	0.46	0.02
28	224	918	50.91	23.95	19.56	0.44	0.00	0.46	0.02

Table1: Average delays experienced at different points of the system (times are in msec).

We argue that this phenomenon is not a peculiarity of our protocol, but rather a consequence of the coexistence of different network technologies with significantly different bandwidth. In other words, we argue that any multicast protocol will be able to take to each MSS, from the wired side, more messages than the MSS may actually broadcast on the wireless side. Experiments done in

[AB00] for the protocol in **[AB96]** corroborate our intuition (see Section 6.5). From another point of view, the observation that MSSs are an intrinsic bottleneck is particularly significant, because it allows using a simple, logically centralized coordinator-based protocol.

The results in Table1 correspond to 3 coordinators such that the boss CB is not associated with any group member. We also considered the case with 2 coordinators and CB that coincides with either C_0 or C_1. The results are very similar, except that the overall average latency is smaller because half of messages do not experience the step from the coordinator to the boss - as expected, CB not associated with any group member is a worst-case scenario (see [ABS99c] for details).

6.3 Effects of host mobility

Figure 6-left shows that the average latency grows for increasing values of $1/T_{cell}$, i.e., of host mobility. In fact, when host mobility increases the number of missed messages increases accordingly. This causes a greater retransmission delay since missed messages need to be retransmitted and a greater buffering delay since a larger number of messages needs to be buffered at the group member. However, it can be seen that, for values of T_{cell} greater than 2 sec ($1/T_{cell}<0.5$), the average latency remains practically constant. It grows significantly only when each group member remains in the same cell, on the average, for only one second or less. It clearly appears that the influence of host mobility on the average latency is very limited. This is a desirable feature for a protocol for distributed mobile systems. Observe that if a host occasionally moves very quickly it experiences a performance penalty but the protocol remains correct. When the host's speed comes back to normal values the performance penalty disappears.

Figure 6-right shows the percentage of *retransmitted messages* (due to mobility) for increasing values of host mobility. This percentage is defined as the number of messages retransmitted by MSSs (i.e., solicited by NACK) over the number of messages delivered by group members. We measured also the percentage of duplicate messages, that exhibits a similar behavior and is not reported for the sake of brevity. It can be seen that the percentage of retransmitted messages is

very limited, less than 1%, even for very high and probably unrealistic host mobility.

The above results corroborate our design choice of not requiring hand-off in the wired network upon movements of group members - retransmissions and duplicates are the potential drawbacks of absence of hand-off. Hand-off would *prevent* the arrival of duplicate messages at group members and useless transmissions of messages caused by host mobility, but the above results show that insisting on that is probably not worthwhile.

6.4 Effects of incomplete coverage

To analyze the influence of incomplete coverage we divided group members into two classes: *non-disconnecting* members, associated with P_{out}=0, and *disconnecting* members, associated with P_{out}>0. There are 10 and 90 members in the two classes, respectively. Figure 7-left shows the average latency perceived by each class, for increasing values of T_{out}. It can be seen that non-disconnecting members are influenced by the behavior of disconnecting members. The reason is in the fact that frequent and long disconnections imply frequent FETCH messages at coordinators, which provokes longer queuing delays at coordinators perceived by non-disconnecting members.

The protocol can be tuned for alleviating this effect, depending on the application needs. By increasing the number of coordinators, thus reducing the load on each coordinator, the latency perceived by non-disconnecting group members grows very slowly with T_{out} (Figure 7-left). It can be seen that adding more coordinators is beneficial also to disconnecting group members.

Figure 7-right shows the average *realignment delay* of disconnecting members for increasing values of T_{out}. Such delay is the average length of the time interval from the instant at which a member reconnects to the instant at which all the messages missed during the disconnection have been delivered. It can be seen that this delay is comparable with the length of disconnection. It is not significantly influenced by the number of coordinators, as queuing delays at coordinators give a small contribution to the realignment time especially in the case of long disconnection periods (delays at MSSs and wireless transmission delays predominate).

