MaGMA: Mobility and Group Management Architecture for Real-Time Collaborative Applications*

Nadav Lavi   Israel Cidon   Idit Keidar

Department of Electrical Engineering, The Technion, Haifa, 32000, Israel

Abstract

We introduce MaGMA, a Mobility and Group Management Architecture, enabling real-time collaborative group applications such as push-to-talk (PTT) for mobile users. MaGMA provides, for the first time, a comprehensive and scalable solution for group management, seamless mobility, and quality-of-service (QoS). MaGMA is a distributed IP-based architecture consisting of an overlay server network deployed as part of the service infrastructure. MaGMA's architecture consists of a collection of mobile group managers (MGMs), which manage group membership and may also implement a multicast overlay for data delivery. The architecture is very flexible, and can co-exist with current as well as emerging wireless network technologies. We see such services as essential components in *beyond-3G (B3G)* networks. We propose two group management approaches in the context of MaGMA. We devise protocols for both approaches, evaluate both solutions using simulations, and validate the results through mathematical analysis. Finally, we present a proof-of-concept prototype implementation.

Key Words: Mobility, Group Communication, Push-to-Talk.

*Preliminary version of this paper appears in Proceedings of the Sixth IEEE IFIP Conference on Mobile and Wireless Communication Networks (MWCN), Paris, October 2004. This work is partially supported by the Intel Research Council Wireless and Computing Group and by the Israeli Science Foundation.

1 Introduction

The widespread availability of the Internet has enabled the use of many groupware and collaborative computing applications (chat, ICQ, NetMeeting, Exchange, Lotus Notes, Webex, desktop video conferencing, etc.). With the advance of wireless personal communication, such groupware applications are becoming popular in cellular and mobile networks [27]. For example, major cellular providers (Verizon, Nextel, Orange) offer, or plan to offer soon, group services such as push-to-talk (PTT) [16, 30]. The PTT cellular revenue, which was $84 million in 2003, is expected to reach $10.1 billion by 2008; and the 2.3 million PTT cellular subscribers community of 2003 is expected to grow to 340 million by 2008 [32]. While traditional PTT is limited to voice, the emerging convergence expected in *beyond-3G (B3G)* will merge real-time and non-real time aspects of group communication.

There is strong evidence that future wireless network infrastructure will conform to the TCP/IP architecture and its related supporting mechanisms for real time applications (VoIP, VCoIP), QoS, and mobility. TCP/IP is rapidly being adopted by emerging standards for cellular networks [1, 2, 19], not only at the transport layer, but also at higher level standards such as the session initiation protocol (SIP). This trend enables the convergence of cellular networks with the global Internet [8]. At the same time, low-cost and high-speed wireless access to IP networks is becoming widely available via WiFi and WiMAX access points. Freed from the wire constrains, Internet endpoint devices are becoming smaller, lighter, and easier to operate under mobility conditions. These two parallel trends are leading to gradual convergence between the previously separate worlds of cellular and wireless IP, both at the mobile device level and at the network infrastructure. Given the importance of groupware, the converged wireless network should support cross-network group services for both real-time and data communications.

Consequently, a clear missing link in this evolution is the lack of comprehensive support for group management as well as adequate solutions for real-time applications (QoS, seamless handoff, etc.) in the IP mobility standards. The emerging real-time groupware applications need a solid and integrated framework on which mobile users can be supported.

We present MaGMA, a novel architecture for group management in mobile networks interconnected via the global Internet. MaGMA provides a comprehensive solution for the mobile world, addressing aspects such as scalable group management, mobility, handoff, and QoS provision. The main goals of MaGMA are mapping group names to their current subscribers, supporting mobility with seamless handoffs, QoS support for RT applications, transport efficiency including the avoidance of triangle routing and a scalable control plane. We believe that such a solution will be an inherent part of 3G and B3G cores. We are not aware of any previous comparable solution for mobility support in groupware applications.

MaGMA's architecture consists of a collection of mobile group managers (MGMs), which manage group membership and may also implement a multicast overlay for data delivery. Each mobile node (MN) is served by an MGM proximate to it. Developing a fully distributed group management and mobility solution that is both efficient and scalable is technically challenging. In this paper, we address this challenge. We offer two group management approaches in the context of MaGMA: The first is a *subscription model*, in which the servers (MGMs) support group management only, and MNs can obtain the list of group subscribers in order to transmit data to them without the servers' intervention. The second is the *multicast overlay model*, where the servers implement the group multicast service. The first approach is appropriate for small groups and light servers, whereas the latter is more appropriate for large groups and clients with battery power constrains. We devise protocols for both approaches, and evaluate both solutions using the ns2 network simulator [18]. We validate our simulation results through mathematical analysis. As a proof-of-concept, we also build a SIP-based prototype running groupware applications over WiFi network with iPAQ MNs. In [11], we provide a formal correctness proof of our protocol for the multicast overlay model.

This paper proceeds as follows: Section 2 discusses previous mobility and group management solutions. Section 3 describes the network model and the MaGMA architecture. Section 4 presents our subscription model solution, and Section 5 presents the control plane of MaGMA's multicast overlay solution. In Section 6, we evaluate MaGMA's control plane via simulations and analysis; the detailed analyses appear in [11]. Section 7 presents MaGMA's data plane and respective simulation results. Section 8 describes a prototype implementation of MaGMA. Section 9 discusses ideas for future work, and Section 10 concludes.

2 Related Work

Mobile IP [21] is the current mobility standard for IPv4. Every MN is associated with a home domain and a server in that domain that acts as its *home agent* (HA). All traffic to an MN is routed via its HA. This *triangle routing* scheme often leads to inefficient routes, and creates a strong dependence of the MN on its home. Although IPv6 [10, 22] can provide better support for communication with MNs, it is not widely deployed.

