On the Structuring of Reliable Multicast Protocols for Mobile Wireless Computing

Giuseppe Anastasi*, Alberto Bartoli**

*Dipartimento di Ingegneria dell'Informazione, Università di Pisa, Via Diotisalvi 2, 56126 Pisa, Italy
Fax: +39-050-5668522, 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

1 Introduction

Technological developments in computer and communication hardware are enabling the deployment of computing systems based on portable computers and wireless networking. Users may be equipped with hand-held computing devices and roam around freely while maintaining connectivity with a wired computing infrastructure through a number of wireless cells. Such architectures may engender novel applications and services in a number of sectors, such as on-site data collection, factory automation, traffic monitoring, emergency management and so on.

Abstractions that have proven their utility in traditional (i.e., static and wired) distributed computing generally require dedicated implementations for these new architectures, due to several peculiar factors. For example, mobile devices have severe constraints in terms of power consumption, the wireless bandwidth is typically one order of magnitude smaller than wired bandwidth, the energy cost of wireless communication is highly asymmetric transmission being much more costly than receipt. In this paper we are concerned with one such abstraction: reliable multicast in a group of processes composed of mobile hosts. By reliable multicast we mean that all multicasts are delivered, multicasts generated by the same host are delivered in order and that there are no duplicates.

Figure 1: Example of system

We consider a set of stationary hosts (SHs) connected by a wired network and a set of mobile hosts (MHs). Mobile hosts may move and communicate through wireless links. Communication between hosts occurs solely through message-passing. Some SHs, called mobile support stations (MSSs), may communicate also through wireless links. Each MSS defines a spatially limited cell covered by a wireless link (Figure 1). A 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 in the wired network and within a single cell is reliable and FIFO-ordered.

In the above architecture, reliable multicast could be layered over existing multicast protocols designed for wired networks and fixed hosts, provided that routing support for MHs is available (e.g., Mobile IP). This approach, however, hides mobility from transport and higher layers and does not take into account essential features of mobile wireless systems, e.g., low bandwidth and error prone wireless links, resource-limited MHs, and so on. An alternative approach is using an indirect
protocol, 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. It has been shown that indirect protocols may offer better performance and more flexibility in mobile wireless environments [BB97]. In this paper we are concerned with indirect protocols.

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. This MSS will send proper messages to the other MSSs that, in turn, will send messages to the MHs in the respective cells. The protocol-dependent details include, in particular, those related to the detection of and recovery from missing multicasts (a MH could miss a multicast due to unfortunate movements, even with reliable communication links [BB97]).

To our knowledge, the first (indirect) reliable multicast protocol for distributed mobile systems was proposed by Acharya and Badrinath [AB96]. In this protocol, hereinafter AB-protocol, each MSS maintains, for each MH in its cell, an array of sequence numbers describing the multicasts already delivered by that MH. The MSS uses this array to forward pending messages in sequence and without duplicates. If the MH switches cell, the array is moved to the new MSSs by means of a proper hand-off procedure. Most reliable multicast protocols proposed later are largely based on the AB-protocol [PRS97,ARR97,AV97,YHH97]. They attempt to optimize this protocol and they are all based on the notion of hand-off.

We have proposed a reliable multicast protocol that is a considerable departure from the AB-protocol and, in particular, it is not based on hand-off [B98,ABS99]. In our protocol, MHs are responsible for discarding duplicates, buffering out-of-order messages, requesting retransmission of lost messages. While hand-off requires messages in the wired network upon each cell switching, i.e., even when no new multicast is generated, in our protocol there are no data structures that need to travel across the wired network upon movements of MHs. This feature allows supporting efficiently a large number of MHs that frequently switch between cells, which is highly desirable due to the current trend toward smaller cells motivated by their advantages in terms of improved aggregate throughput and smaller power required for transmitting [CP98]. Absence of hand-off also enabled us to keep the protocol simple and to support a general system model. For example, our protocol supports incomplete spatial coverage of wireless cells and unreliable communication within each cell (more details and additional features can be found in [B98,ABS99,ABS99a]).

We analyzed the performance of our protocol through simulation and found very good results [ABS99a]. We were not able to compare our results to the AB-protocol as we are not aware of any performance analysis for it. In this paper we compare the two protocols through simulation and use this comparison for gaining new insights into the design of reliable multicast protocols for mobile wireless systems.