6.5 Observations

The protocol in **[AB96]** was the first indirect protocol to be proposed for reliable multicasting in mobile wireless systems and its structuring has been highly influential in the design of later protocols (see section 2.2). We were not aware of any performance analysis for this family of protocols, so we performed ours by simulation and compared the performance of **[AB96]** to our proposal **[AB00]**. Although the scenarios analyzed in **[AB00]** and in the present work are different, we summarize the main results below for sake of completeness.

In order to exercise the two protocols in the same conditions, we had to use a scenario different from the one presented here but compatible with the protocol in **[AB96]**, in particular, *FIFO* multicast, *complete* coverage, *reliable* wireless network. Our protocol outperformed the one in **[AB96]** with respect to all indices that we analyzed: latency, scalability, bandwidth usage efficiency (wired and wireless), quickness in managing cell switches of MHs. For example, the latency of our protocol exhibits the same trend as in

Figure 5, whereas for the other protocol latency increases noticeably with the number of receivers. Furthermore, for **[AB96]**, the slope of the latency vs. number of receivers curves increases very quickly with the number of senders.

The key rationale for these results is that our protocol allows a better exploitation of the broadcast nature of the wireless medium. Whereas in our protocol MSSs "blindly" *broadcast* new multicasts in the respective cell, in the protocol in **[AB96]** MSSs uses *individual unicasts* for each MH in order to preserve the exactly-once delivery property.

With respect to wireless and wired bandwidth usage, we performed also analytical predictions in addition to simulations. We interpreted the much better results of our protocol as due, respectively, to exploitation of the broadcast nature of the wireless medium and absence of hand-off.

In general, we found further support for two claims that we made in the previous sections: (I) the percentage of lost messages is very small, even for high mobility; and (II) the main component of

the overall delay and main scalability bottleneck is queuing delay at MSSs. Specifically, we found that even when using the protocol in **[AB96]** the potential bottleneck of the system remains the wireless interface between MSSs and group members, thus confirming that this is not a peculiarity of our protocol.

Further simulation results for our protocol can be found in **[ABS99a]**. In that analysis we considered causally-ordered multicast with wired and wireless bandwidth 100 Mbps and 10 Mbps, respectively. Some other parameters were different from those assumed here (e.g., higher message generation rate and more MSSs), see the cited report for details. We found the same trend observed here and in **[AB00]**. Furthermore, we found that although causality information is maintained on a per-coordinator basis, rather than per-group member, unnecessary buffering delays incurred at group members for preserving Causal order are negligible (such unnecessary delays are called *inhibition* **[AV97]**). Similar conclusions are drawn by Alagar and Venkatesan in their paper **[AV97]**.

7 Conclusions

We have presented and evaluated a reliable multicast protocol for distributed mobile systems that supports dynamic membership and accommodates neatly three increasingly strong delivery ordering guarantees: FIFO, Causal and Total. The system model assumed is quite general and includes incomplete spatial coverage of the wireless network while no limitations is posed on the mobility pattern of group members. Movements of MHs do not imply the exchange of any message in the wired network and, in particular, cell switching does not require any interaction between MSSs. The protocol has been designed by keeping in mind that MHs may have scarce resources. For example, the size of data structures at MHs and of message headers do not depend on the number of MHs. Furthermore, MHs do not acknowledge every received multicast individually.

Simulation results have shown that the protocol scales well and that the main limiting factor is the waiting time spent for the wireless transmission at MSSs. Since this queuing delay is due to the

mismatch between the bandwidth of the wired network and the wireless network, this factor appears to be a necessary consequence of the underlying computing platform rather than a peculiarity of our protocol. From another point of view, the fact that queuing delays at MSSs are the likely bottleneck of any similar protocol implies that our simple coordinator-based architecture is indeed sound and practical. Another important result is that absence of hand-off does not introduce significant costs in terms of duplicate messages or useless retransmissions. It follows that the ability to accommodate efficiently frequent movements of a large number of MHs comes at almost no cost.

Acknowledgements

The authors wish to thank the anonymous referees for their detailed and constructive comments.

