

A flexible 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 Multicast communication is a powerful paradigm for structuring distributed programs in which multiple processes must closely cooperate together. In this paper we propose a multicast protocol for a distributed system that includes mobile hosts. Mobile hosts communicate with a wired infrastructure by means of wireless technology. Our solution is flexible and provides several novel features. The sender of each multicast may select among three increasingly strong delivery ordering guarantees: FIFO, Causal, Total. By deciding on how many “coordinators” should be reserved on the wired network for supporting the protocol, an implementation may select the proper trade-off between run-time costs, i.e., message size and memory overhead against latency. Movements do not trigger the transmission of any message in the wired network and no notion of hand-off is used. The set of senders and receivers (group) may be dynamic. The wireless network provides only incomplete spatial coverage and physical obstructions could cause message losses even within cells. A user that leaves a cell may enter any other cell, perhaps after a potentially long disconnection. The simulation results show that the protocol scale well with the group size and the overhead introduced by not using the hand-off procedure is very limited.

1 Introduction

Multicast communication is a powerful paradigm for structuring distributed programs in which multiple processes must closely cooperate together [B93,KT96]. Supporting such paradigm in the context of mobile computing poses a number of problems not present in “stationary” distributed computing, due primarily to resource constraints on mobile devices and limited bandwidth of wireless links [FZ94]. Furthermore, mobility could cause loss of multicasts even in the absence of failures [AB96]. In this paper we propose a multicast protocol for a distributed system in which mobile hosts communicate with a wired infrastructure by means of wireless technology. Our solution is flexible and provides several novel features.

The set of senders and receivers (*group*) may be *dynamic*. The sender of each multicast may select among three increasingly strong delivery ordering guarantees: *FIFO* (i.e., single-source ordering); *Causal* (i.e., deliveries occur in an order consistent with the “happened-before” ordering of transmissions [L78,PRS96,AV97]); *Total* (i.e., all group members deliver multicasts in the same order and this order is consistent with causal order). The protocol makes use of a statically defined set of *coordinators* in the wired network. The size of message headers and data structures at mobile hosts depend only on the number of coordinators rather than on the number of group members, which is useful for saving scarce resources on mobile hosts [AV97]. On the other hand, the latency of delivery may increase as the number of coordinators decreases (i.e., each coordinator has to process more multicasts). An implementation may thus allocate on the wired network the set of coordinators that provides a trade-off suitable to its own needs.

Mobile hosts communicate with the wired network through a number of spatially limited cells that define wireless links. We assume that cells provide only *incomplete* spatial coverage and that localized *physical obstructions* could cause message losses even within a cell [B98]. Mobile hosts do not necessarily move among physically adjacent cells. A mobile host that leaves a cell may enter any other cell, or even re-enter the same cell. Furthermore, before re-entering, the mobile host could remain unreachable for a “long” amount of time. The resulting scenario is quite general because it accommodates contemporary wireless LAN’s, disconnected modes of operation, long-range movements, picocellular wireless networks in which the cell size is of the order of a few meters, such as a room in a building [GS94].

Movements of mobile hosts do not trigger any message exchange in the wired network. In contrast, the protocol in [AV97] enforces causal order by broadcasting a message to *all* mobile support stations upon each cell switching. Other protocols are based on the notion of *hand-off* [AB96,AV97,PRS96], in which each cell switching of a mobile host triggers the forwarding of pending messages and administrative information to the new “mobile support station” in the wired network.

Finally, the correctness of the protocol depends only on *global* assumptions about movements of users, e.g. of the form “a group member does not move very fast all the time” [B98]. Other protocols assume that users do not move while certain actions are in progress [AB96] or (appear to assume) that the speed of users is always slower than the rate at which hand-off completes [PRS96,AV97].

2 Overview

2.1 System model

We consider *mobile hosts* and *stationary hosts* that communicate through message-passing. Mobile hosts (MHs) may move and communicate through wireless links. Stationary hosts (SHs) are connected to a wired network that provides reliable and FIFO-ordered communication. Some SHs, called mobile support stations or *gateways*, may communicate also through wireless links. Each gateway defines a spatially limited *cell* covered by a wireless link. A gateway 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 gateway of the cell where it happens to be located. We do not assume any network support for routing messages to a specific MH. To take into account physical obstructions, communication within a cell is FIFO-ordered but messages may be lost. Furthermore, MHs may roam in areas that are not covered by any cell (Figure 1 left). Host crashes are beyond the scope of this paper. SHs may communicate among themselves via FIFO-multicast. For brevity, we shall not consider the implementation of this primitive as it does not provide any additional insight.

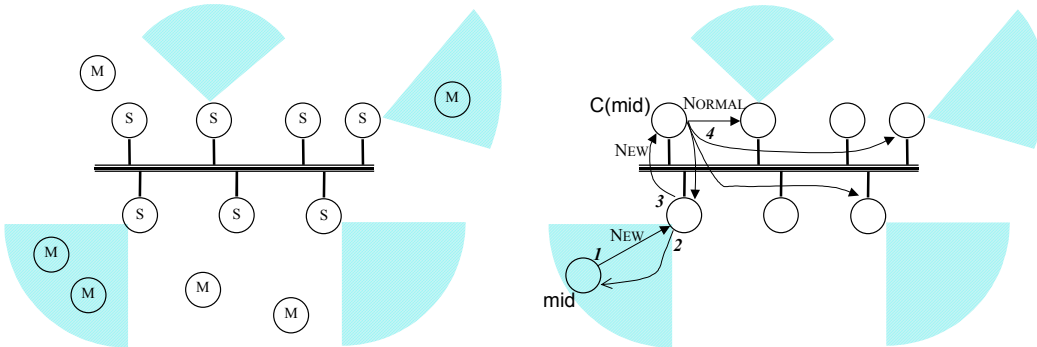


Figure 1 System with 7 SHs, 4 gateways, 6 MHs (left). Example of message-exchange pattern for F-Cast () and C-Cast () (right).

Each gateway becomes aware of which MHs are in its cell by means of a *beaconing* protocol. The minimum requirements that we place on such a protocol are detailed in [B98]. We only point out that gateways need not interact among themselves upon movements of group members. Gateways use the beaconing protocol only for allocating and deallocating *local* data structures. Location information at each gateway need not reflect instantaneously the actual cell composition and may be (temporarily) inaccurate. A MH could even leave and re-enter a given cell so quickly that this movement is not tracked by the beaconing protocol: even if this movement caused some message loss, the effect would be the same as if the MH moved behind a physical obstruction within the cell. In short, location information about mobile hosts need not be fully consistent across the network: different hosts could have different ideas about the location of a mobile host without affecting correctness. This feature contributes substantially to the simplicity of the protocol.

3 The protocol

3.1 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,PRS96,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, AR95]:

- (**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 m_2 is sent via either C-Cast () or T-Cast (). Total Order applies to any pair of multicasts sent via T-Cast (). Moreover, each multicast is delivered exactly once, e.g. duplicates are discarded.

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 gateway 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. Properties Total Order and Causal Order apply to membership changes.

3.2 Implementation overview

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. 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. The first field of each message is a value of an enumerated type called *tag* and indicated in SMALLCAPS. We shall refer to fields of message with the dot notation, e.g. $m.f$ for field f of message m .

A statically defined set of SHs, called *coordinators*, constitutes the *coordinator team*. A statically defined coordinator is the *team boss*, indicated CB . Each group member is assigned a coordinator $C \neq CB$ when it joins the group and this association persists until leaving. As clarified below, each coordinator processes the multicasts issued by group members with which it is associated. T-Cast () are processed also by the team boss CB . The proposed protocol is a generalization of a previous protocol that supports only T-Cast () and makes use of a single coordinator [B98]. 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. In Section 3.6 we shall remove these assumptions.

Each group member has a unique *member identifier*, denoted mid , that it selects upon joining the group (Section 3.6). A MH encloses its member identifier and a coordinator identifier in each message m that it sends. The gateway forwards m to that coordinator (unless m is tagged NACK, see Section 3.5). Furthermore:

- When a gateway G receives a message m , it responds to the sending MH with an acknowledgment $ack(m)$.
- Whenever a MH sends a message m to a gateway, MH periodically re-sends a copy of m (after expiring a specified time-out) until receiving the “matching” acknowledgment $ack(m)$ ¹; furthermore, MH does not send any new message m' until receiving $ack(m)$.
- A MH discards duplicate acknowledgments.

The above actions make communication between MHs and gateways reliable but allow the delivery of out-of-order and duplicate messages at coordinators. To clarify, suppose that MH needs to send reliably a message m to a coordinator S while roaming. To this end, MH keeps on re-transmitting copies of m until receiving the matching acknowledgement $ack(m)$. S may clearly receive duplicates of m . Furthermore, since these duplicates could pass through different gateways and along different paths in the wired network, S could even receive them after receiving other messages that follow m , i.e., that MH sent after receiving $ack(m)$. In other words, any SH participating in the protocol must be prepared to cope with duplicates of *any* message already received in the past. Notice that we are assuming that MH can move at *arbitrary* times. For instance, MH could send m while in a cell, leave the cell before receiving the matching acknowledgment from the gateway, re-send m in a new cell and so on.

A group member mid issuing either F-Cast () or C-Cast () sends a NEW message containing the payload to its own coordinator $C(mid)$. mid maintains a **Sent** sequence number that increases upon sending each NEW message. This sequence number is enclosed in the NEW message, which enables $C(mid)$ to process NEW messages in the same order in which they were sent and to discard duplicates. $C(mid)$ transforms the NEW message into a NORMAL message, assigns a sequence number $cseq$ and multicasts the resulting message to gateways² (Figure 1 right). Gateways then broadcast NORMAL messages to group members in their cells. If

¹ If MH happens to be located outside of all cells, it postpones retransmissions until entering a cell again.