We anticipate that our protocol outperforms the AB-protocol in several aspects: latency, scalability, bandwidth usage efficiency, quickness in managing cell switches of MHs. Our results show that, when designing algorithms for distributed mobile systems, the shift of computation and storage costs from MHs to SHs [BAI94] should be done judiciously: we retained more such costs on MHs than the AB-protocol, but this design choice allowed us to obtain a more efficient protocol. Moreover, since the two protocols may exhibit similar performance only when the number of MHs is close to the number of cells, it appears that hand-off may be a suitable structuring for reliable point-to-point communication [BB97] but probably it is not so for reliable multicast, in particular, for local environments populated by a large number of mobile hosts. Notice that the AB-protocol expected that the number of MHs is much greater than the number of cells. A discussion of the lessons learned is given at the end of the paper.

2 Overview of the two protocols

In this section we provide a brief and high-level overview of the two protocols. Full details can be
found in the cited papers. For ease of presentation we shall assume that all group members run on MHs and that there is one group member at each MH. 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. Both protocols require reliable FIFO-ordered multicast in the wired network.

Our protocol is based on a system model in which wireless cells provide only incomplete spatial coverage and such that message losses might occur even within cells, for example due to physical obstructions or to the high error rate typical of wireless technology. The description below considers these two aspects because they cannot be factored out, but simulations assumed complete spatial coverage and reliable wireless communication for homogeneity with the AB-protocol. The description omits features not provided by the AB-protocol, i.e., dynamic membership and delivery ordering guarantees stronger than FIFO [ABS99,ABS99a].

2.1 The AB-protocol

A MH wishing to send a multicast sends the payload to the local MSS, say MSS_i, MSS_n, that we call the initiator of m, constructs a message m containing the payload, a locally-generated sequence number, and the set of intended recipients dest(m). Then MSS_i multicasts m to all MSSs. Upon receiving m, each MSS associates it with an empty set ack(m) and stores the resulting item in a local FIFO-ordered queue called M_buffer.

Consider a message m in M_buffer at a MSS. When the cell of MSS contains a MH_j in dest(m), MSS performs the following steps: (i) send m to MH_j; (ii) wait for an acknowledgment; (iii) remove MH_j from dest(m); and (iv) insert MH_j into ack(m). If MH_j already delivered m, then MSS performs only step (iii). MSS realizes whether MH_j has delivered m or not by the recd array below.

When there are no MHs in the cell that are members of dest(m), MSS sends ack(m) to the initiator of m. This MSS removes the members of the received ack(m) from the local copy of dest(m) and multicasts a delete(m) to all MSSs when dest(m) has become empty. Upon receiving delete(m), each MSS removes the entry for m from the local M_buffer.

Each MH is associated with an array, called recd, with one element for each MSS. The element associated with MSS_i contains a sequence number seq_i with this interpretation: MH has delivered all messages originated by MSS_i and carrying a sequence number seq <= seq_i. This array is stored at the MSS of the cell where MH happens to be located and is updated by this MSS as necessary. When MH switches cell, recd moves from the old MSS to the new one, as part of the hand-off procedure.

2.2 Our protocol

A statically defined SH acts as coordinator, denoted C. A MH wishing to send a multicast sends a NEW message containing the payload to C, through the MSS of the cell where MH happens to be located (we indicate the "tag" of each message in SMALLCAPS). When a MSS receives a message m from a MH, it responds with an acknowledgment and forwards m to C. MH periodically re-sends a copy of m until receiving a matching acknowledgment. Retransmissions and acknowledgments are mandatory in our system model, due to the combination of incomplete spatial coverage, unreliable communication within cells, unpredictable movements of MH. Notice that MH could send m while in a cell, leave the cell before receiving the acknowledgment, re-send m in a new cell and so on, until receiving the expected acknowledgment from a MSS.