References

[AB96] A. Acharya, B. Badrinath, “A framework for delivering multicast messages in networks with mobile hosts”, *ACM/Baltzer Mobile Networks and Applications*, vol.1(2), June 1996, pp. 199-219.

[AB00] G. Anastasi, A. Bartoli, "On the structuring of reliable multicast protocols for mobile wireless computing", Technical Report DII/00-1, January 2000. Submitted for publication (available at *http://www.iet.unipi.it/~anastasi/papers/tr00-1.pdf*).

[ABS99a] G. Anastasi, A. Bartoli, F. Spadoni, “A flexible multicast protocol for distributed mobile systems: design and evaluation”, MOSAICO Project Technical Report PI-DII/1/99, February 1999 (*http://www.iet.unipi.it/~anastasi/papers/tr99-1.pdf*).

[ABS99b] G. Anastasi, A. Bartoli, F. Spadoni "Group Multicast in Distributed Mobile Systems with Unreliable Wireless Network", Proceedings of the *18th IEEE Symposium on Reliable Distributed Systems (SRDS'99),* Lausanne (CH), October 18-21, 1999.

[ABS99c] G. Anastasi, A. Bartoli, F. Spadoni, "A Reliable Multicast Protocol for Distributed Mobile Systems: Design and Evaluation (extended version)”, Technical Report DII/99-2 (*http://www.iet.unipi.it/~anastasi/papers/tr99-2.pdf*), September 1999.

[AL00] G. Anastasi, L. Lenzini, “QoS Provided by the IEEE 802.11 Wireless LAN to Advanced Data Applications: a Simulation Analysis”, *ACM/Baltzer Wireless Networks,* Vol.6, No.2, pp. 99-108, 2000.

[ARR97] V. Aravamudhan, K. Ratnam, S. Rangajaran, "An Efficient Multicast Protocol for PCS Networks", *ACM/Baltzer Mobile Networks and Applications*, vol.2, n.4, pp. 333-344, 1997.

[AV97] S. Alagar, S. Venkatesan, "Causal Ordering in Distributed Mobile Systems", *IEEE Transactions on Computers*, 46 (3), March 1997, pp. 353-361.

[B98] A. Bartoli, "Group-Based Multicast and Dynamic Membership in Wireless Networks with Incomplete Spatial Coverage", *ACM/Baltzer Mobile Networks and Applications*, vol.3(2), June 1998, pp. 175-188.

[BB97] A. V. Bakre, B. R. Badrinath, “Implementation and Performance Evaluation of Indirect TCP”, *IEEE Transactions on Computers*, 46 (3), March 1997, pp. 260-278.

[BPT96] P. Bhagwat, C. Perkins, S. Tripathi, “Network Layer Mobility: An Architecture and Survey”, *IEEE Personal Communications*, June 1996, pp. 54-64.

[BSS91] K. Birman, A. Schiper, P. Stephenson, “Lightweight causal and atomic group multicast”, *ACM Transactions on Computer Systems*, vol.9, n.3, pp.272-314, August 1991.

[CP98] R. Caceres, V. Padmanabhan, "Fast and Scalable Wireless Handoffs in Support of Mobile Internet Audio", *ACM/Baltzer Mobile Networks and Applications*, vol.3, n.4, pp. 351—363, 1998.

[CWBM98] V. Chikarmane, C. Williamson, R. Bunt, W. Mackrell, "Multicast Support for Mobile Hosts Using Mobile IP: Design Issues and Proposed Architecture", *ACM/Baltzer Mobile Networks and Applications*, vol.3, n.4, pp. 365-379, 1998.

[DC90] S. Deering, D. Cheriton, “Multicast routing in datagram internetworks and extended LANs”, *ACM Transactions on Computer Systems*, vol.8, n.2, May 1990, pp.85-110.

[ES96] D. Eckhardt, P. Steenkiste, “Measurement and Analysis of the Error Characteristics of an In-Building Wireless Network”, *Proc. of ACM SIGCOMM’96,* August 1996.

[FZ94] G.H. Forman, J. Zahorjan, “The challenges of mobile computing”, *IEEE Computer*, vol.27, n.4, pp.38-47, April 1994.