Several solutions, e.g. [23, 31], eliminate triangle routing by sending binding information when an MN moves. It is unclear whether such solutions can support simultaneous movements of both endpoints of a communication. In addition, establishing new connections to an MN always involves its home domain, even if the MN is distant from it for an extensive period.

Balakrishnan and Snoeren [24] propose a DNS-based solution to IP mobility. In order to avoid the use of stale binding information, DNS caching is minimized (by setting TTL=0). The major drawbacks of this approach are that both endpoints cannot move simultaneously, that DNS standards do not support the proposed user self-configuration, and that operating systems and DNS servers often do not comply with DNS TTL caching directions. Finally, eliminating DNS caching is bound to overload the DNS system.

None of these solutions explicitly supports group communication. On the other hand, current group management protocols, e.g., [3], were not designed with mobility in mind and do not incorporate a handoff solution.

Several solutions [14, 20, 6, 26], suggest the use of the IP multicast infrastructure with host mobility. While these solutions can potentially provide good performance, unfortunately, IP multicast is not widely deployed.

Current industrial solutions for PTT in cellular networks [17, 13, 28] implement the OMA-PoC [19] standard, which uses a centralized server. This solution, although used by various cellular operators, suffers from lack of scalability and excessive end-to-end data delay since it routes all data through the centralized server. An alternative approach for supporting instant messaging and chat in the wireless world is proposed by Jabber [9]; it uses a distributed architecture of servers with e-mail-like addressing. Due to the use of such addressing, data is always sent through the home servers of both the sender and the recipient, which is not the optimal route at times of mobility, and can degrade the performance of real-time applications.

3 Network Model and Architecture

3.1 Network model

We model the network as a collection of autonomous domains. Every MN has a unique ID (UID), which identifies the MN at all locations. Upon moving to a new domain, the MN obtains a new local IP address (e.g., using DHCP). We assume that a micro-mobility mechanism is in place in each domain (whether cellular or WiFi), and that an adequate IP routing protocol exists in each domain.

We assume that message propagation time is bounded by some Δ time units and that all the network channels support reliable FIFO communication. We further assume that the network supports smooth handoff, and that the handoff delay is negligible, that is, an MN detects that it enters a new domain immediately. When an MN moves to a new domain, it does not move again to yet another domain for at least 10Δ time units. This movement bound is more than reasonable, for example if $\Delta = 100msec$ (a reasonable end-to-end delay in a network that support RT communication [7]), we expect the MN to stay in a domain for at least a second. In addition, we assume that MGMs do not crash and that MNs can crash, and such crashes are detectable: when an MN crashes, its local MGM detects the crash by $2\Delta < \tau < 3\Delta$ time units.

3.2 The Architecture

MaGMA consists of MGMs distributed throughout the network. Ideally, an MGM is located in each domain. For the sake of simplicity, we assume that the MGMs are static and well-known. The MGMs form an overlay network among them. We propose two approaches to support groupware: in the subscription model, the MGMs are only in charge of group management, whereas in the multicast overlay model, they are in charge of both group management and data delivery over the overlay network. The overlay construction can employ known techniques for building efficient QoS-aware overlays, e.g., [25, 5], and is beyond the scope of this paper. In the multicast overlay model, MGMs distribute traffic within their domains to the appropriate MNs. To this end, they can use either unicast or multicast– we recommend the use of multicast where available as it is more efficient.

MaGMA can be used in several ways to allow interoperability among WiFi and cellular users. For example, the cellular provider can implement a single MGM at the cellular core managing the entire cellular network, or a distributed MGM system where each MGM manages a domain of several base stations. Other MGMs can be located at WiFi/WiMAX domains. An example of such an integrated architecture is depicted in Figure 1.

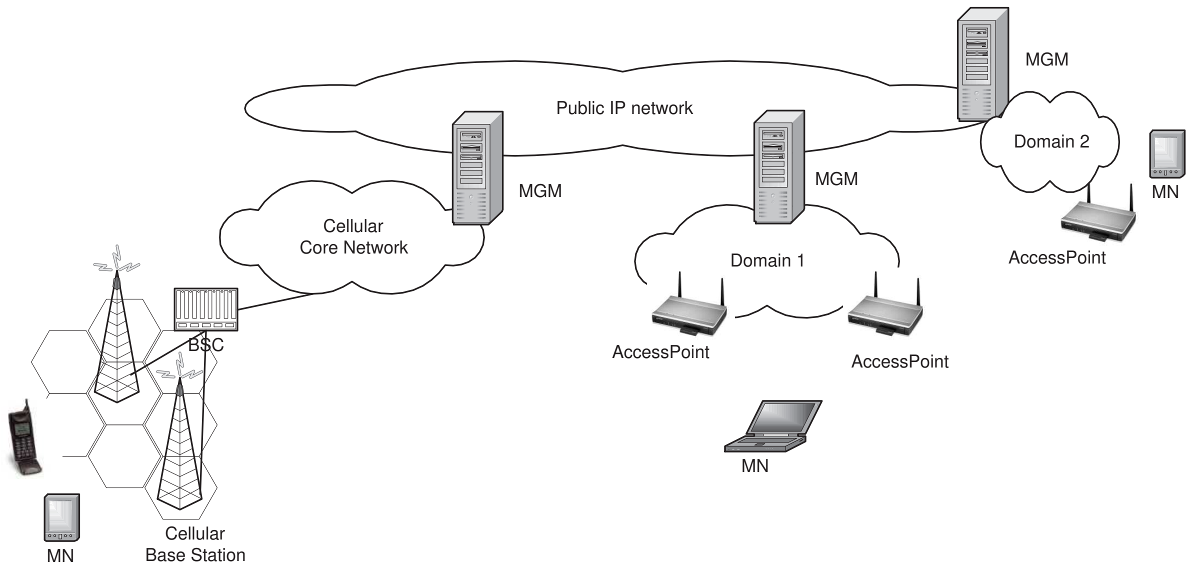


Figure 1: An example integrated network architecture.