² We are assuming that a NORMAL message is broadcast to all gateways. However, one may use the simple technique presented in

mid issues a T-Cast (), then $C(mid)$ does not multicast the NORMAL message but forwards it to the team boss CB , that appends a further sequence number $cseqB$ and multicasts the resulting message to gateways.

A group member discards a received NORMAL message m if it satisfies the proper *discard condition* (see next section). There is a different condition for each primitive, selected by means of a dedicated field in m . When the discard condition does not hold the proper *delivery condition* is checked. If the delivery condition is satisfied the message is delivered, otherwise it is buffered. Upon delivering a message, the group member checks to see whether any buffered message now satisfies its delivery condition and can thus be delivered.

The structure of a NEW message is $\langle NEW, mid, Coord, Order, Sent, Deliver, p \rangle$, where: mid identifies the sending group member; $Coord$ identifies its coordinator $C(mid)$; $Order$ specifies the multicast primitive invoked; $Sent$ is the sequence number selected by mid ; $Deliver$ is an array with one entry per coordinator, where $Deliver[k]$ contains the largest sequence number among the messages delivered by mid and sent by Ck at the moment mid sends this message; p is the payload. As an optimization, when $Order$ specifies F-Cast () the array $Deliver$ may be omitted.

The structure of a NORMAL message is identical to that of the matching NEW message except for the tag and for the following additional fields inserted by $C(mid)$: $cseq$, a sequence number increased upon sending each NORMAL message; $cseq-mid$, the value of $cseq$ assigned upon processing the last NEW message sent by mid (*originated* by mid for short). When the NORMAL message is processed by CB , which happens when $Order$ indicates T-Cast (), then CB adds a further field $cseqB$, that is a sequence number that CB increases upon sending each NORMAL message. In the following, the (unqualified) sequence number of a NORMAL message m identifies field $m.cseqB$ if m is a T-Cast () and field $m.cseq$ otherwise.

3.3 Handling of received messages at group members

A group member that receives a NORMAL message m selects the delivery condition based on $m.Order$. Consider predicates $Delivered(C, s)$ and $DeliveredAll(C, s)$, where C is a coordinator and s is the sequence number of a NORMAL message sent by C . The former predicate is true iff the group member has delivered the message sent by C with sequence number s , while the latter is true iff the group member has delivered *all* messages sent by C with sequence number smaller than, or equal to, s . The delivery conditions are:

- F-Cast () : $Delivered(m.Coord, m.cseq-mid)$
- C-Cast () : $Delivered(m.Coord, m.cseq-mid)$ and $(\forall Ci, DeliveredAll(Ci, m.Deliver[Ci]))$
- T-Cast () : $Delivered(m.Coord, m.cseq-mid)$ and $Delivered(CB, m.cseqB - 1)$ and $(\forall Ci, DeliveredAll(Ci, m.Deliver[Ci]))$

The data structures maintained by each group member mid for evaluating these conditions are:

- *mystable*: an array with one entry per coordinator. $mystable[Ck]$ is a sequence number with this meaning: mid has delivered all messages sent by Ck and having sequence number $s \leq mystable[Ck]$.
- *myahead*: an array with one entry per coordinator. $myahead[Ck]$ is a *set* of sequence numbers with this meaning: mid has delivered all messages sent by Ck with sequence numbers in $myahead[Ck]$; furthermore, all elements in $myahead[Ck]$ are greater than $mystable[Ck]$.

For brevity, we omit the obvious translation of $Delivered()$ and $DeliveredAll()$ in terms of *mystable* and *myahead* as well as their update upon message delivery. A received message m that does not satisfy the delivery condition is discarded as a duplicate when:

- F-Cast (), C-Cast () : $m.cseq \leq mystable[m.Coord]$ or $m.cseq \in myahead[m.Coord]$
- T-Cast () : $m.cseqB \leq mystable[m.CB]$

If m satisfies neither the discard condition nor the delivery condition, then it should be buffered. To this end, a group member maintains an array *mybuf* where to store messages that cannot yet be delivered.

Figure 2 shows three simple examples about the delivery condition at $mid1$ upon receiving $m2$, that is issued with C-Cast (). Messages going up are tagged NEW, those going down are tagged NORMAL and some NORMAL messages are omitted for clarity. Notice that the senders of $m3$ and $m2$ have different coordinators. Intermediate message-exchange at gateways are omitted for simplicity: Mobility and incomplete spatial

[B98] for restricting the broadcast only to those gateways whose cells indeed contain group members. The cost of the technique is, essentially, a remote procedure call on the wired network whenever a cell switches between empty and not empty. We do not give the related details for reasons of space.

coverage manifest themselves as loss of messages and of FIFO order between messages from a coordinator to a MH. The numbers next to each message indicate the value of the Deliver field.

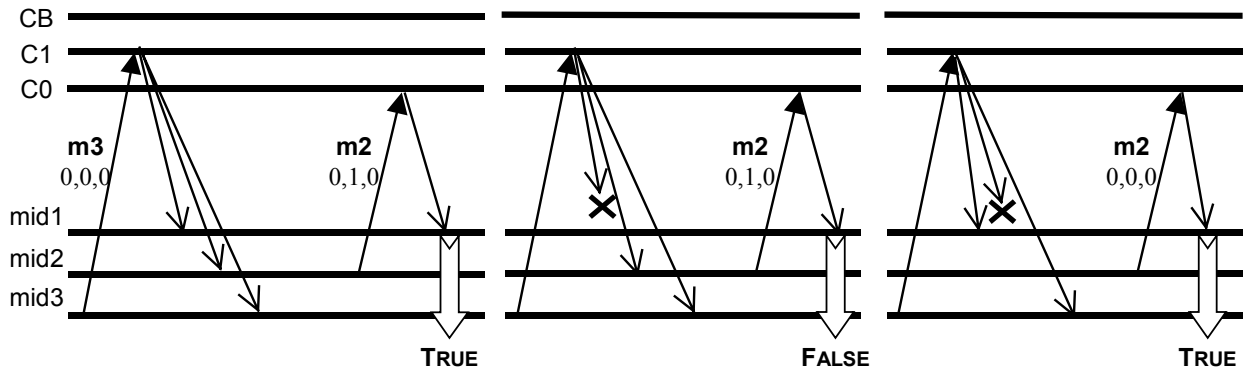


Figure 2 Delivery conditions at mid1 upon receiving C-Cast (m2) .

3.4 Discussion of delivery conditions

Condition $\text{Delivered}(m.\text{Coord}, m.\text{cseq}-\text{mid})$ obviously enforces FIFO Order among messages originated by $m.\text{mid}$. As an example, consider Figure 3 (left) and suppose $m1$, $m2A$ and $m2B$ are all sent via $F\text{-Cast}()$. mid3 buffers $m2B$ because it has not yet received $m2A$ ($m2B.\text{cseq}-\text{mid} = 2$); upon receiving $m2A$, mid3 delivers $m2A$ and $m2B$ even though it has not yet received $m1$.

As for the delivery condition of $C\text{-Cast}()$, consider mid that issues $C\text{-Cast}(m)$. Suppose that mid does not issue any other multicast until delivering m . It is straightforward to see that the subcondition $\forall C_i, \text{DeliveredAll}(C_i, m.\text{Deliver}[C_i])$ enforces a delivery order consistent with Causal Order: m cannot be delivered if any message delivered by mid at the time it originated m has not been delivered already. Suppose now that the mid can issue further multicasts even before delivering m . In this case, one must be sure that any message $m1$ sent by mid in between the sending and delivering of m be delivered after m . This is guaranteed by the sub-condition $\text{Delivered}(m.\text{Coord}, m.\text{cseq}-\text{mid})$, because all multicasts issued by mid are sequentialized by the same coordinator $C(\text{mid})$.

The resulting delivery order is clearly stronger than Causal Order: it could order pairs of causally unrelated messages sent by group members that happen to have the same coordinator. The “unnecessary” ordering manifests itself as increased delivery latency: It could happen that a MH buffers a message $m1$ and waits for another message $m2$ that, actually, need not be delivered before $m1$. As an example, suppose $m2$ in Figure 3 (middle) be sent via $C\text{-Cast}()$. Values of cseq attached by $C0$ to $m1$, $m3$, $m2$ are 1, 2, 3, respectively. When mid3 receives $m2$, it does not realize that it could deliver $m2$ immediately: the delivery condition does not hold because mid3 has not yet delivered *all* messages of $C0$ with sequence number $\text{cseq} \leq 2$; however, message $m1$ with $\text{cseq} = 1$ does not causally precede $m2$.