[GS94] R. Ghai, S. Singh, “An architecture and communication protocol for picocellular networks”, *IEEE Personal Communications*, Third Quarter 1994, pp.36-46.

[HR94] H. Attiya, R. Rappoport, “The level of handshake required for establishing a connection”, *Distributed Algorithms*, Lecture Notes in Computer Science 857, Springer Verlag 1994, pp.179-193.

[HT93] V. Hadzilacos, S. Toueg, “Fault-tolerant broadcasts and related problems”, in *Distributed Systems (2nd edition)*, Sape Mullender ed., ACM Press 1993.

[KT96] F. Kaashoek, A. Tanenbaum, “An evaluation of the Amoeba group communication system”, *Proc. of the 16-th IEEE International Conference of Distributed Computing Systems*, May 1996, pp.436-447.

[IEE97] IEEE standard for Wireless LAN- Medium Access Control and PhysicalLayer Specification, P802.11, November 1997.

[L78] L. Lamport, "Time, Clocks, and the Ordering of Events in a Distributed System", *Communications of the ACM*, vol.21 (7) , July 1978, pp. 558--565.

[L93] B. Lampson, “Reliable messages and connection establishment”, in *Distributed Systems (2nd edition)*, Sape Mullender ed., ACM Press 1993.

[LBA00] F. Luccio, A. Bartoli, G. Anastasi, “Fault-Tolerant Support for Totally Ordered Multicast in Mobile Wireless Systems”, Technical Report DII/00-2, April 2000. Submitted for publication (*http://www.iet.unipi.it/~anastasi/papers/aal00.pdf*).

[LK82] A. M. Law, W. D. Kelton, “Simulation Modeling and Analysis”, McGraw-Hill Book Company, 1982.

[MAO92] K.S. Meier-Hellstern, E. Alonso, D.R. O'Neil, "The Use of GSM to Support High Density Personal Communication", *Record of the IEEE International Conference on Telecommunications*, June 1992.

[MDC93] B. Marsh, F. Douglis, R. Caceres, “Systems Issues in Mobile Computing”, Technical Report MITL-TR-50-93, Matsushita Information Technology Laboratory, 1993.

[P96] C. Perkins, "IP Mobility Support", RFC 2002, Mobile IP Working Group, October 1996.

[PGM00] T. Speakman et al., “PGM Reliable Transport Protocol Specification”, *Internet*

Draft, available at *http://www.ietf.org/internet-drafts/draft-speakman-pgm-spec-04.txt.*

[PRS97] R. Prakash, M. Raynal, M. Singhal, “An Adaptive Causal Ordering Algorithm Suited to Mobile Computing Environments”, *Journal of Parallel and Distributed Computing*, March 1997, pp. 190-204.

[PSLB97] S. Paul, K. K. Sabnani, J. C.-H. Lin, S. Bhattacharyya, “Reliable Multicast Transport Protocol (RMTP)”, *IEEE Journal on Selected Areas in Communications*, Special Issue on Network Support for Multipoint Communication, Vol. 15, N.3, April 1997.

[RST91] M. Raynal, A. Schiper, S. Toueg, "The causal ordering abstraction and a simple way to implement it", *Information Processing Letters* vol. 39, n.6, September, 1991, pp. 343--350.

[RV98] L. Rizzo, L. Vicisano, “RMDP: an FEC-based Reliable Multicast protocol for wireless environments”, *ACM Mobile Computing and Communications Review*, Vol. 2, N.2, April 1998.

[WT00] B. Whetten, G. Taskale, “An Overview of Reliable Multicast Transport Protocol II”, *IEEE Network*, Vol. 14, N.1, January/February 2000.

[XP97] G. Xylomenos and G. Polyzos, "IP Multicast for Mobile Hosts", *IEEE Communications Review*, January 1997, pp. 54—58.

[XP99] G. Xylomenos, G. C. Polyzos, “TCP and UDP Performance over a Wireless LAN”, *Proc. of IEEE INFOCOM’99, 1999.*

[YHH97] L. Yen, T. Huang, S. Hwang, "A Protocol for Causally Ordered Message Delivery in Mobile Computing Systems", *ACM/Baltzer Mobile Networks and Applications*, vol. 2, n. 4, pp. 365-372, 1997.