Each MH maintains a sequence number that encloses in each NEW message. This sequence number enables C to discard duplicates and to process NEW messages from a given MH in the order in which they were generated rather than in the order in which they are received (duplicates and out-of-order messages may result from retransmissions and MH’s unpredictable movements). Upon

---

1 Actually, the full version of our protocol is based on a set of coordinators. The size of this set may be tuned for improving performance with incomplete spatial coverage [ABS99,ABS99a]. Here we used only one coordinator because we assumed complete spatial coverage, for uniformity with the AB-protocol. Using a single coordinator is a worst-case scenario, as will be clear from the following protocol description.
receiving a NEW message, C changes the tag to NORMAL, appends the fields described below and multicasts the resulting message to MSSs. MSSs then broadcast NORMAL messages in the respective cells. C maintains a sequence number incremented whenever it constructs a NORMAL message. A NORMAL message contains the current sequence number (cseq) and the sequence number (cseq-mh) that was current when constructing the last NORMAL message on behalf of the same sending MH.

A MH receiving a NORMAL message m determines whether: (i) m has to be discarded (i.e., it is a duplicate); (ii) m has to be delivered; (iii) m has to be buffered as its immediate delivery would violate the FIFO ordering. The three cases may result from the combination of incomplete coverage, unreliable wireless links, MH’s unpredictable movements. MH discriminates among them by comparing the sequence number cseq-mh enclosed in the message just received against the sequence number of the last NORMAL message received. We omit the details for brevity.

When MH receives an out-of-order message, it sends to the MSS a NACK specifying the missing (consecutive) sequence numbers. The same retransmission mechanism of NEW messages ensures that a NACK arrives at a MSS. A MSS that receives a NACK relays to the sending MH a copy of the missing multicasts, through a sequence of TRANSFER messages. TRANSFER messages are equivalent to NORMAL ones but are destined to a specific MH. MSS aborts the sequence if the destination MH leaves the cell.

Each MSS maintains locally a cache of previous multicasts. A MSS fetches from C messages needed for a TRANSFER sequence that are not in the local cache. C stores a copy of each NORMAL message that has not been acknowledged by all recipients. MHs enclose acknowledgment information in NACK messages. Such information is extracted by MSSs and propagated to C as necessary.

2.3 Observations

The AB-protocol stores state information at MSSs. Part of such information travels across the wired network upon each cell switching (e.g., which messages have been delivered by the moving MH). Other information is static and resides at a location (the initiator) that may be different for different messages (e.g., which MHs are not known yet to have delivered a given message). Special emphasis is placed in preventing the transmission of duplicates along the wireless links and for ensuring that a MH never receives out-of-order messages. A different copy of each message is sent to each MH. A MH has to acknowledge each received multicast individually and as soon as it receives it.

Our protocol stores state information at the coordinator C (past multicasts, which MHs have not yet acknowledged a given multicast) and caches some information at MSSs (copies of past multicasts). MHs may receive duplicates and they are responsible for discarding them. Multicasts are not acknowledged individually. Delivery information propagates asynchronously with respect to new multicasts. MHs may receive out-of-order messages, hence a message may spent some time in a buffer at the MH before being delivered. A MSS broadcasts new multicasts in the cell rather than directing them to a specific MH.

3 Simulation Environment

To compare the performance of the two protocols we used discrete event simulation. We considered an operating environment similar to a small campus or building. We assumed a wired bandwidth of 10 Mbps with propagation delay uniformly distributed within the range [0.5 - 2.5] msec. We assumed a wireless bandwidth of 1 Mbps, in line with the bandwidth available in current Wireless LANs [AL99]. Wireless propagation delays are negligible as cells were supposed to be very small, (e.g., ten meters) and the coverage is complete. MHs generate 512-byte messages according to a Poisson process, i.e., times between the generation of successive messages are

---

2 Results have been estimated by using the independent replication method and assuming a confidence level of 90%.
random variables exponentially distributed. A MH 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 ($T_{cell}$).

In order to make the comparison meaningful we exercised the two protocols in the same conditions. Hence, we assumed that (i) wireless communications inside cells are reliable (though our protocol does non rely on such assumption); (ii) all messages require Fifo ordering of delivery (though our protocol also supports Causal and Total orderings).

4 Results

Unless stated otherwise, the following values have been used in the simulation experiments for the set of parameters common to both protocols. There are 40 MSSs ($N_g=40$) and 50 MHs ($N_r=50$). All MHs receive multicasts whereas a subset of them with $N_s$ members may generate messages. Each sender generates, on the average, 8 messages/sec corresponding to a bit rate of approximately 33 Kbps. The average cell permanency time for each MH is 5 seconds.