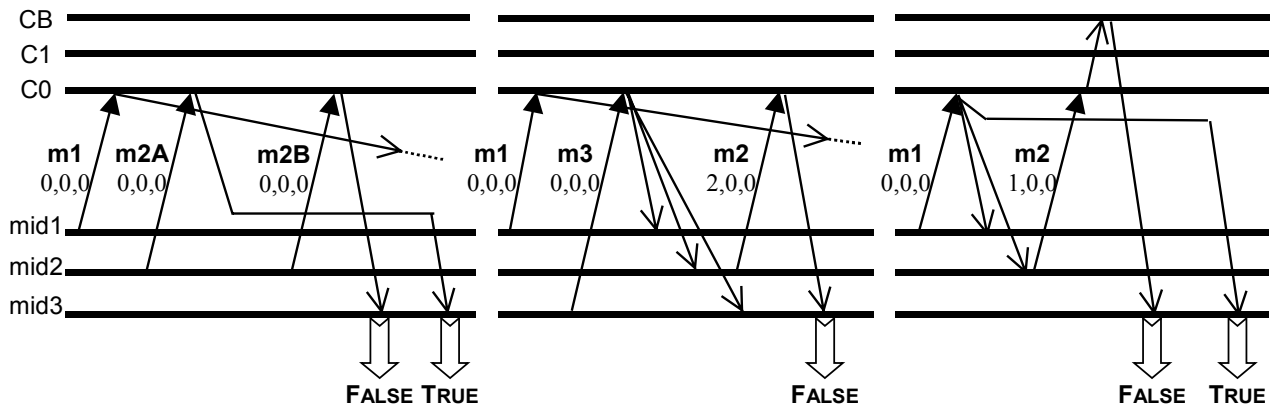


Figure 3 Examples for clarifying delivery conditions (see the text).

The ultimate reason for this feature (called *inhibition* in [AV97]) is because causality information is maintained on a per-coordinator basis, rather than per-group member. The advantage is that the size of data structures at MHs and message size overhead depend on the number of coordinators, not on the number of group members. It should be pointed out, however, that an unnecessary buffering of a message $m2$ sent by C may occur only when: (1) $m2$ is “overtaken” by another message $m1$ sent by the *same* coordinator C ; and

(II) m_1 and m_2 were originated by *different* group members (Figure 3 middle). Since the underlying network is FIFO, this is going to happen only if the receiving group member misses m_2 and receives m_1 , for instance, because of its movement across cells. Furthermore, as shown by results in [AV97], occasional unnecessary buffering does not necessarily result in overall worse performance.

The delivery order is intermediate between two extremes: with only one coordinator C-Cast () would provide Total Order, whereas with one coordinator per group member C-Cast () would provide a “pure” Causal Order, that is, it would not force any delivery order among causally unrelated pairs of messages. It follows that tuning the number of coordinators allows balancing between latency and memory/message size overhead.

Finally, as for T-Cast (), the sub-condition `Delivered(CB, m.cseqB - 1)` enforces in-order delivery of messages sent by CB. This sub-condition alone would enforce a Total Order that might not be consistent with Causal Order. As an example, see Figure 3 (right). Suppose that m_1 be sent with F-Cast () and m_2 with T-Cast (). When `mid3` receives m_2 , Total Order alone would allow delivering m_2 immediately, but in that case Causal Order would not be satisfied (i.e., the delivery condition must be false, as indicated). To achieve Causal Order it suffices to couple the sub-condition being discussed with the condition for C-Cast (). Intuitively, the intermediate processing step at `C(mid)` is necessary to enclose `cseq-mid` in the message, which allows preserving causal order without maintaining per-group member counters.

3.5 Missing messages

When a group member buffers a received message, it requests retransmission of the missing messages immediately. The rationale is the following: since the wired network and the wireless network preserve FIFO order, the missing messages have been probably, although not certainly, lost. Modifying the protocol so that retransmission is requested only “after a while”, however, is straightforward.

Retransmission is requested by sending to the gateway a NACK message m : `<NACK, mid, pair1, ..., pairN>`, where there is a `pair` field for each coordinator and each such field contains two sequence numbers s_{1_k} and s_{2_k} . The meaning of the pair associated with C_k is this: $m.mid$ has delivered all messages sent by C_k with sequence number up to s_{1_k} (included) but it has missed those with sequence number $s \in [s_{1_k} + 1, s_{2_k} - 1]$; in case $s_{2_k} = 0$, $m.mid$ has missed all messages with sequence number $s \geq s_{1_k}$. Upon receiving m , the gateway will relay to $m.mid$ a copy of the missing multicasts, through a sequence of TRANSFER messages. Such messages have the same structure as NORMAL ones except for the tag, that is different only for clarity. Group members handle TRANSFER messages in the same way as NORMAL messages. The gateway obtains a copy of the missing multicasts as clarified below.

The values for the `pair` fields are selected as follows: $s_{1_k} = \text{mystable}[C_k]$. Let lowahead_k denote the smallest sequence number among those in `myahead[Ck]`, or 0 if such entry is empty; let lowbuf_k denote the smallest sequence number among messages in `mybuf` sent by C_k , or 0 if there is no such message:

```

s2k = min(lowbufk, lowaheadk);
if (s2k == 0) then s2k = lowbufk;
if (s2k == 0) then s2k = lowaheadk;

```

Clearly, `mid` shall not send a NACK upon every insertion in `mybuf`, because it could have just sent one. On the other hand, the fact that `mid` has sent a NACK does not imply that it will receive all required multicasts, because some TRANSFER messages could be lost. For this reason: (I) upon sending a NACK, `mid` sets a timer; (II) while the timer is set, `mid` does not send any further NACK; (III) when the timer expires, `mid` sends a further NACK only if it is still aware that it missed a multicast (i.e., there are still buffered messages).

Consider a gateway G that is sending a sequence of TRANSFER message originated by C_k to `mid`. Let the related sequence numbers be `[slowk+1, shighk-1]`. Suppose G receives from `mid` a further NACK message m and let s_{1_k} , s_{2_k} denote the values in the `pair` field about C_k : (I) if $s_{1_k} \leq \text{slow}_k$, then G discards m ; otherwise, (II) G aborts the sequence and starts a new one according to s_{1_k} , s_{2_k} . Furthermore, if G detects through beaconing that `mid` has left its cell, G aborts the sequence.

Notice that `mybuf` is not necessary for *correctness*, but it is merely an *optimization*. Upon receiving a message m that should be buffered, one could even build the NACK and then discard m . Enclosing this idea in the protocol is fairly simple [B98]. Each MH may thus trade size of `mybuf` against likelihood of unnecessary NACK messages. This feature may be useful for MHs with scarce memory resources.

Gateways maintain locally a cache of previous NORMAL messages. Each coordinator stores a copy of each NORMAL message m previously sent until it knows that m is *stable*, which means that m has been delivered by all group members [BSS91]. Whenever a gateway experiences a cache miss while processing a NACK, it

will contact the proper coordinator by a FETCH message. For instance, if a group member *mid* remains unreachable “for a while” and then enters a cell, the related gateway 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.

Stability information may flow to coordinators in a variety of ways. The simpler method consists in the following: when a gateway receives a NACK message *m*, it extracts from each *m.pair_k* field the sequence number *s1_k* and sends to each coordinator *C_k* a message <STABINFO, *m.mid*, *m.s1_k*>. Upon receiving this message, *C_k* learns that *m.mid* has certainly delivered all messages with sequence number *s* ≤ *m.s1_k*. The fact that stability information is piggybacked into other protocol messages that group members have to send anyway is beneficial for reducing power consumption on the MHs and for decreasing the aggregate load on the wireless links. Other protocols instead acknowledge every multicast individually [AB96,AV97].

3.6 Dynamic membership

A MH wishing to become a group member sends to CB a message <JOIN, *M*, *join-id*, CB>, where *join-id* is an identifier selected by *M* so that it is different from any other *join-id* that *M* selected in the past (i.e., a 32-bit random pattern [L93]). The pair (*M*, *join-id*) constitutes the member identifier of the group member, e.g. *mid*. Upon processing a JOIN message, CB admits *mid* into the group and associates it with a coordinator in such a way that coordinators manage (approximately) the same number of group members. The data structures maintained by CB include: (I) the *membership table* *memb-t*, that contains all current member identifiers and the identifiers of the associated coordinators; and (II) the *membership cache*, where CB records the member identifier *mid* as soon as *mid* has left the group; such entry will be purged after a time sufficiently long to assume that no messages related to *mid* are still in transit [L93]. A group member *mid* wishing to leave the group sends to CB a message <LEAVE, *mid*, CB>. All coordinators have a copy of *memb-t*, whose updates are driven by CB.

A MH may send NEW messages to its own coordinator and JOIN, LEAVE messages to CB. Such a communication may exhibit duplicates and non-FIFO arrival (Section 3.2). The related problems are solved as follows.

- A1) Each coordinator discards NEW messages carrying member identifiers that are not in *memb-t*.
- A2) CB discards any LEAVE message carrying a member identifier that is not in *memb-t*.
- A3) CB discards any JOIN message carrying a member identifier that is either in *memb-t* or in the membership cache.

A1 discards duplicates of NEW messages originated by hosts that have already left the group, or that are group members but constructed the NEW message in a previous participation to the group activity (i.e., before leaving and re-entering the group). A2 is the same for LEAVE messages. A3 allows CB to discard duplicate JOIN's (i.e., a duplicate JOIN arriving after the matching LEAVE should not let the sending host become again a group member).

Membership changes are propagated through NORMAL messages that CB generates upon processing JOIN and LEAVE messages. The Order field of such NORMAL messages tells T-Cast () and a bit field specifies that these messages carry membership changes¹. Their payload contains: (I) a description of the new membership; (II) the identifier of the coordinator of each just-joined member; (III) the current value of *cseq* at all coordinators, that CB obtains through a proper message-exchange round.

A host *M* joins the group as follows: (i) start listening to messages directed to the group; (ii) upon receiving a message *m1* sent by CB, assign *mystable[CB] = m1.cseqB*; (iii) send a <JOIN, *M*, *join-id*, CB> message reliably; (iv) start executing the protocol except that: (iv-a) it ignores messages sent by coordinators other than CB; (iv-b) it applies the delivery condition *Delivered(CB, m.cseqB - 1)*; (iv-c) it discards every message that could be delivered. As soon as *M* receives the NORMAL message whose payload notifies about its joining, it extracts the values of *cseq* for initializing all entries of *mystable* and switches to the “ordinary” protocol.