Figures

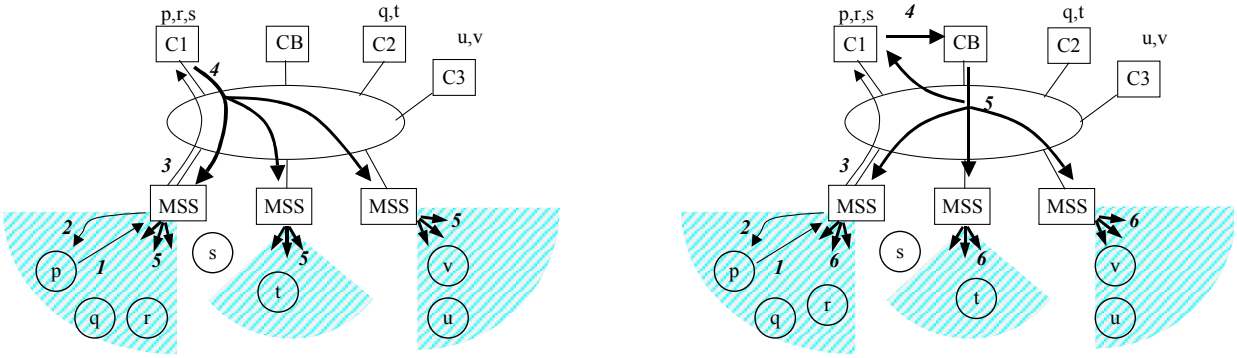

Figure 1. Message patterns for F-Cast(), C-Cast() (left) and for T-Cast() (right). Thin lines indicate NEW messages and acknowledgments, thick lines indicate NORMAL messages.