4.1 Scalability with the number of receivers

A key performance index is the average message latency, defined as the average time elapsed from the instant at which a message is generated at a sending MH to the instant at which the same message is delivered by a destination MH. In this section we discuss the average message latency as a function of the number of receiving MHs, $N_r$.

Figure 2-left shows that, with our protocol, the average latency is almost independent of $N_r$. In contrast, the AB-protocol scales poorly: (i) the latency increases noticeably with $N_r$; and (ii) the slope of the curves increases quickly with the number of senders $N_s$.

![Figure 2: Average latency as a function of the number of MHs $N_r$ (left) and the $N_r/N_g$ ratio (right).](image)

The AB-protocol performs slightly better than our protocol only when the aggregate message rate is low (e.g., $N_s=10$ corresponds to less than 33% of the wireless link’s capacity) and the number of receivers is very small. This result is highlighted, for the case $N_s=1$, by Figure 2-right: the AB-protocol exhibits a slightly smaller latency when the $N_r/N_g$ ratio ($N_g$ being the number of MSSs) is less than 1. Note than in [AB96] it is expected $N_r>N_g$ in a realistic scenario.

The reason is that in our protocol each message is sent to the coordinator before being broadcast to all MSSs. Hence, the message latency includes a component (related to the MSS-to-coordinator transmission) not present in the AB-protocol. However, when the aggregate message rate and/or the number of receivers increases this component is overwhelmed by other delay components, in particular queuing delays experienced at MSSs (see section 5).

Figure 2-right also shows that the performance of both protocols do not depend on the number of cells $N_g$ but only on the ratio $N_r/N_g$. 


4.2 Scalability with the number of senders

Figure 3-left shows the average latency as a function of the number of senders $N_s$ (and, hence, of the aggregate message rate) for different $N_r$ values. In each experiment the number of receiving MHs $N_r$ is constant while the number of sending MHs $N_s$ varies between 1 and a maximum value less than or equal to $N_r$. For our protocol a single curve (for $N_r=50$) is reported since the average latency does not depend on the $N_r$ value (as shown is Section 4.1).

In our protocol the average latency increases slightly with the number of senders. In contrast, the increase is remarkable and highly influenced by the number of receivers $N_r$ with the AB-protocol.

Notice that in our protocol the average latency remains at acceptable values (tens of msec) even when $N_s=26$, which corresponds to an aggregate message rate greater than 80% of the wireless network capacity. On the other hand, the AB-protocol exhibits an average latency greater than 100 msec (with a small number of receivers, $N_r=20$) though the aggregate message rate is still under 50% of the wireless capacity. This means that in our protocol the limiting factor is the wireless bandwidth while in the AB-protocol the limiting factor is the protocol itself.

We analyzed the delay components of both protocols in order to identify their potential bottlenecks. We omit the details for space reasons and report only the main highlights. We found that, in both cases, the main bottleneck is the queuing delay at MSSs, as this component tends to become predominant when the aggregate bit rate grows. We found also that in our protocol the delay experienced at the coordinator remains negligible, even at high message rates. This means that: (i) the only cost of the intermediate step at the coordinator is the forwarding delay, from the MSS to the coordinator; (ii) this additional delay tends to be overwhelmed by queuing delays at MSSs. It follows that our simple coordinator-based structuring is indeed practical (section 5).

4.3 Effects of mobility

An important issue to take into account when dealing with protocols for distributed mobile systems is the influence of mobility on the protocol behavior. To investigate this influence we considered the following indices

- **average message latency**;
- **percentage of lost messages**, i.e., the percentage of messages lost due to mobility (see below) to the total number of delivered messages. Since lost messages have to be retransmitted, this index gives a measure of the additional consumption of wireless bandwidth introduced by mobility.
- **average response delay**, i.e., the time interval between a cell switch of a MH and the delivery of the next message at this MH. It is an indication of the **quickness** of the protocol in managing cell switches of MHs.
Figure 3-right shows the average latency as a function of mobility (the number of cell switches per second per MH is the inverse of the average cell permanency time defined in Section 3). It can be seen that both protocols perform well in the sense that latency is almost unaffected by mobility.

![Graph showing average latency and lost messages as a function of mobility.]