A group member *mid* leaves the group as follows: (i) send a <LEAVE, *mid*, CB> reliably; (ii) keep on executing the protocol until it can deliver the NORMAL message *m1* whose payload notifies about its leaving; (iii) send a <LEAVEOK, *mid*, CB> reliably. At this point, *M* can stop participating in the protocol. The LEAVEOK acknowledges the delivery of *m1* and of all preceding messages. Upon receiving the LEAVEOK, CB will free all resources allocated to *mid* and will instruct the other coordinators (and gateways [B98]) to do the same. In

¹ The transmission of these messages to group members that are not interested to membership changes may be skipped simply [B98].

particular, coordinators will not wait anymore for stability information about mid. It could be possible to allow a group member mid to leave the group as soon as it completes step (i), which would result in a looser semantics. In this case, the resources held by mid would be freed when CB receives the LEAVE.

4 Simulation Environment

To analyze the performance of the protocol we developed a discrete event simulation model and implemented it in C++ language. This section describes the assumptions underlying the simulation model, the performance measures taken into consideration and the motivations based on which the parameter values have been set. The results of the simulation experiments are discussed in Section 5.

4.1 Assumptions

In order to keep the complexity of the simulation model low and, consequently, shorten the execution times of the experiments we introduced the following simplifying assumptions on the simulated system.

We considered a single multicast group which was assumed to be static for the whole duration of the simulation experiment. Hence, the membership protocol was not implemented. Furthermore, we assumed that all the group members are located on mobile hosts and there is a single group member on each mobile host.

Network protocols located between the multicast protocol and the network interface were not included explicitly in the simulation model though some meaningful aspects of them were taken into consideration, e.g., the reordering buffer (see below). Therefore, the conceptual internal organization of the group member can be thought of as shown in Figure 4 (left).

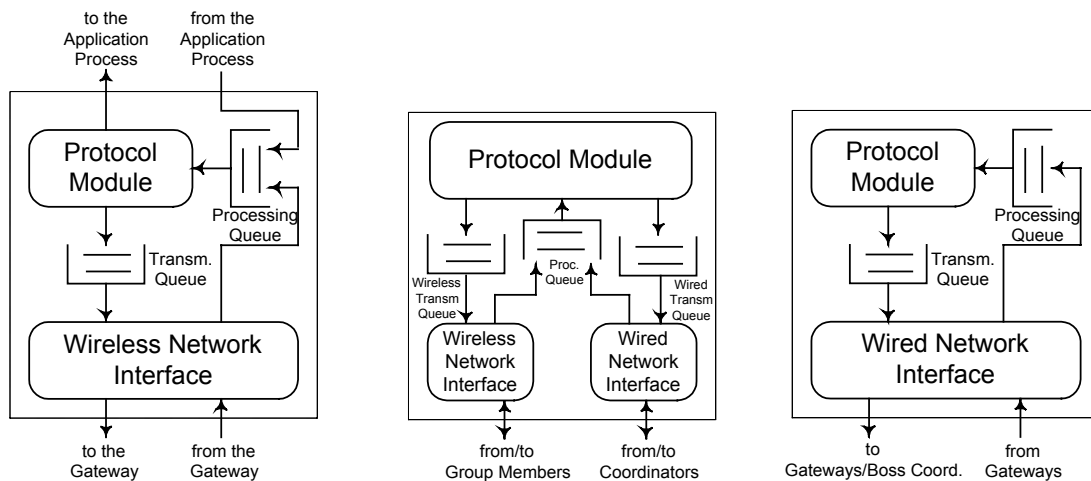


Figure 4: Simulation models of group member (left), gateway (middle) and coordinator (right).

The *Protocol Module* implements the multicast protocol and receives messages generated by the *Application Process*. However, if the *Protocol Module* is currently processing another message, these messages are temporarily queued in the *Processing Queue*¹. Once a message has been processed by the *Protocol Module* is passed to the *Wireless Network Interface* which is responsible for the transmission on the wireless medium toward the associated gateway. As for the *Protocol Module*, if the *Wireless Network Interface* is busy the message is queued in the *Wireless Transmission Queue*. Messages received from the gateway through the *Wireless Network Interface* are passed up to the *Protocol Module* unless the latter is processing another message: in that case the message is put temporarily in the *Processing Queue* (it was assumed there is a single *Processing Queue*). From the *Protocol Module* the message passes to the *Application Process*: the *Protocol Module* includes the *Mybuf* buffer where messages are put until the delivery condition is satisfied. Note that the *Mybuf* buffer is different from the *Processing Queue*: the latter stores messages waiting to be processed while the former stores messages waiting for delivery.

The gateway and coordinator models are also shown in Figure 4 (middle and right, respectively). As far as the coordinator model, the only difference with respect to the group member model is the *Wired Network Interface* (and the related *Wired Transmission Queue*) instead of the *Wireless Network Interface* (and the

¹ The *Processing Queue* is part of *Protocol Module* but is shown as external for ease of presentation. For the same reason the *Transmission Queues* are shown as external to the *Network Interfaces*.

related *Wireless Transmission Queue*): coordinators are connected to the wired network and communicate with gateways (or with the Team Boss coordinator). On the other hand, gateways have both a Wireless and a Wired Network Interface (and the related Transmission Queues) to communicate with group members and coordinators, respectively.

In our simulation model it was assumed that time intervals between consecutive messages generated by the Application Process are random variables with a predefined distribution. According to a common practice, in the experiments we assumed that these time intervals are exponentially distributed¹ (i.e., the message generation process is Poisson).

Our protocol assumes that the communication in the wired network is reliable and FIFO ordered but, generally, messages experience a random propagation delay. To simulate FIFO communications with random delays we modeled FIFO channels as a pipe with a *reordering buffer* at its end. The pipe introduces a random delay which takes into account the total propagation delay from the sending node to the destination node (i.e., propagation times in communication links, buffering and transmission times at intermediate routers, if any). Due to the random propagation delay messages may arrive at the destination out-of-order. The reordering buffer is then used to restore the original order. In real distributed systems message reordering is usually done at the transport layer (e. g., by the TCP protocol). Therefore, the reordering buffer, not shown in models of Figure 4, has to be located between the Wired Network Interface and the Processing Queue.

The protocol also assumes that wireless communications are FIFO ordered but unreliable. In this case FIFO order does not require any reordering buffer since it is guaranteed by the *stop and go* protocol by which a group member and the corresponding gateway synchronize each other². On the other hand, the unreliability of the wireless medium was modeled by assuming that messages get lost with a predefined probability P_l and that consecutive message losses are independent. The wireless transmission time was assumed to be a random variable with minimum value given by the ratio between the message size and the wireless bandwidth. However, the transmission time may also include additional components, like contention delays in contention based WLAN (e.g., 802.11 WLAN). It was also assumed that the wireless network is broadcast so that a message is multicast to all group members by a single wireless transmission.

Finally, as far as host mobility, in our simulation model group members were assumed to move as freely as possible. Each group member remains in a cell for a time, the *cell permanency time*, which was assumed to be a random variable exponentially distributed as in [AV97]. After this time the group member moves either out of the covered area (with a probability given by the *out-of-coverage probability*) or in *any* other cell. In the simulation experiments discussed below it was assumed that all cells have the same probability to be accessed, but, in general, any discrete probability distribution can be used. If the coverage of the wireless network is incomplete, after leaving a cell a group member may go in the uncovered area where it remains for a time interval, the *outside permanency time*, which was assumed to be a random variable with an exponential distribution.

4.2 Performance Measures

In this section we introduce the performance measures (or indices) used to characterize the performance of the protocol. The main performance index is the *average end-to-end delay* (or *average latency*) defined as the average time elapsed from the time instant at which a multicast message is generated at the sending group member to the time instant at which the same message is delivered by a destination group member.

To better analyze the performance of the protocol, it is useful to consider the components of the average end-to-end delay, i.e., times spent by a single message along its way from a sending group member mid_s to a receiving group member mid_r . These components are shown in Figure 5 which refers to the case when no message loss occurs (the length of time intervals does not reflect their actual value). When messages are lost, the end-to-end delay of a single message includes a further component, the retransmission delay, which is the time necessary to recover the message.

Our protocol corresponds to the three upper layers, i.e., those denoted as processing, buffering and queuing for processing. Queuing delays (for processing) arise since messages are processed in sequence and are experienced in the Processing Queue (see Figure 4) while buffering delays are experienced when the message

¹ The exponential distribution function is $F(t) = 1 - e^{-rt}$ where r is the message rate, i.e., the number of messages generated per time unit.

² The group member does not send another NEW message until the ACK related to the previous message has been received.

has to be delayed because it arrived out-of-order. The transmission delays are the times necessary to actually put bits on the wireless/wired medium while queuing delays for transmission are experienced in the Transmission Queues (see Figure 4).

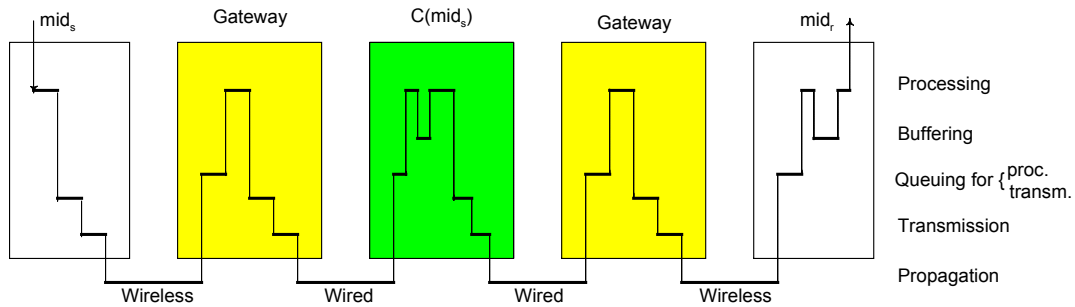


Figure 5: Delays experienced by a message from the sending group member (mid_s) to a receiving group member (mid_r).