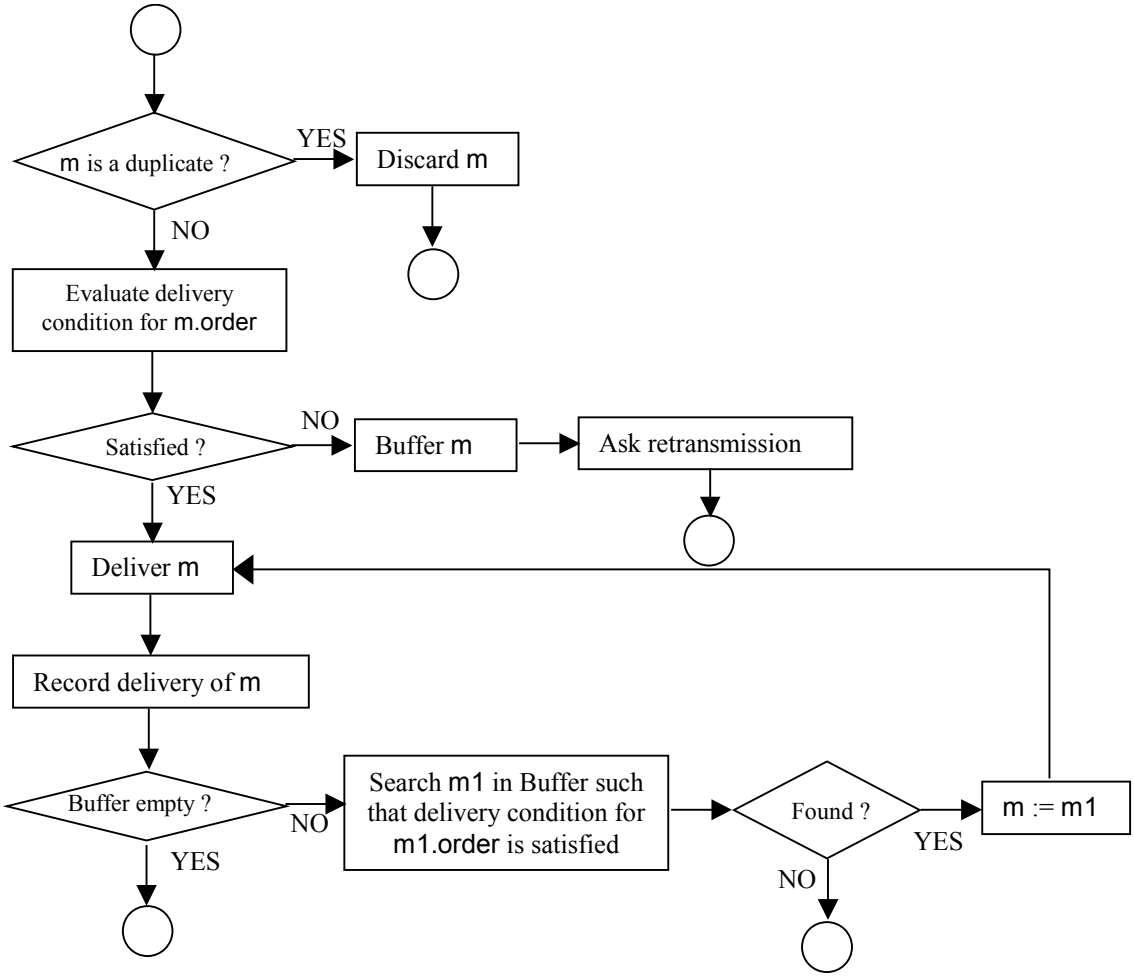

Figure 2. Group member's behavior while processing a NORMAL message.