The apparently strange behavior of the curve related to the AB-protocol with $N_s=10$ can be explained as follows. When the aggregate message rate is high there might be an occasional concentration of MHs in a cell so that the bandwidth necessary to serve them all is greater than the available wireless bandwidth. In such a situation the message latency experienced in that cell tends to be unbounded, which greatly increases the overall average latency. Each such concentration lasts for an amount of time that depends on mobility: the higher the mobility, the earlier the concentration disappears. The contribution of such unfortunate concentrations to the average latency thus decreases when mobility increases.

Obviously, occasional concentrations of MHs in a cell can also occur when using our protocol. However, they have no influence on the average latency because a single broadcast transmission is done by MSSs, irrespective of the number of MHs in the cell.

Before presenting our results about the percentage of lost messages, it is useful to analyze message loss in more detail. Consider a message $m$ sent by a MSS to a MH (similar consideration apply to messages sent on the reverse path). In the AB-protocol $m$ is lost if either of the following occurs: (a) MH moves to a new cell while $m$ is being transmitted; or (b) $m$ is passed to lower layer protocols for transmission to MH, but is actually transmitted after MH has left the cell. In our protocol $m$ is lost whenever MH leaves a cell before receiving $m$ and moves to a new cell where $m$ has been already broadcast. In other words, as anticipated in Section 2, mobility of MHs makes it possible the loss of messages even with reliable communications.

In our protocol, mobility makes it possible the reception of duplicate messages at MHs. However, we did not take duplicates into account because: (i) their number is usually very small (in the order of 1% [ABS99a]); and (ii) unlike lost messages, duplicates do not necessarily consume additional wireless bandwidth since, in general, the message could not be a duplicate for some other MH in the same cell.

Figure 4-left shows the percentage of lost messages as a function of mobility. In the AB-protocol this percentage is higher than in our protocol. This result is the consequence of sending a message copy per MH in a cell, as follows. Message copies are passed down to lower layer protocols for transmission to MH and transmitted in sequence. Even with a few MHs in the cell (e.g., 2 or 3) the interval from the time instant at which a message copy is passed down to lower layer protocols to the time instant at which the same copy is completely transmitted may be large. Accordingly, the probability that the MH leaves the cell during this time interval, thus missing the message, may be significant. The reason why the percentage of lost messages decreases when the number of senders...
The average response delay is analyzed as a function of the beacon period $T_b$ (the beacon is the signal periodically broadcast by each MSS to announce its presence to MHs). To understand why $T_b$ plays a key role in the average response delay, consider a MH that has just entered the cell of a MSS. The AB-protocol may start delivering messages to MH only after MSS has completed the hand-off, which begins when MH receives the first beacon from MSS. Our protocol does not use hand-off but piggybacks in beacons the sequence number of the last message broadcast in the cell, which enables MH to discover more promptly whether it missed messages during the cell switch (this feature is not necessary for correctness but is an optimization useful when the message rate is low).

Figure 4-right shows the average response delay as a function of the beacon period $T_b$. It can be seen that $T_b$ has a key impact on the AB-protocol, whereas its influence in our protocol is much smaller and can be noticed only at low message rate. Most importantly, the figure also shows that that our protocol is more responsive than the AB-protocol, except when the beacon period is very small (less than 30 msec in the above experiments). However, such values for $T_b$ do not appear realistic since they cause a significant amount of wireless bandwidth to be wasted by beacons themselves.

For a given $T_b$, our protocol tends to be more responsive when the aggregate message rate grows because messages missed because of a cell switch are detected earlier, whereas the average response delay of the AB-protocol is not significantly influenced by the message rate.

### 4.4 Other cost factors

#### Wireless bandwidth consumption

An important cost factor to consider is the consumption of wireless bandwidth, because this is a limited resource. To this end, we considered the consumption factor ($f_wl$), i.e., the total number of bits transmitted in the wireless network to multicast one payload bit, normalized to the number of MHs. We estimated this factor through simulation and, in addition, we derived analytical relations for it. Such relations are approximated and, in particular, they neglect message losses caused by mobility. Nevertheless, they are useful because they describe the behavior of the two protocols in a synthetic but sufficiently accurate manner.