In our simulation model we assumed that processing times are constant, but different at coordinators, gateways and group members. Furthermore, processing times depend on the message tag. Buffering and queuing times cannot be predicted in advance and depend on a number of factors, including message generation rate and group size. One of the valuable results of our simulations are estimates of buffering times and queuing times. We anticipate that queuing delays at gateways, and specifically at the wireless interface between gateways and group members, are predominant. This result is particularly significant because this delay component is due to the mismatch between the wired bandwidth and the wireless bandwidth, hence gateways are likely to be the ultimate bottleneck of any multicast protocol for distributed mobile systems. The transmission and propagation times have been modeled by random variables whose average values and distribution will be given later. Of course, wired and wireless medium are differentiated.

In some simulation experiments, in addition to the end-to-end delay, we also estimated other performance measures. In particular, to evaluate the effects of host mobility on the performance of the protocol we estimated the *average number of duplicates* per message and the *average number of retransmissions* per message.

The *average number of duplicates* per message is defined as the average number of times a message is received by the destination group member minus 1. Assuming that the retransmission mechanism does not cause duplicates, if group members are quiet each message is received *exactly once*. On the other hand, when group members are mobiles duplicate messages can be received as a consequence of cell switching. Hence, the average number of duplicates is a measure of the additional overhead (i.e., additional consumption of wireless bandwidth, computing power and energy resources) caused by host mobility.

Similar to the previous performance measure is the *average number of retransmissions* per message, defined as the average number of times a message is retransmitted by the gateway to the destination group member. Clearly, if the wireless network provides a reliable service and all the group members are quiet messages are never lost and each message need to be never retransmitted. On the other hand, even if the wireless network is reliable messages can be lost due to the mobility of group members. Hence, under the assumption of reliable wireless network the average number of retransmissions per message gives a measure of the overhead caused by host mobility. Instead, in a scenario where the wireless network is not reliable the above index take into account the retransmissions due to both wireless network unreliability and host mobility. However, as it is shown in the Appendix, the contribution due to the network unreliability can be analytically derived and, hence, the contribution due to host mobility can be determined by difference.

4.3 Parameter Setting

The performance of the protocol is, in principle, influenced by a very large set of parameters: system parameters like the group size, the number of coordinators, and so on; protocol parameters like the size of the `Mybuf` buffer, the size of the gateway cache, etc; and group member parameters like the message rate or the mobility pattern. Even considering only few values for each parameter the number of possible scenarios to analyze becomes unmanageable.

Therefore, in this paper we use the following approach. We define a *Basic Scenario* by selecting a set of operation parameters in such a way that they represent a typical utilization environment for the protocol and analyze the protocol performance with reference to this Basic Scenario. Then, to investigate the influence of

the main parameters we start from the Basic Scenario and vary the value of only one of these parameters at a time. The set of system, protocol and group member parameters and the corresponding values in the Basic Scenario are reported in Table 1, Table 2 and Table 3, respectively.

4.3.1 System Parameters

We assumed that the Basic Scenario corresponds to the case when the protocol is used in a local environment like a Campus or a building where each room corresponds to a wireless cell. Accordingly, the number of gateways was fixed to 50 while the number of group members was chosen equal to 100, among which 40 actually multicast messages while the remaining ones only receive messages sent by other group members. The number of coordinators is not included in Table 1 since in the simulation experiments we varied this number between 1 and the number of transmitting group members.

We assumed that the wired network is a high speed network with a bandwidth equal to 100 Mbps (as in FDDI and Fast Ethernet LANs) while the wireless bandwidth was set to 10 Mbps. As it will better clarified in Section 5.2, the value of the wireless bandwidth was intentionally chosen greater than the bandwidth provided by current Wireless LANs (in the order of 1 Mbps) to banish the doubt that using a single coordinator may create a bottleneck for the system performance when the wireless bandwidth increases.

The values of the *propagation delays* were selected accordingly to the local environment where the protocol is operating. Since wireless cells are supposed very small in size (their diameter is approximately ten meters) the propagation delay in the wireless cell was assumed to be negligible. Instead, the average propagation delay on the wired network was assumed to be a random variable with a uniform distribution in the range [0.5 - 2.5]¹. Note that the propagation delay does not include the delay introduced by the reordering buffer (see Section 4.1) to enforce FIFO ordering.

The message loss probability, which models the wireless network unreliability, was set to 0.001. For ACK messages the loss probability was assumed to be null due to their small size. However, ACK messages can be lost due to the mobility of group members.

Finally, the *out-of-coverage probability* gives the probability that, when exiting a cell, a group member moves in the non covered area. Of course, this probability depends on the coverage of the wireless network. In the experiments discussed below it was assumed to be null, which means that the coverage is complete. However, we have planned experiments with incomplete coverage in order to assess its influence on the quality of service achieved by group members and on the system performance.

Parameter	Value
Number of group members (N_m)	100
Number of sending group members (N_s)	40
Number of gateways (N_g)	50
Wired Bandwidth	100 Mbps
Wireless Bandwidth	10 Mbps
Wired Propagation Delay	[0.5 - 2.5] msec
Wireless Propagation Delay	0.0 msec
Message loss probability (P_l)	0.001
ACK message loss probability	0.0
Out-of-coverage probability	0.0

Table 1: System parameters

4.3.2 Protocol Parameters

For sake of simplicity, NEW, NORMAL, TRANSFER and NACK messages were assumed to have the same size of 512 bytes, while the ACK size was set to 50 bytes since ACK messages do not include any payload. For all the message types, the above sizes include control information introduced by both the multicast protocol and the underlying network protocols and - for NEW, NORMAL and TRANSFER messages - the payload as well. The beacon message is sent periodically and the period was set to 100 msec as in [GS94].

Processing times at group members were assumed to be negligible given the simplicity of the actions group members execute. On the other hand, to determine processing times at gateways and coordinators we used

¹ The average value of 1.5 msec was estimated by using a large number of samples obtained by pinging remote hosts in a local environment (the campus of the Faculty of Engineering at the University of Pisa).

the following approach. We measured the average CPU times the simulator needs to perform protocol actions on different message types and used these times as an approximations of the real processing times¹. Even though the simulator does not implement the protocol as exactly as a real system, however, estimated times can be considered a good approximation.

For simplicity we assumed an equal value for the processing times at the coordinators of NEW and FETCH messages. This is not true in reality since FETCH messages require more complex operations and, hence, would need a longer CPU time. However, the number of FETCH messages managed by a coordinator is noticeably less than the number of NEW messages. Therefore, in our simulation experiments we considered a single value obtained as the weighted sum of CPU times of NEW and FETCH messages, respectively (weight being the relative fractions). Instead, for the processing times at gateways we used a different value for each message type. As shown in Table 2, the processing time for NACK messages is much greater than the processing times of NEW and NORMAL messages since the gateway need to look for the requested messages in the local cache.

The size of the gateway cache was selected in such a way to achieve a good hit ratio in the operating condition characterizing the Basic Scenario. The size of the group member `Mybuf` buffer was chosen as a trade-off between its memory requirements and the ability to store all messages not yet delivered (recall that the protocol is correct even though there is no more space in `Mybuf` and some received message has to be discarded). We started with a buffer of 1000 512-bytes messages, corresponding to approximately 512 Kbytes which seems affordable even for palm-top computers. However, we found that even a buffer of 100 512-bytes messages (less than 64 Kbytes of memory) does not degrade performance significantly (see Section 5.5).

The *retransmission timeout* is the time interval the group member uses to set the retransmission timer after sending a NEW message. If the timer expires before the related ACK has been received from the gateway the message is retransmitted and the timer set again. Of course, the value of the retransmission timeout must be greater than the sum of the transmission times of the NEW message and the related acknowledgement but it is also necessary to take into account possible queuing delays which arises when the traffic in the wireless networks is high. The value of 20 msec was chosen by estimating (via simulation) the time interval between the transmission of a NEW message and the reception of the related acknowledgement when the offered load in the wireless network is in the order of the 80% of the capacity.

Parameter	Value
NEW/NORMAL/TRANSFER message size	512 bytes
ACK message size	50 bytes
NACK message size	512 bytes
Message processing time at group members	0 μ sec
Message processing time at coordinators	25 μ sec
NACK message processing time at gateways	1000 μ sec
NEW message processing time at gateways	25 μ sec
NORMAL message processing time at gateways	290 μ sec
<code>Mybuf</code> buffer size	1000 messages
Gateway cache size	1000 messages
Retransmission timeout	20 msec
NACK timeout	40 msec
Beacon message size	64 bytes
Beacon period	100 msec

Table 2: Protocol parameters

The *NACK timeout* is used to set a timer after a NACK message has been sent and until this timer has expired the group member is not enabled to send any further NACK messages. Hence, from one hand, the NACK timeout should be long enough to allow all the requested message to be received by the group member. On the other hand, it should be as short as possible so that lost messages detected after a NACK message has been sent can be requested readily. Based on these two conflicting requirements and after some (simulated) trials the NACK timeout was set to 40 msec.