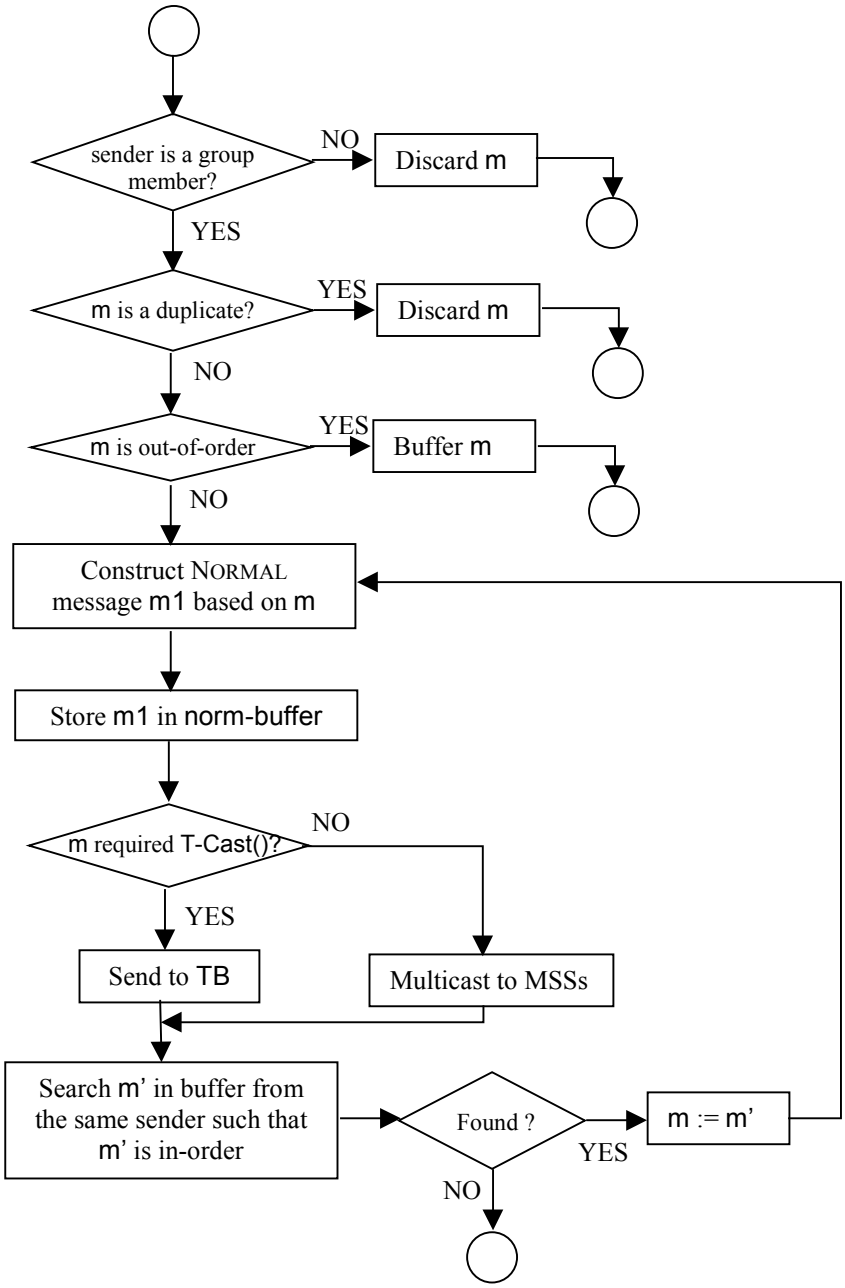

Figure 3. Coordinator's behavior while processing a NEW message.

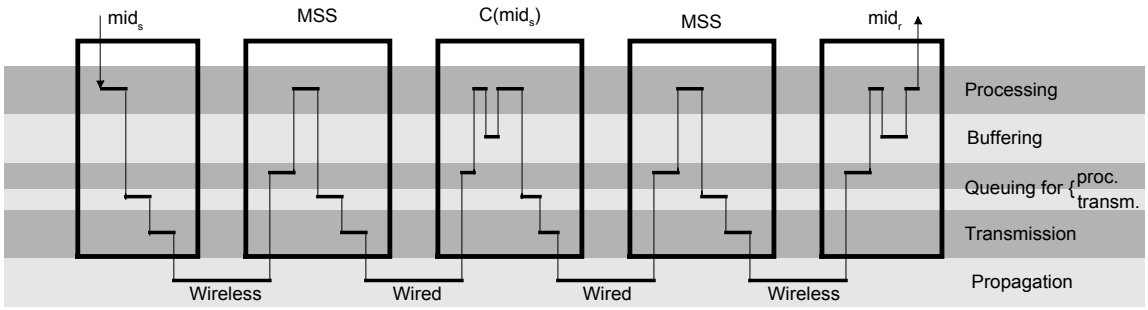

Figure 4. Delays experienced by a message in case no message loss occurs (relative lengths of time intervals do not reflect their actual values).

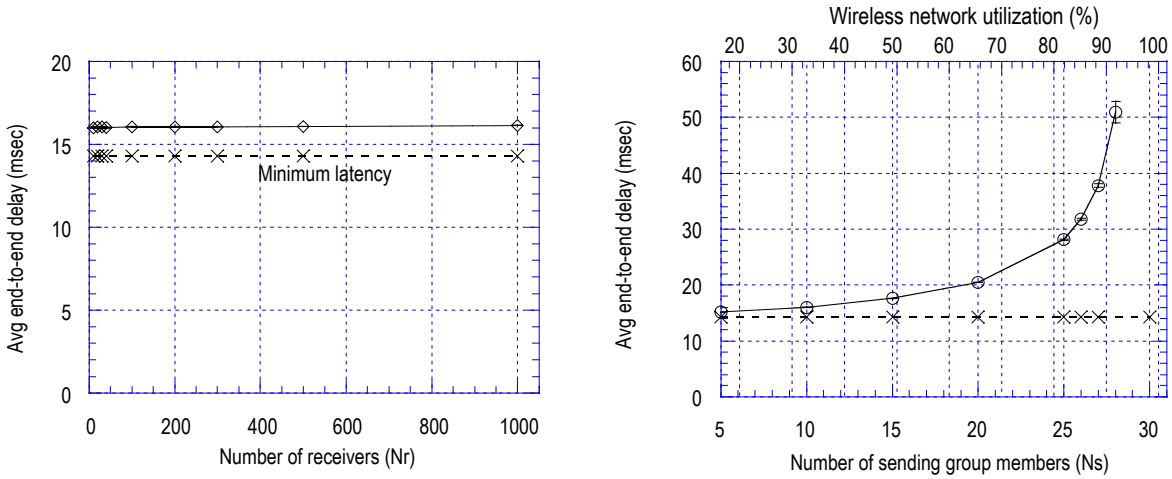

Figure 5. Latency with varying receivers (left) and with varying senders (right).