MaGMA calls for the use of distributed servers for the following reasons:

- to offer scalability in the number of groups and the number of group members;
- to efficiently support groups with geographically dispersed members;
- to facilitate QoS reservation among domains;
- to reduce traffic overhead; and
- to enable vertical handoff between networks.

Each MN is served by the MGM closest to its domain.

The MGMs provide to the MNs group services such as *joining* and *leaving* a group, and they enable MNs' movement between domains. In the subscription model, MNs can retrieve group views and receive continuous reports of membership changes from the MGMs. In the multicast overlay model, MGMs provide multicast and data delivery services. In addition, MGMs may provide advanced application support services, which are left for future research.

We assume that MNs are likely to remain in groups for considerable periods. Therefore, we assume that move operations dominate the control traffic.

4 MaGMA Subscription Model Solutions

In the subscription model, MGMs manage MN-level group membership. That is, they keep track of which MNs are in the group at any given time. We present two solutions for the subscription model. We begin, in Section 4.1, by presenting a simple solution in which all MGMs keep track of all the groups, regardless of whether they have group members. Then, in Section 4.2, we present an optimized solution where only MGMs that have group members are involved in managing the group.

4.1 MGMFlood

In our first scheme, MGMFlood, each MGM forwards (floods) to all other MGMs all control messages (*join/leave/move*) received from MNs in its domain. When an MN crashes, its local MGM detects the crash and sends an appropriate *leave* message to all other MGMs.

MGMFlood is simple and allows for seamless handoff due to its prompt reaction to mobility updates. However, it entails high control message overhead, as all MGMs keep views of all groups, including groups not residing in their domains. This solution may be appropriate for a service managing a few large groups, but it does not scale well, especially if there are many small groups and localized memberships.

4.2 MGMLeader

Our second solution reduces the overhead by propagating updates only to those MGMs that have group members in their domains. When an MGM receives an MN's message regarding a group that is represented in its domain, it extracts the MGMs that have members in the group from its local view, and forwards the message only to those MGMs.

If an MGM receives a control message (*join* or *move*) for a group that does not yet exist in its domain, then it needs to discover the group's up-to-date view, and to forward the event to the appropriate MGMs. The challenge is preserving a coherent view at all MGMs in the presence of concurrent operations without inducing excessive overhead.

In order to minimize the control overhead and ensure view consistency, only one of the participating MGMs sends the view to the new MGM. To this end, one MGM is designated as the *coordinator* of the group. Every active group has a coordinator, and a single MGM can be the coordinator of multiple groups. If the coordinator leaves the group (because all the MNs in its domain leave) then a new coordinator is elected, as explained Section 4.2.2 below.

We now proceed to describe our protocol. We present a general description of the protocol without pseudo-code only in order to provide intuition for the issues and solution techniques. In the next chapter, we formally describe an extension of this protocol for the multicast overlay model. In Section 4.2.1, we explain how the coordinator manages the group while there are no coordinator changes. In Section 4.2.2, we discuss how a new coordinator is elected when there is none, and in Section 4.2.3, we describe the coordinator transition process. In Section 6.2 we evaluate the control overhead using both mathematical analysis and simulations.

4.2.1 Normal operation

When a new MGM joins a group due to a *move* event, it extracts the moving MN's former MGM from the *move* message, and sends the event message to that MGM. The former MGM, in turn, forwards the message to the coordinator. When the coordinator receives a *move* message originating from an MGM that is not already in the group, it sends the group's view to the new MGM and forwards the message to all the group's MGMs. This message flow is illustrated in Figure 2. This communication between the two MGMs also facilitates establishing a tunnel from the former MGM to the new one, so that the former MGM can forward data packets destined to the moving MN via its new MGM, to guarantee smooth handoff; such tunneling is suggested in [23].

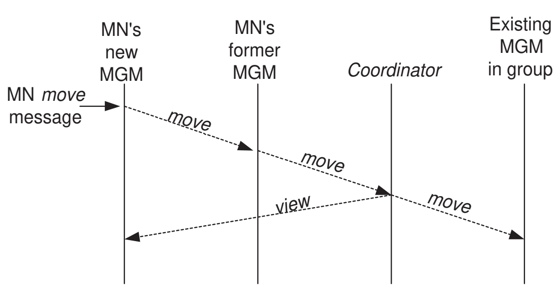


Figure 2: *Move* message flow.

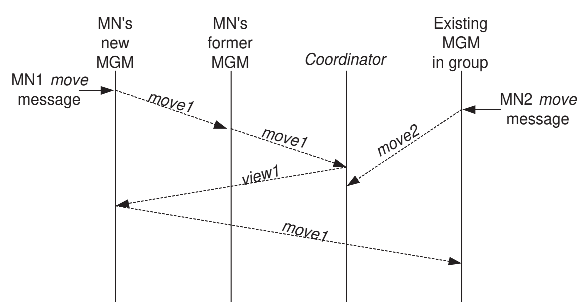


Figure 3: Potential view inconsistency in over-simplified leader-based solution.

When a new MGM joins a group due to a *join* message, it has no knowledge of which MGMs are currently in the group. Therefore, the joining MGMs broadcast a join message to all the MGMs in the system. As before, when the coordinator receives this message from the new

MGM, it sends the group's view to the MGM.