4.3.3 Group Member Parameters

This section introduces the group member parameters. The *cell permanency time* is the time interval a (mobile)

¹ The simulator was running on a PC based on a 200 MHz Pentium Pro processor with 64 Mbytes of memory and Linux operating system and it was the only active user process.

group member remains in a wireless cell while the *outside permanency time* is the interval during which the (mobile) group member is out of any cell, under the hypothesis that the geographical coverage of the wireless network is incomplete. Both the cell and outside permanency times were assumed exponentially distributed. The mean cell permanency time was chosen according to the following considerations. Assuming that the diameter of a cell is about 10 m and that group members move with a uniform speed then a mean cell permanency time of 10 sec corresponds to a speed of 3.6 km/h which is approximately the speed of a person walking within a building. In the Basic Scenario it was assumed that the wireless network has a complete coverage and, hence, the mean outside permanency time was set to zero.

Finally, we assumed that messages are generated at group members according to a Poisson process with a rate of 20 messages/sec (approximately 80 Kbps). Since, the number of group members actually sending messages is 40 the resulting aggregate message rate is 800 messages/sec (approximately 3.2 Mbps).

Parameter	Value
Mean cell permanency time	10 sec
Mean out-of-coverage permanency time	0 sec
Message rate	20 mess/sec

Table 3: Group member parameters

5 Results

This section is devoted to the discussion of the results obtained by simulation. These results were estimated by using the independent replication method and assuming a confidence level of 90% [LK82]. We first discuss the results related to the Basic Scenario and then analyze the influence of the main parameters on the performance of the protocol. In the experiments it was assumed that all messages are issued by the C-Cast () primitive, i.e., they require Causal order.

5.1 Basic Scenario

Figure 6 (left) shows the average end-to-end delay as a function of the number of coordinators (N_c) in the Basic Scenario: the average end-to-end delay slightly decreases as the number of coordinators increases. This shape is the result of two contrasting effects due to the buffering and the queuing components, respectively. Figure 6 (right) shows that, as the number of coordinators grows, the average queuing delay decreases while the average buffering delay slightly increases. However, the queuing component is largely predominant irrespective of the number of coordinators.

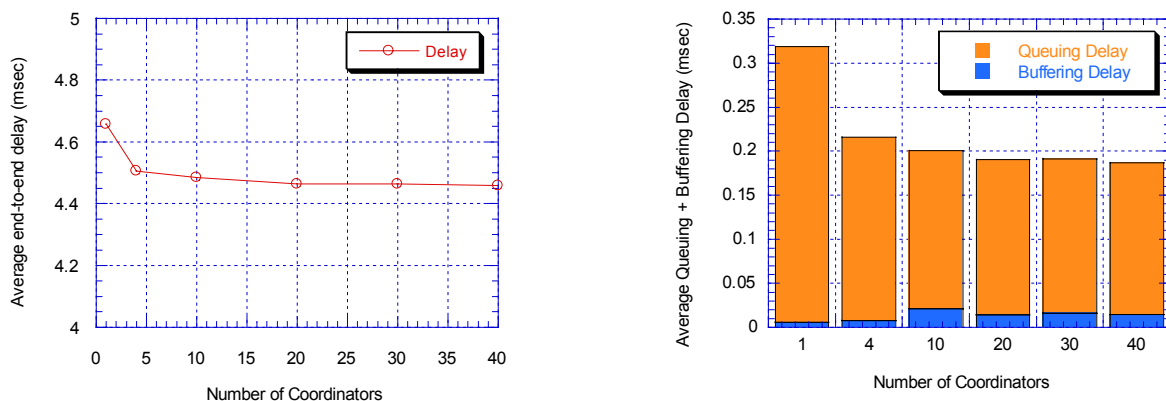


Figure 6: Average end-to-end delay as a function of the number of coordinators (left). Buffering and queuing delay components (right).

It is interesting to note that the buffering delay slightly increases with the number of coordinators, while one would have expected exactly the opposite. With more coordinators, each coordinator has to manage less group members, hence the likelihood of introducing «false» causal precedence relationships among messages originated by different group members decreases. It follows that unnecessary message buffering should decrease. We interpret the increasing of the buffering delay resulting from simulation results as follows. With more coordinators, a group member takes more time to detect a message loss, because messages coming from the same coordinator arrive less frequently (lost messages are detected only when the next

message from the same coordinator is received). Hence, messages causally dependent from lost messages tend to remain in the Mybuf buffer for a longer time. As it turns out, this effect predominates over the decrease of unnecessary buffering.

As we shall see in Section 5.2, Table 5, the queuing delay is primarily due to the waiting times messages experience in the Wireless Transmission Queue at the destination gateway, i.e., for the wireless transmission to group members. Instead, waiting times in the Wired Transmission Queue (for transmission from gateways to coordinators and back) and in the Processing Queue both at coordinators and gateways are negligible. The reasons are as follows: (i) the wired bandwidth is one order of magnitude greater than the wireless bandwidth; (ii) processing times are small compared with the transmission times of the wireless network (with exception for NACK messages whose fraction, however, is very low); (iii) coordinators process less messages than gateways.

Waiting times in the Transmission and Processing Queues at coordinators decrease when there are more coordinators, because each coordinator has to manage less messages. However, such decrease has no meaningful effect on the overall delay because waiting times at coordinators are negligible with respect to waiting times at the destination gateway (even when there is a single coordinator).

On the other hand, the number of messages to broadcast in the wireless cell served by a gateway does not depend on the number of coordinators. Hence, the waiting time in the Wireless Transmission Queue at gateways should remain constant. Instead, the simulation experiments show that this waiting time decreases when the number of coordinators increases. This decrease is caused by the reordering buffer used to enforce FIFO order in the communications between gateways and coordinators (see Section 4.1). In fact, consider the case when the destination is a gateway: due the reordering process messages received from the coordinator are passed to the Protocol Module in batches, i.e., back to back. After processing at the Protocol Module, the same messages are passed down to the Wireless Network Interface for transmission to group members. Since processing times are small compared with wireless transmission times, messages arrive at the Wireless Network Interface still in batches and, thus, some of them experience a waiting delay in the Wireless Transmission Queue. Of course, the batch size and, hence, the average waiting time in the Wireless Transmission Queue, depends on the number of coordinators: the less the number of coordinators, the more the number of messages arriving back to back at the Wireless Network Interface and, hence, the more the average waiting time. This justifies the decrease in the queuing delay component when the number of coordinators increases. Of course, without the reordering buffer the queuing delay would be approximately constant or slightly decreasing.

5.2 Scalability

In this section we assess the scalability of the protocol, i.e., how the protocol performs when the number of group members become larger and larger. First we analyze the case in which the number of group members grows but the number of sending group members remains constant, then the case in which also the latter grows.

Plots in Figure 7 show that the average end-to-end delay is not influenced in a meaningful way by the group size. Several features of the proposed protocol contribute to make it very well scalable (in terms of receiving group members), including: a single wireless transmission for broadcasting within each cell (unlike [AB96]); absence of hand-off upon cell switching; cache of past messages at gateways.

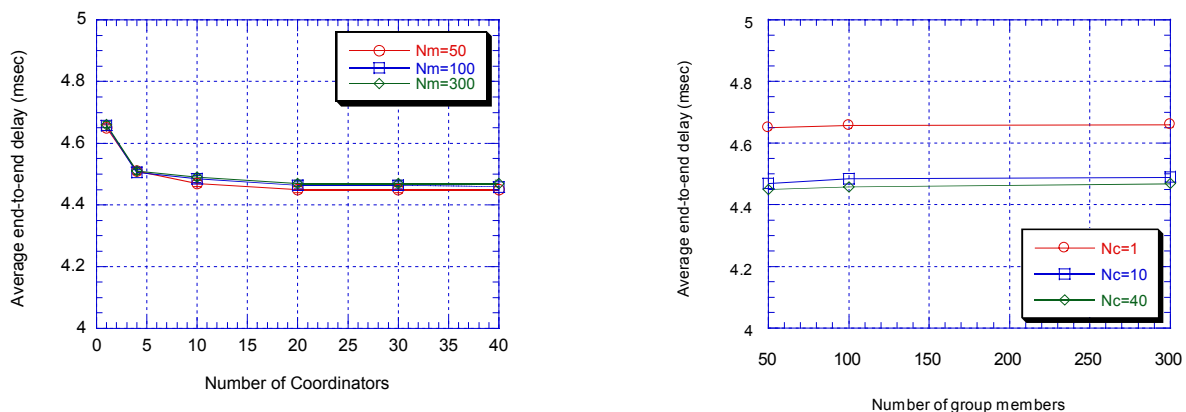


Figure 7: Influence of the number of group members on the average access delay.

However, when the group size grows it is reasonable to expect that the number of sending group members (N_s) increases accordingly. Therefore, it is important to analyze the influence of this parameters on the protocol performance. Plots in Figure 8 show that, as expected, the average end-to-end delay increases when the value of N_s goes up. This can be justified by observing that a greater number of sending group members implies a higher aggregate message rate and, hence, greater queuing delays.