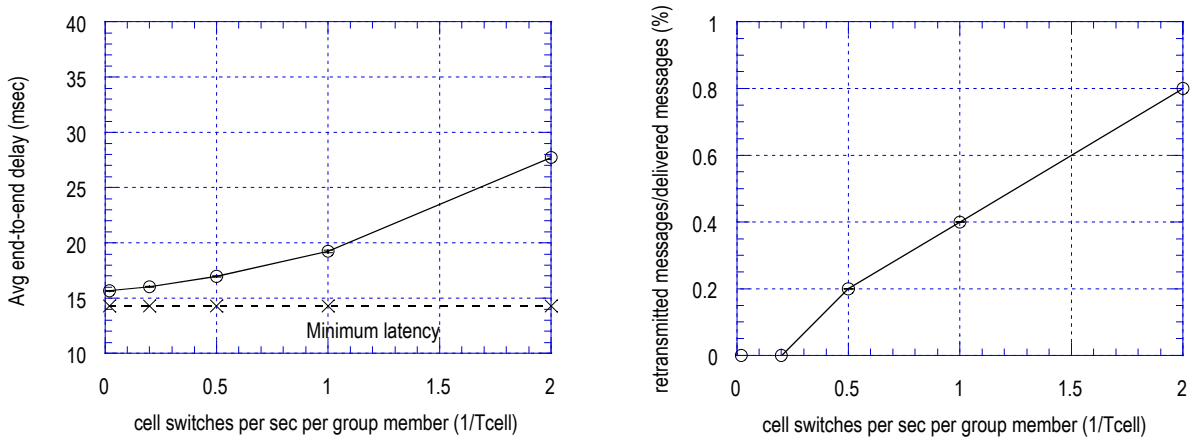

Figure 6. Average latency (left) and percentage of retransmitted messages (right) vs. host mobility.

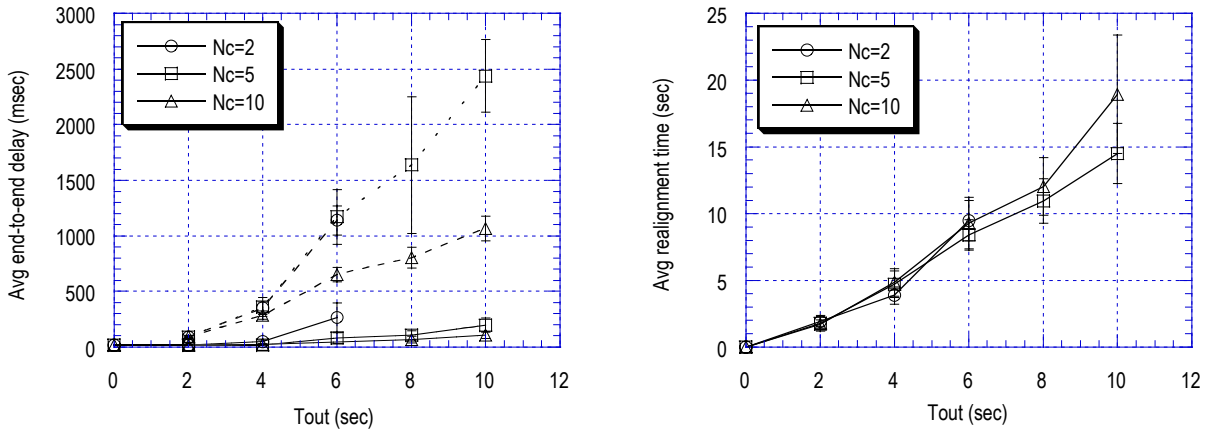

Figure 7. Average latency (left) and average re-alignment delay (right) vs. out-of-coverage permanency time (Tout) for disconnecting (dashed lines) and non-disconnecting (continuous lines) members. N_c does not include the boss.