Different MGMs may receive certain event messages in different orders. Thus, there is a need to ensure view consistency when MGMs dynamically join and leave the group. Figure 3 illustrates a problematic scenario that can occur if concurrent joins are handled carelessly. In this example, while a new MGM retrieves the group's view from the coordinator, an existing MGM sends another event to the group's MGMs. The existing MGM is unaware of the new MGM and thus does not forward the message to it. This causes the new MGM to have an inconsistent view of the group.

In order to address this difficulty, each MGM maintains an increasing *Local Event Counter* (LEC) for every group. Whenever an MGM receives a *join*, *leave*, or *move* message from a local MN, it increments the appropriate LEC. The group's LEC is included in every message pertaining to this group sent by the MGM. When an MGM joins a group, it initiates the group's LEC to 1. In addition, the MGM keeps, for every active group, a *LECvector*, holding the highest known LEC for each MGM in this group.

In every message sent from one MGM to another, both the sender's LEC and the receiver's latest known LEC (from the *LECvector*) are included. When an MGM receives a packet, it checks the LECs. If its local LEC is higher than the one known to the sender it sends back its local view and LEC. If it discovers that the sending MGM's LEC is higher than the one it knows, it retrieves the local view of the sending MGM. When the coordinator forwards *move* messages of new MGMs, it includes the LECs corresponding to the view it is sending to the new MGM. In case some events are not reflected in this view, the receiving MGMs forward their local views to the new MGM.

4.2.2 Election procedure

When an MGM first joins a group, it has no information regarding the group members and its coordinator. Therefore, it broadcasts to *all* MGMs a *join* message that includes the group name. In case a coordinator is active, it replies with a *view* message conveying to the joining MGM the necessary group information (the members and the coordinator's UIDs). If a coordinator does not exist, i.e., there are no MGMs participating in this group, then this broadcast initiates a coordinator election procedure.

The election procedure can be initiated by several MGMs that almost simultaneously attempt to join the group. We must ensure that all MGMs participating in the election procedure have the same view of which other MGMs are participating. In order to keep track of the participants, an MGM that sends a *join* message buffers all the *join* messages it receives during a period of 3Δ time units after its own broadcast. This guarantees that MGMs wait long enough in order to get the coordinator's response during a coordinator transition, as explained in Section 4.2.3 below. During this period, new messages received from MNs are also buffered; the MGM processes these messages only after the coordinator election ends. MGMs that do not send a *join* message before receiving another MGM's join cannot join the group for a period of 3Δ time units. During this time, all messages received from MNs are buffered; these messages are processed only after this guard time ends. In order to prevent inconsistencies in the election of a coordinator, the guard timer must be restarted upon reception of every new *join* message.

When the guard time interval ends, MGMs that participate in the election procedure elect an MGM from among the MGMs whose *join* messages have been received to be the group's coordinator. The election can be based on UIDs or any other parameter included in the *join* messages, e.g., which MGM serves the highest number of MNs, or administrative priority. When the guard time expires at an MGM that does not participate in the election, this MGM can join the group (if necessary) by broadcasting its queued *join* message.

We illustrate the need for a correct measurement of the guard-time in Figure 4. In this scenario, MGM3 and MGM2 first initiate the election procedure. MGM1 receives the *join* message of MGM3 and starts the 3Δ guard time. In Figure 4(a) we show what happens if MGM1 does not restart the guard time when it receives MGM2's *join* message. When the guard time ends, MGM1 joins the group (due to an MN's *join* message). In this case, MGM2 receives the message before t_2, which is the end of its 3Δ guard time, whereas MGM1 does not receive the join before t_1, which is the end of its guard time. Thus, they elect two different coordinators. In Figure 4(b), MGM1 correctly restarts the guard time when it receives MGM2's *join* message. Thus, MGM2 receives MGM1's *join* message only after the end of its guard time, and it does not influence its election of the coordinator.

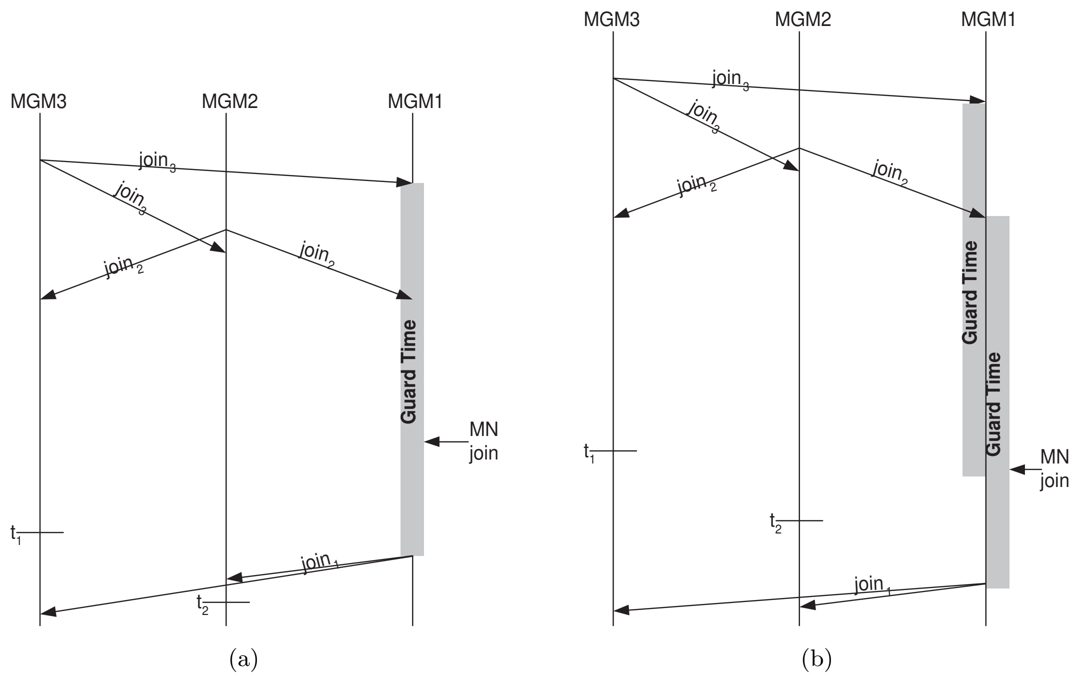


Figure 4: Illustrating the need for restarting the guard: (a) an incorrect election procedure without restarting; and (b) a correct election procedure.