Figure 5-left reports both analytical predictions (see below) and simulation estimates of the wireless consumption factor $f_wl$ for different numbers of receiving MHs $N_r$. In the AB-protocol $f_wl$ is constant and slightly greater than 1, i.e., each payload bit destined to a specific MH requires the transmission along the wireless links of slightly more than 1 bit. In our protocol the consumption factor is approximately the same as in the AB protocol when there are few receivers, but it becomes significantly smaller than 1 as the number of receivers grows up.

The analytical predictions result from the following considerations. The consumption factor is given by the total number of bits transmitted on the wireless network to multicast a message, normalized to the number of MHs ($N_r$) and message size ($S_{msg}$). Let $S_{ack}$ be the size of an acknowledgment message. With the AB-protocol, the following wireless transmissions occur:

1. message transmission from the sending MH to its local MSS ($S_{msg}$ bits);
2. message transmission from the MSSs to the other MHs ($(N_r-1)\cdot S_{msg}$ bits);
3. ack transmission from each receiving MHs to its MSS ($(N_r-1)\cdot S_{ack}$ bits)

When using our protocol the message is transmitted by the MH ($S_{msg}$ bits) and acknowledged by the related MSS ($S_{ack}$ bits). The message is then broadcast in all non-empty cells whose number is equal to $\min(N_g, N_r)$, if we assume that MHs are uniformly distributed between cells ($S_{msg} \cdot \min(N_g, N_r)$ bits).

Hence, it is
By considering that acknowledgments are usually much smaller than the message size (e.g., $S_{ack} = 0.1 \times S_{msg}$) and that it is reasonable to expect $N_r >> 1$ and $N_g >> 1$, equation (1) can be approximated as

$$f_{wl} = \begin{cases} \frac{S_{msg} + (N_r - 1) \cdot S_{msg} + (N_r - 1) \cdot S_{ack}}{N_r \cdot S_{msg}} = 1 + \frac{(N_r - 1) \cdot S_{ack}}{N_r \cdot S_{msg}} & \text{AB protocol} \\ \frac{S_{msg} + S_{ack} + \min(N_g, N_r) \cdot S_{msg}}{N_r \cdot S_{msg}} = 1 + \frac{S_{ack}}{S_{msg} + \min(N_g, N_r)} & \text{Our protocol} \end{cases}$$

Figure 5-left confirms that mobility, neglected in the analytical predictions, introduces an additional component in the consumption of wireless bandwidth. Notice, however, that when the number of MHS falls in the range between 30 and 70 our protocol has a consumption factor less than the analytical predictions provided by equation (1). This behavior can be explained by recalling that in deriving equation (1) for our protocol it was assumed that MHS are uniformly distributed between cells and, hence, the number of non-empty cells is given by $\min(N_g, N_r)$. However, when the number of MHS is similar to the number of cells (40 in our experiments, see Section 4), there may be a considerable fraction of time in which the number of non-empty cells may be even less than $\min(N_g, N_r)$ due to mobility.

Figure 5-right shows the percentage of wired bandwidth consumed by each protocol as a function of the number of MHS. In general, the AB-protocol is more consuming than our protocol especially for large numbers of MHS, $N_r$, and high $N_l$ values. Notice that in our protocol the consumption of wired bandwidth remains approximately constant when the number of MHS increases.

**Wired bandwidth consumption**

Another important factor to take into account is the overhead introduced in the wired network, i.e., the amount of wired bandwidth consumed by protocols to accomplish their task. This overhead includes two components: (i) delivery of messages to MHS; and (ii) mobility management. In the AB-protocol the handoff procedure is responsible for the latter component: it is greater than zero even when no message is multicast. Of course, this component is not present in our protocol where handoff is not used.

Figure 5-right shows the percentage of wired bandwidth consumed by each protocol as a function of the number of MHS. In general, the AB-protocol is more consuming than our protocol especially for large numbers of MHS, $N_r$, and high $N_l$ values. Notice that in our protocol the consumption of wired bandwidth remains approximately constant when the number of MHS increases.
Memory Occupancy at MHs

Since our protocol does not use hand-off, at first glance MHs have to store more data structures than in the AB-protocol. For example, they have to remember the sequence number of the last delivered message (for correctness) and they have to buffer out-of-order messages that cannot be delivered yet (for performance). However, data structures at MHs are necessary also with the AB-protocol. Although this protocol assumes that wireless communication is reliable, messages could nevertheless be lost due to mobility. A MH becomes aware of a message loss after a cell switch, as part of the handoff procedure. It follows that a MH has to buffer all messages sent in a cell, until the next handoff.