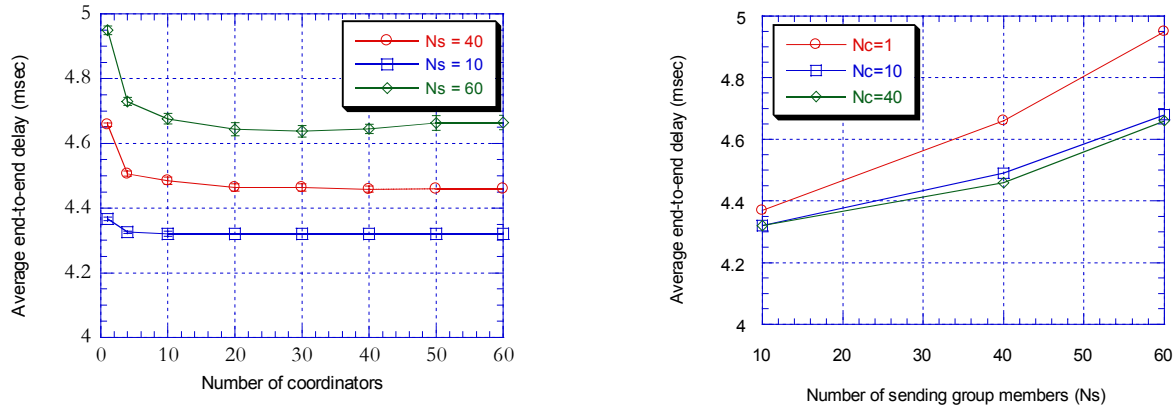


Figure 8: Influence of the number of sending group members on the average access delay.

Furthermore, Figure 8 (left) shows that, for a fixed value of N_c , the delay variations due to the number of coordinators are more evident when the number of sending group members (i.e., the aggregate message rate) is higher. Remembering that these variations are basically due to the fact that messages are extracted from the reordering buffer in batches (i.e., back to back) and that batch sizes increase with the message rate (see Section 5.1) it is easy to understand the reasons of this behavior.

Figure 8 (right) highlights that the number of coordinators determines the slope by which the average end-to-end delay curve grows when the number of sending group members increases. The lower the number of coordinators, the greater this slope. This means that using a larger number of coordinators makes the system more scalable.

A key question is whether using a single coordinator may create a bottleneck when the aggregate message rate becomes larger and larger. By simulation we found that the likely bottleneck of the system, rather than the single coordinator, appears to be the wireless interface at *gateways*. Table 4 reports - in addition to the average end-to-end delay - the average total delays experienced by a message at the (receiving) gateway and at the coordinator, respectively, for aggregate message (bit) rates up to 2000 512-bytes messages/sec (8.2 Mbps¹).

N_s	Message Rate (mess/sec)	Bit rate (Mbps)	Average end-to-end delay (msec)	Average total delay at gateway (msec)	Average total delay at coordinator (msec)
10	200	0.8	4.36	0,73	0.066
40	800	3.2	4.65	0.88	0.066
60	1200	4.9	4.94	1.05	0.066
80	1600	6.5	5.28	1.25	0.066
100	2000	8.2	6.03	1.78	0.066

Table 4: Average delay components experienced at gateways and at the coordinator.

As it clearly appears, the average total delay at the coordinator is small when compared to the average total delay at the receiving gateway. The total delay experienced at either a coordinator or a gateway, has three components: queuing, transmission and processing. The transmission and processing delays are fixed. Instead, mainly due to the mismatch between the bandwidth of the wired and wireless networks, the queuing component experienced at the destination gateway (both for processing and transmission) tends to

¹ The actual aggregate bit rate on the wireless network is even greater since it includes traffic caused by ACK, NACK, TRANSFER and retransmitted NEW messages

become the predominant factor of the end-to-end delay (apart from the propagation delays) and increases as the aggregate message rate increases approaching the wireless network's capacity while the queuing delay at the coordinator is negligible (see Table 5). As a final consideration, note that the wireless bandwidth we considered in our experiments (10 Mbps) is greater than that provided by current WLANs (1-2 Mbps) and the transmission to group members in a cell is achieved by a single wireless transmission. Without these assumptions the limitation due to (destination) gateways would be even more evident.

N_s	Message Rate (mess/sec)	Bit rate (Mbps)	Average queuing delay at gateway (msec)	Average queuing delay at coordinator (msec)
10	200	0.8	0.03	0.0
40	800	3.2	0.18	0.0
60	1200	4.9	0.35	0.0
80	1600	6.5	0.55	0.0
100	2000	8.2	1.08	0.0

Table 5: Average queuing delays experienced at the gateway and the coordinator.

We argue that even the limitation due to the wireless interface between gateways and group members is not a peculiarity of our protocol. Instead, it is a consequence of the coexistence of different network technologies with significantly different bandwidth. It follows that any multicast protocol based on a "two-tier" architecture like ours (i.e., in which MHs offload storage, processing and transmission load to SHs) [AB96], is likely to exhibit a bottleneck at gateways. The observation that gateways are likely to constitute an intrinsic potential bottleneck is particularly useful, because it allows using a simple, logically centralized coordinator-based protocol.

5.3 Influence of host mobility

This section is devoted to investigate the sensitivity of the protocol to the mobility of group members. To quantify mobility we refer to the mean cell permanency time (MCP), i.e., the average time a group member remains in a cell. Figure 9 (left) shows the average end-to-end delay vs. the number of coordinators for three different values of MCP : 1 sec, 10 sec (as in the Basic Scenario), and a value approaching the infinity. Assuming that group members move at uniform speed and that the cell diameter is approximately 10 m, the above values of MCP correspond to speeds of 36, 3.6 and 0 Km/h, respectively.

Figure 9 (left) shows that the average end-to-end delay tends to increase as the mean cell permanency time increases while the shape of the curve does not vary meaningfully with respect to the Basic Scenario. This behavior can be easily justified by observing that when the frequency of cell switches increases the number of missed messages increases as well. However, even in presence of highly mobile group members, the number of messages missed due to mobility is very small.

This result is particularly significant because: (i) it shows that the influence of mobility on the performance of the protocol is practically negligible, which is a desirable feature for a protocol for mobile systems; and (ii) from a broader point of view, it is probably not worthwhile designing multicast protocols that insist on highly optimized solutions for coping with message loss due to mobility, as this effect does not occur often.

Figure 9-right shows the average number of retransmissions per message due to both network unreliability (i.e., message losses in the wireless network) and host mobility. In the appendix it is proven that the contribution due to message losses is approximately equal to $P_l=0.001$ (see equation [3] in the Appendix) and, hence, the difference is caused by host mobility. From Figure 9-right it emerges that the percentage of messages retransmitted due to cell switches is different from zero only when host mobility is high and, even in that case, it is less than or equal to 0,3%, i.e., extremely low. And, in fact, the increase of the average end-to-end delay with respect to the Basic Scenario is very small (0.10 msec in the worst case).

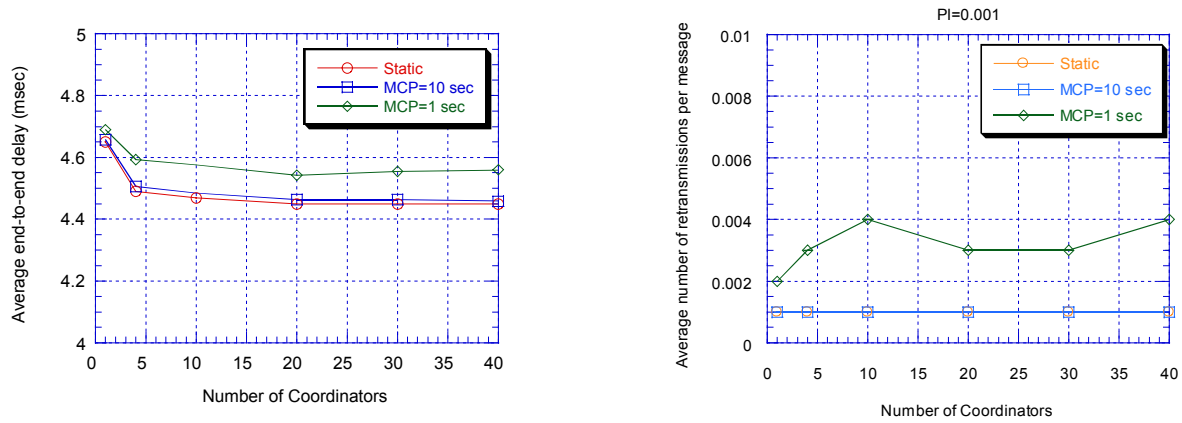


Figure 9: Average end-to-end delay for several mobility patterns (left). Average number of transmissions per message as a function of group member mobility (right).

We also found that the number of messages discarded by group members as duplicate messages is very small (less than 1 every 1000). These results show that it is probably a good idea to not build multicast protocol over hand-off procedures: hand-off would prevent the arrival of duplicates 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.

5.4 Influence of (wireless) network unreliability

In addition to host mobility there are other reasons by which messages may be lost: incomplete coverage of the wireless network, physical obstructions inside cells, environmental noise and so on. The influence due to mobility has been investigated in the previous section. In this section we analyze the protocol sensitivity to the unreliability of the wireless network. To this end we ran simulation experiments by considering three increasing values for the message loss probability P_l (see Figure 10-left). As expected, the average end-to-end delay is greater when the wireless network is more unreliable since (i) there are more messages to be recovered by retransmission; (ii) a larger number of messages need to be buffered waiting for previous messages to be recovered; (iii) retransmitted messages increase the traffic on the wireless network. Figure 10-right shows that when the message loss rate is high, as a consequence of (ii), the buffering delay is no longer negligible as in the Basic Scenario (see Figure 6-right for comparison). Furthermore, as a consequence of (iii), the average queuing delay increases as well with respect to the Basic Scenario.