4.2.3 Coordinator transition

When the last MN in the coordinator's domain leaves the group or moves to another domain, the coordinator appoints a new MGM as the new coordinator of the group and informs the group's MGMs of the new coordinator in the forwarded *move* or *leave* message. Subsequently, the leaving coordinator creates a tunnel to the chosen coordinator, and forwards control messages that it still receives on this tunnel. In order to avoid appointing an MGM that has already left, an MGM can not leave the group until it receives the coordinator's permission. If the coordinator notices, after receiving a *leave* or *move* message, that an MGM has no members in the group, it sends a *permission-to-leave* message to that MGM. The only scenario where the coordinator does not permit the MGM to leave when its group empties is if the coordinator has already appointed the MGM to be the new coordinator. In this case, the chosen MGM needs to find a new MGM to replace it as the group's coordinator. When the group is empty, i.e., there are no MNs participating in the group, the coordinator can leave the group without informing other MGMs.

We now explain the selection of the 3Δ time units guard time in the election procedure. When a new joining MGM sends a *join* message for an active group, the group's coordinator may be in transition. In this case, the coordinator does not reply with the group's view immediately. When the transition ends, (i.e., the new coordinator receives the *transition* message), the new coordinator starts handling messages received both directly and via the tunnel from the previous coordinator. Therefore, the *join* sent by new joining MGM is received by the new coordinator at most 2Δ time units after being sent, and thus, the joining MGM receives the group's view from the coordinator within 3Δ time units.

According to the guard-time and the transition time, we calculate the required bound on MN movements. As explained above, when an MN moves into a new domain, the handoff mechanism exploits the fact that the previous MGM holds the coordinator information. To ensure that the handoff mechanism will work properly, we use a bound on MN movements, requiring that an MN will not move to another domain before its current MGM completes its joining procedure and receives the group's view and coordinator information. This way, *handoff* messages can be forwarded to the coordinator within 2Δ time units. We now examine the worst case scenario of the joining procedure in terms of the time it takes the MGM to set the view. The scenario is illustrates in Figure 5.

Let T_{guard} be the guard-time as described in Section 4.2.2. Let $T_{respond}$ be the time that elapses since a coordinator (active or previous) receives a *join* message from an MGM and until the MGM receives a view back from the coordinator. If no transition is taking place, this takes Δ time units. In case of transition, it takes Δ to tunnel the message to the new coordinator, and Δ more from the new-coordinator to the MGM. Thus, in this protocol, $T_{repond} = 2\Delta$. As illustrated in Figure 5, an election begins when MGM1 sends a *join* message at time t. MGM3 receives the message almost immediately, and due to the fact that it does not participate in the group it enters into a guard interval for T_{guard} time units. Right after that, MGM3 receives a *join* message from an MN located in its domain, but MGM3 can send a *join* message only after the guard-time ends. MGM2 joins the group as well, and MGM3 receives its *join* message at $t + 2\Delta$. Therefore, MGM3 resets its guard timer at $t + 2\Delta$ and broadcasts its *join* message only at $t + 2\Delta + T_{guard}$. MGM3 cannot receive additional *join* messages after $t + 2\Delta$, as all MGMs receive MGM1's *join* message by $t + \Delta$, and therefore MGMs that did not send their *join* messages before that time have to wait at least T_{guard} time units before they are allowed to broadcast their *join* message. Therefore, $t + 2\Delta + T_{guard}$ is the latest time in which an

MGm can still have a guard from this election. The group's coordinator receives the message at $t + 3\Delta + T_{guard}$, and sends back the view. MGM3 receives the *view* message by $T_{respond}$. Therefore, only after $3\Delta + T_{guard} + T_{respond}$ time units, after receiving the MN's *join* message, MGM3 receives the group's view and coordinator information. In the subscription model, $T_{guard} = 3\Delta$, and $T_{respond} = 2\Delta$. Thus, after 8Δ time units MGM3 receives the group's view. Using the 10Δ time units movement bound, as described in Section 3.1, we allow a successful handoff procedure in case the MN moves into another domain.