To compare the memory requirements of both protocols we measured, by simulation, the memory occupancy at MHs. For the AB-protocol, we estimated the minimum buffer size necessary to cope with 99.9% of lost messages. For our protocol, we measured the minimum buffer size necessary for storing the 99.9% percentile of the out-of-order messages (we also considered the 1-message buffer used to store the last transmitted message until the related acknowledgement is received). We considered only these buffers as the size of the other data structures is very small. We simulated the usual scenario with $N_s=10$ and the final result was that the memory occupancy is approximately the same in both protocols, roughly 1 KB.

5 Concluding remarks

In this paper we have compared two protocols for reliable multicast communication in mobile wireless systems. These protocols are very different in their design philosophy. The AB-protocol applies a principle suggested in [BAI94] to its maximum extent, i.e., shifting computation and communication costs from MHs to MSSs. This shift is applied by means of data structures associated with each MH, that are stored at a MSS and that are moved to another MSS upon each cell switch of the MH. Furthermore, the AB-protocol insists in preventing the transmission of duplicates at each MH. Our protocol, in contrast, shifts very little from MHs to MSSs. The responsibility for detecting duplicates and out-of-ordering messages is retained at MHs. Essentially, stationary hosts are responsible only for storing past messages that could still be needed by some MHs.

We have compared the above protocols both in terms of performance and costs and we have found that our protocol outperforms the AB-protocol in several aspects: latency, scalability, bandwidth usage efficiency, quickness in managing cell switches of MHs. The only scenario in which the two protocols exhibit similar performance is when there is approximately only one MH per cell and, in addition, the message generation rate is moderate.

In our opinion, the key rationale for this result is that our protocol allows exploiting better the broadcast nature of the wireless medium. Whereas in our protocol MSSs "blindly" broadcast new multicasts in the respective cell, in the AB-protocol MSSs maintain individual connections to each MH. This feature, coupled with the acknowledgment that in the AB-protocol a MH has to send upon receiving each message, explains the substantial differences in scalability and bandwidth usage efficiency observed. The structuring of the AB-protocol also implies longer queuing delays at MSSs, which in turn negatively affect latency and quickness in managing cell switches.

We have learned several lessons. First, shifting responsibilities from MHs to MSSs should be done judiciously. One has to balance simple and cheap algorithms at MHs, ability to exploit efficiently the wireless medium, short queuing delays at MSSs. The proper trade-off among these factors is not necessarily achieved by privileging the first of them. Second, handoff may be a suitable structuring for reliable point-to-point communication (e.g., indirect TCP) but it does not appear to be so for

---

3 Our protocol would work even with no such buffer, i.e., by discarding out-of-order received messages: one could trade resources at MHs against performance [ABS99,ABS99a]. However, a fair comparison of memory occupancy has to assume a maximally-sized buffer.
reliable multicast communication, in particular, for local environments populated by a large number of mobile hosts. Third, queuing and transmission delays at MSSs give the main contribution to the overall message latency especially when the aggregate message rate goes up. Due to the mismatch between the wired and wireless bandwidth, this feature is likely to apply in any reliable multicast protocol for mobile wireless systems. It follows that, for such protocols, MSSs are likely to be the main bottleneck and that an additional hop on the wired network has a cost almost negligible.

Fourth, being able to afford an additional hop makes it practical a simple coordinator-based architecture, in which MSSs merely cache state information whose primary copy is stored at another stationary host (the coordinator). This feature may have several implications. For example, that enforcing totally ordered multicast delivery rather than FIFO (per-source) ordering has little additional cost [B98]. Furthermore, that extending our protocol toward tolerating crashes of stationary hosts is not difficult (we are working on it). Essentially, the crash of a MSS only affects the extension of the coverage region. It does not affect correctness because critical data structures are not stored at MSSs (unlike the AB-protocol and all protocols based on hand-off). Crashes of coordinators can be tolerated with established techniques for fault-tolerance, e.g., replicated state machine or primary-backup based on group communication [B98,ABS99].

References