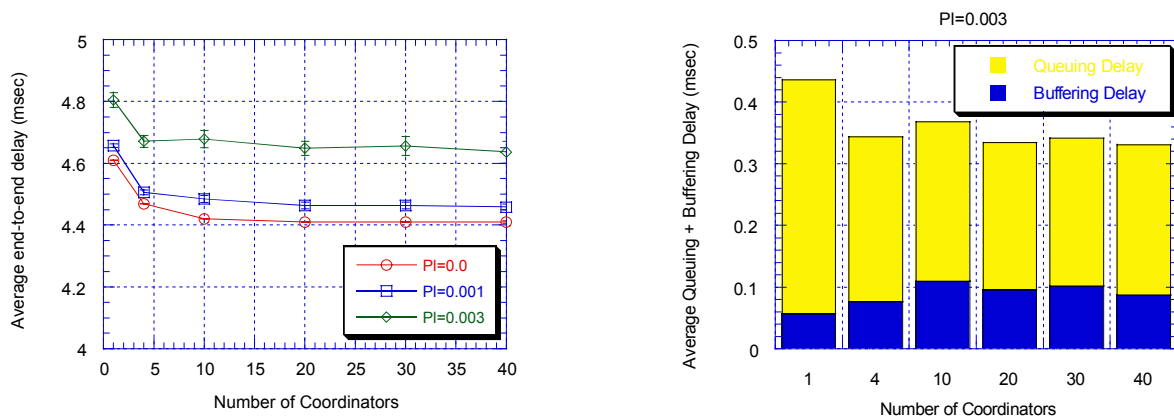


Figure 10: Average end-to-end delay for different message loss probabilities (left). Buffering and queuing delay components (right).

It is worthwhile to observe that, even increasing the message loss probability the average queuing delay continues to be the predominant delay component, apart from the propagation times which, however, cannot be avoided. By remembering that queuing times are experienced mostly at the Wireless Transmission Queue of the destination gateway, the results of Figure 10-right confirms that the architecture based on coordinators is well suited.

5.5 Influence of the Buffer Size

The `Mybuf` buffer is used to increase the performance of the protocol. Careful sizing of this buffer is of important since an undersized buffer could severely degrade the protocol performance but, on the other hand, an oversized buffer would waste memory space which might be a limited resource on some mobile computers. It should be pointed out, however, that the optimal buffer size depends on many factors and mainly on the message loss probability and the aggregate message rate. If messages are lost rarely (e.g., because the underlying wireless networks provides a reliable service, the geographical coverage is complete, and mobiles move very slowly) the protocol works well even with a small or null buffer. On the other hand, if messages are lost very often a larger buffer is required. To understand the influence of the aggregate message rate note that when a message loss is detected and before the same message is recovered by retransmission the group member receives a certain number of messages which cannot be delivered and thus need to be buffered: of course, this number is proportional to the aggregate message rate.

In this section we investigate the influence of the buffer size assuming the Basic Scenario as operating scenario. Figure 11 shows the shape of the average end-to-end delay, as a function of the number of coordinators, for buffer sizes equal to 100 and 1000 512-bytes-messages, respectively. We also considered a buffer with a capacity of only 10 messages but simulation experiments clearly showed that this size is too small when operating in the Basic Scenario. The consequence is that a large number of messages are lost due to lack of buffer space at the group members and need to be retransmitted by the gateway. This makes the number of messages waiting for transmission or processing at gateways grow without no limit up to a complete saturation of the gateway resources.

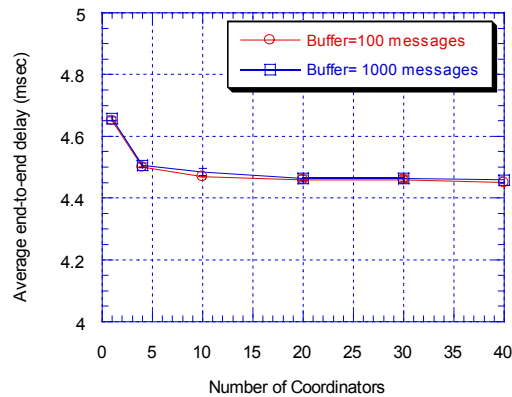


Figure 11: Average end-to-end delay vs. the number of coordinators for several buffer sizes

On the other hand, curves related to buffer sizes of 100 and 1000 do not show significant differences (see Figure 11). This means that a 100 message buffer capacity is sufficient when operating in the Basic Scenario. Considering that the message size is 512 bytes the required amount of memory is less than 64 Kbytes which is affordable even for palm-top computer.

5.6 Influence of the Cache Size

This section analyzes the influence of the gateway cache size on the protocol performance. It is clear that an insufficient cache size may cause a large number of `FETCH` requests from gateways to coordinators with an evident increase in the average end-to-end delay experienced by messages. Like the size of the buffer at group members, the optimal cache size depends on the conditions under which the system is operating (host mobility, message loss probability, coverage of the wireless network, and son on). Unlike the buffer at group members, however, there are not strong limitations for the cache size since gateways are stationary hosts with a lot of resources.

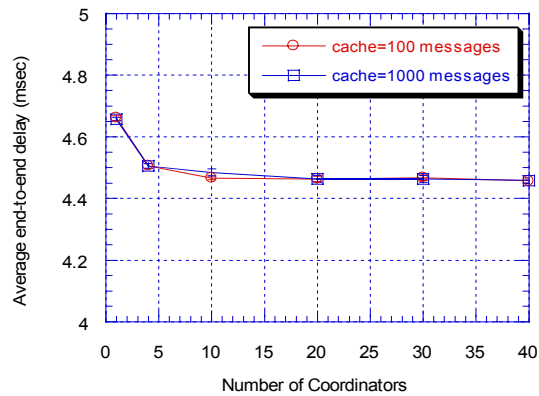


Figure 12: Average end-to-end delay vs. the number of coordinators for several cache sizes

Figure 12 reports the average end-to-end delay for two different cache sizes (100 and 1000 messages, respectively) assuming that the operating conditions are those specified in the Basic Scenario: the curves do not exhibit meaningful differences. A third cache size, equal to 10 messages, was considered but the related curve is not reported in Figure 12 since this size is too small and makes the system become unstable. This is due to the fact that, most of the time, gateways do not find messages requested by group members (by NACK messages) in the local cache and must ask for them to the coordinator. This increases the processing time at gateways and makes the number of messages in the Processing Queue at gateways increase without limit.

6 Concluding remarks

We have presented and evaluated a multicast protocol for distributed mobile systems that accommodates neatly three increasingly strong delivery ordering guarantees: FIFO, Causal and Total. The protocol is flexible because it allows tuning the resources on the wired network in order to achieve the desired trade-off between run-time costs and performance. The system model assumed is quite general and includes incomplete spatial coverage of the wireless network while no limitations are posed on the mobility pattern of group members. Furthermore, movements of group members need not trigger the exchange of any state information in the wired network. The protocol has been designed by keeping in mind that MHs may have scarce resources.

Simulation results show that the protocol scales well with the group size while the limiting factor seems to be the wireless network. Even considering a wireless bandwidth greater than that provided by current WLANs and assuming that message broadcast in a wireless cell can be obtained by a single transmission, the main contribution to the average end-to-end delay (apart from the propagation delay) is due to waiting times for the wireless transmission at the gateway. Furthermore, this component tends to increase as the aggregate message rate goes up approaching the capacity of the wireless network.

Another important result is that the absence of the hand-off procedure does not seem to introduce prohibitive costs in terms of duplicate messages or useless retransmissions: their number is very limited. Finally, the protocol exhibits very good performance figures with a buffer size of 512 Kbytes which should be affordable even for palm-top computers.

Acknowledgements

The work described in this paper has been carried out under the financial support of the Italian Ministero dell'Università e della Ricerca Scientifica e Tecnologica (MURST) in the framework of the MOSAICO Project (Design Methodologies and Tools of High Performance Systems for Distributed Applications).

Michel Raynal provided us with comments on a draft version.

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.
- [AR95] M. Ahuja, M. Raynal, "An Implementation of Global Flush Primitives Using Counters", *Parallel Processing Letters*, vol. 5 (2), 1995, pp. 171-178.

- [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.
- [B93] K. Birman, "The process group approach to reliable distributed computing", *Communications of the ACM*, vol.36, n.12, pp.36-53, December 1993.
- [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.
- [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.
- [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.
- [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.
- [LK82] A. M. Law, W. D. Kelton, "Simulation Modeling and Analysis", McGraw-Hill Book Company, 1982.
- [PRS96] R. Prakash, M. Raynal, M. Singhal "An Efficient Causal Ordering Algorithm for Mobile Computing Environments", *Proc. of the 16th International Conference on Distributed Computing Systems*, May 1996, pp. 744-751.

Appendix

This appendix is aimed at finding the fractions of retransmissions per message due to network unreliability (i.e., message losses in the wireless network) and host mobility, respectively.

Let R denote the total number of retransmissions per message, R_l the number of retransmission per message due to network unreliability and R_m the number of retransmission per message due to host mobility. Of course, R is a random variable such that $R=R_l+ R_m$ and, hence,

$$E[R_m] = E[R] - E[R_l] \quad [1]$$

where $E[X]$ indicates the average value of the random variable X . Under the assumption that (i) ACK (NACK) messages are never lost; (ii) messages losses are independent and occur with probability P_l it is

$$Prob\{R_l = n\} = (1 - P_l)P_l^n \quad n = 0,1,2,\dots \quad [2]$$

which implies

$$E[R_l] = \sum_{n=0}^{\infty} n(1 - P_l)P_l^n = (1 - P_l)P_l \sum_{n=1}^{\infty} nP_l^{n-1} = \frac{(1 - P_l)P_l}{(1 - P_l)} = P_l \quad [3]$$

By introducing [3] into [1] it follows that

$$E[R_m] = E[R] - P_l \quad [4]$$

In conclusion, once the average total number of retransmissions (per message) is known, equation [4] provides average number of retransmissions (per message) due to host mobility.