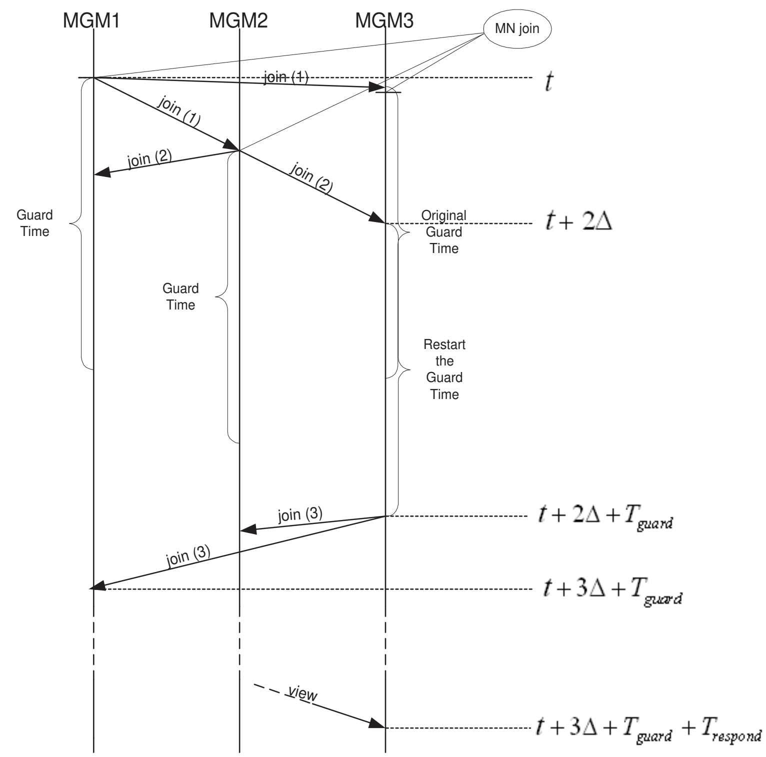


Figure 5: Illustration of the maximal response time of the coordinator to an MGM join.

5 MaGMA Multicast Overlay Solution

In this section, we discuss an alternative MaGMA solution, in which the MGMs provide the multicast service. In this approach, MNs do not obtain the list of subscribers in the group in order to send messages to them, but instead hand multicast messages over to their MGMs. Therefore, there is no need for each MGM to track the full list of subscribers in each group and their addresses. Rather, each MGM needs to know the identities and addresses of its *local* subscribers, (i.e., MNs in its domain), as well as which other MGMs have members in the group.

MNs in other domains are represented by their domains' MGMs. In other words, the control plane of the multicast overlay solution maintains an MGM-level membership and a local MN-level membership, but not a global MN-level membership as the subscription model solutions do.

We present a protocol for this model based on the MGMLeader approach described above. We now overview the group management protocol and present a detailed description and pseudo-code.

An MGM participates in the protocol if it manages at least one MN that belongs to this group. In addition, the MGMs participate in the distribution of data. When an MN needs to send data to a group, it forwards the data packets to its MGM indicating the destined group. The MGM forwards the packets to the other MGMs that participate in the group via the overlay. When an MGM receives data packets from another MGM, it forwards the packets to its local MNs that participate in the group.

An MGM that is not a member of a group joins the group either when an MN in its domain sends a *join* message, or when an MN that already participates in the group moves into its domain and sends a *move* message to inform the MGM of its arrival.

An MN sends a *leave* message to the MGM when it wishes to leave the group. An MGM receiving this message removes the MN from its local view. If the local view becomes empty, i.e., no local MNs are group members, the MGM initiates a procedure for leaving the group.

5.1 Data Structures and Message Staructures

Every MGM and MN holds a unique ID (UID), MGMs also manage an increasing *Local Event Counter* (LEC). The LEC is initialized to 1 and incremented upon local join, leave, and move events, and during transition. In addition, MGMs hold the data structures described in Table 1.

The formats of the messages structure sent by the MGMs are described in Figure 6.

5.2 *LEC* Verification

When an MGM receives a message from another MGM, it verifies that the *LEC* included in the received message is higher than the one stored in its view. If the *LEC* is lower, the receiver discards the message. Otherwise, it updates the *LEC* in the view and processes the message

Data Structures			
Data Structure	**Description**	**Type**	**Init**
groupView	MGMs participating in the group	set of $\langle MGM, LEC\rangle$	$\emptyset$
localView	MNs participating in the group	set of MNs	$\emptyset$
joiners-list	MGMs participating in the election procedure	set of $\langle MGM, LEC\rangle$	$\emptyset$
coord	UID of the group's coordinator	MGM	$\perp$
next-coord	UID of the next coordinator in case of transition	MGM	$\perp$
active-tunnels	MGMs that joined up to Δ time units ago	set of $MGMs$	$\emptyset$
msgBuffer	received messages	set of msgs	$\emptyset$
abstinence	Indicating on election participation	{true, false}	false
leaveFlag	Indicating if an MGM can leave	{true, false}	true
$lec(groupView, i)$	returns the LEC of an MGM from *groupView*	func:(set of ⟨MGM,LEC⟩,MGM)→ LEC	
Externally Provided Function			
coordSelect	determenistic function for coordinator selection	func:set of MGMs→ MGM	

Table 1: Data structures and functions used by MGMs.

join: $\langle MGM_i, join, LEC_i\rangle$
leave: $\langle MGM_i, leave, LEC_i\rangle$
handoff: $\langle MGM_i, handoff, LEC_i, UID_{MN}\rangle$
view: $\langle MGM_i, view, groupView\rangle$
transition-request: $\langle MGM_i, transition\text{-}req, groupView, LEC_i\rangle$
transition-granted: $\langle MGM_i, transition\text{-}granted, LEC_i\rangle$
transition-denied: $\langle MGM_i, transition\text{-}denied, LEC_i\rangle$
new-coord: $\langle MGM_i, new\text{-}coord, LEC_i\rangle$

Figure 6: Message types (MGM_i is the sender).

according to its type as described below. The *LEC* verification is shown in Figure 7 Lines 2–5.

5.3 Joining a Group

An MGM joins the group by broadcasting a *join* message to all MGMs in the network. At the same time, the MGM initiates a guard-timer Figure 8 line 5. This timer period is used for the bootstrapping procedure and coordinator election in case no coordinator exists at the time of join. The coordinator election procedure is similar to the one presented in Section 4.2.2. The only differences are that the guard time is 4Δ time units instead of 3Δ as above. The difference in the guard time stems from the different transition procedure employed in this protocol,

```
upon receiving msg ⟨..., LEC_k⟩ from MGM_k do
  if LEC_k ≤ lec(groupView, k) then                    {LEC verification}
    discard msg
  else
    lec(groupView, k) ← LEC_k
    if next-coord≠⊥ and coord= i then                  {tunneling}
      forward msg to next-coord
      add msg to msgBuffer
    else if active-tunnels≠ ∅ and coord= i then
      forward msg to active-tunnels
    else
      process msg according to type
```

Figure 7: Message pre-processing.

as described in Section 5.4 below. As joining is not synchronized among the MGMs, several MGMs can join simultaneously. During the timer period, the joining MGM stores in *joiners-list* UIDs of other MGMs that send *join* messages. If the group is active, i.e., a coordinator exists, it sends back the group's view, Figure 9 line 5, before the timer expires and the MGM stops the bootstrapping procedure (Figure 10). If the group is inactive, after the timer expires, the joining MGM can elect a coordinator according to the UIDs stored in *joiners-list* and the deterministic function *coordSelect* and assign *groupView* to hold the identities of the MGMs stored in *joiners-list*, i.e., the MGMs that participated in the election (Figure 8 lines 7–8). After setting the coordinator and the group's view, due to the end of the election procedure or due to the reception of *view* message from the coordinator, the joining MGM flushes its *joiners-list*. To avoid inconsistency, an MGM not participating in the group that receives a *join* from another MGM will not join the group for 4Δ time units (by turning on its *abstinence* flag for 4Δ time units).

```
1:  upon joining group do
2:    wait until !abstinence
3:    broadcast ⟨join, LEC⟩
4:    add ⟨MGM_i, LEC_i⟩ to joiners-list
5:    wait 4Δ time units or until processing view message
6:    if coord=⊥ then
7:      coord = coordSelect(joiners-list)
8:      groupView ← joiners-list
9:      if coord=i then
10:       set leaveFlag←false for Δ time units
11:   joiners-list = ∅
```

Figure 8: MGM join code.

```
processing ⟨MGM_k, join⟩ do
  if coord≠⊥ then
    add ⟨MGM_k, LEC_k⟩ to groupView
    if coord=i then                                   {MGM_i is the coordinator }
      send ⟨view, groupView⟩ to MGM_k
      add MGM_k to active-tunnels for Δ time units
  else
    if joiners-list=∅ then                            {Not in group}
      set abstinence← true for 4Δ time units
    else
      add MGM_k to joiners-list
```

Figure 9: Actions taken upon receiving an MGM join message.

```
processing ⟨MGM_k, view, v⟩ do
  coord ← k
  groupView ← v
  set leaveFlag←true for Δ time units
```

Figure 10: Actions taken upon receiving the group's view.

5.4 Leaving a Group

When an MGM intends to leave the group, it sends a *leave* message to the group's members (Figure 11 line 7). An MGM can leave the group only if its *coord*$\neq\perp$ and if Δ time units passed after it received the group's view. If an MGM needs to leave the group during an ongoing election procedure, it must wait for the election to end (Figure 11 line 2). An MGM receiving a leave message updates its *groupView*, i.e., removes the leaving MGM from its view (Figure 12). If the coordinator intends to leave the group, it must appoint another MGM to become the new coordinator. If the group's view is empty, then the coordinator can leave without sending any messages to other MGMs. In the MaGMA overlay solution, we use a different approach to coordinator transition from the one used in MGMLeader above. Instead of requesting permission to leave from the coordinator, non-coordinator MGMs may leave the group at will. The coordinator, on the other hand, needs to ensure that it has a successor before leaving the group. The coordinator elects an MGM, queries this MGM using a *request-transition* and wait for the MGM's reply (Figure 11 lines 12–14). If the elected MGM declines the transition the coordinator needs to find a new candidate for the transition. If it grants the transition, i.e., sends a *transition-granted* message, the coordinator establishes a tunnel towards the new coordinator for Δ time units and then leave the group. After granting the transition the new coordinator updates the rest of the MGMs participating in the group regarding the transition by sending them a *new-coord* message (Figure 14 line 10). When an MGM receives a *new-coord* message it removes the previous coordinator from its *groupView* and updates its *coord* value with the new coordinator UID (Figure 13). This approach expedites leaving the group for all MGMs except the coordinator. In Section 5.6 we further discuss the transition procedure.

```
upon leaving group do
  wait until coord≠⊥
  wait until !abstinence
  wait until !leaveFlag
  LEC←LEC+1
  if coord≠ i then
    send ⟨leave, LEC⟩ message to MGMs in groupView
  else
    wait until activeTunnels= ∅
    remove MGM_i from groupView
    while coord= i and ‖ groupView ‖> 1 do
      next-coord← coordSelect(groupView)
      send ⟨transition-request, groupView, LEC⟩ to next-coord
      wait for reply from next-coord
      if receiving transition-denied then
        next-coord←⊥
        process messages in msgBuffer and clear msgBuffer
      else                                        {received transition-granted}
        coord←next-coord
        establish a control tunnel towards coord for Δ time
    LEC←LEC+1
    coord←⊥
    next-coord←⊥
    groupView← ∅
```

Figure 11: MGM leave code.

```
processing ⟨MGM_k, leave⟩ msg do
  wait until coord≠⊥
  remove MGM_k from groupView
```

Figure 12: MGM actions upon receiving leave message.

processing $\langle MGM_k, \textit{new-coord}\rangle$ *msg* **do**
  if $\textit{coord}=\perp$ **then** {*this can occur during transition*}
    wait till receiving *view* message and process
  remove *coord* from *groupView*
  set $\textit{coord} \leftarrow k$

Figure 13: MGM actions upon receiving new-coord message.

processing $\langle MGM_k, \textit{transition-request}, v\rangle$ *msg* **do** {MGM_k *is the coord*}
  LEC←LEC+1
  if $\textit{localView}=\perp$ **then**
    send $\langle \textit{transition-denied}, LEC\rangle$ to *coord*
  else
    send $\langle \textit{transition-granted}, LEC\rangle$ to *coord*
    $\textit{groupView} \leftarrow v$
    remove *coord* from *groupView*
    $\textit{coord} \leftarrow i$
    send $\langle \textit{new-coord}, LEC\rangle$ message to MGMs in *groupView*
    set *leaveFlag*←false for Δ time units

Figure 14: MGM actions upon receiving transition-request message.

5.5 Handling MNs movements

When an MN moves from one domain to another, it performs a WiFi or cellular handoff and receives a new IP address. It then sends a *move* message to the new MGM with its UID and the UID of its former MGM. The MGM adds the MN to its local view and sends a *handoff* message to the former MGM, only if it needs to join the group, i.e., the moving MN is the first MN participating in the group (Figure 16). When the former MGM receives such a message (Figure 17), it removes the MN from its local view, forwards the *handoff* message to the coordinator. The coordinator, in turn, forwards *join* messages on behalf of the new MGM to the group's MGMs, and sends the current group view to the new MGM. This message flow is illustrated in Figure 15.

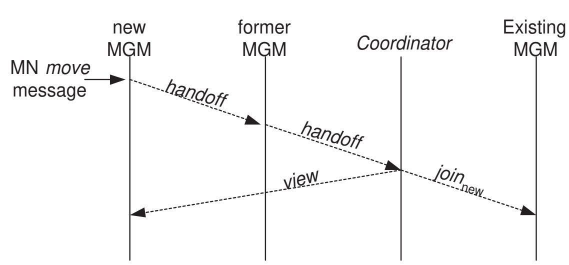


Figure 15: Move message flow in MaGMA Multicast Overlay.

If the new MGM already participates in the group, it does not send the *handoff* message to the previous MGM. The previous MGM detects the absence of the MN and removes the MN from its *localView*. If the *localView* becomes empty due to the MN's movement the MGM leaves the group as described in Section 5.4.

upon receiving MN_k move message $\langle move, MGM_l \rangle$ **do**
  wait until !*abstinence*
  LEC←LEC+1
  if *groupView*= ∅ **then** {*need to join the group*}
    send $\langle i, handoff, LEC, MN_k \rangle$ to MGM_l

Figure 16: MGM move code.

5.6 Coordinator Transition

When all the MNs located in the coordinator's domain leave the group or move to another domain, the coordinator will attempt to leave the group. The coordinator can not leave the group immediately after its *localView* becomes empty. It needs to verify that its *active-tunnels* is empty as well, and that its *leaveFlag= true*, i.e., the MGM has functioned as the coordinator at least Δ time units. When all three conditions hold, the coordinator can start the transition procedure. It is the coordinator's responsibility to designate another MGM as the new coordinator. The coordinator elects an MGM from its group's view, and sends a query message, *transition-request*, to that MGM. If the MGM is active, i.e., its local view is not empty, then it confirms the transition by sending a *transition-granted* message and informs the rest of the MGMs in the group about the transition by sending them *new-coord* messages. If its local view is empty (this can happen if the MGM has sent a *leave* message during the transition query), then it replies with a *transition-denied* message, and the coordinator needs to query another MGM.

During the transition procedure, all MGM messages received by the coordinator are buffered and forwarded to the new coordinator, as described in Figure 7 Lines 6–8. If the coordinator receives a *transition-denied*, then it handles the buffered messages. If it receives a *transition-granted* message, it leaves the group. In addition, it forwards to the new coordinator all MGM messages sent prior to the transition, for up to a Δ time units. To avoid multiple tunneling, which would cause extra delay in message processing, a coordinator can initiate a new transition

```
processing ⟨MGM_k, handoff, MN_l⟩ msg do
  wait until !abstinence
  if MN_l ≠⊥ then                                   {MGM_i is the previous MGM}
    remove MN_l from localView
    LEC←LEC+1
  if coord≠ i then
    add MGM_k to groupView
    send ⟨MGM_k, handoff, LEC_k, ⊥⟩ to coord
    if localView= ∅ then
      wait until !leaveFlag
      send ⟨leave, LEC⟩ message to MGMs in groupView
  else
    if MGM_k ∉ groupView then
      send MGM_k join message to MGMs in groupView
      add MGM_k to groupView
      send ⟨view, groupView⟩ to MGM_k
    if MN_l ≠⊥ then
      while localView= ∅ and activeTunnels= ∅ do
        wait until !leaveFlag
        next-coord← coordSelect(groupView)
        send ⟨transition-request, groupView, LEC⟩ to next-coord
        add next-coord to active-tunnels
        wait for reply from next-coord
        if receiving transition-denied then
          remove next-coord from active-tunnels
          process messages in msgBuffer and clear msgBuffer
          next-coord←⊥
        else                                        {received transition-granted}
          coord←next-coord
          establish a control tunnel towards coord for Δ time
      LEC←LEC+1
      coord←⊥
      next-coord←⊥
      groupView← ∅
```

Figure 17: MGM handoff code.

only after it functions as a coordinator at least Δ time units.

If the coordinator's local view and group view are empty, i.e., no MNs and MGMs are participating in group, then it can leave the group without sending any transition message. In order to avoid frequent coordinator changes and election procedures that flood the network, the coordinator needs to wait a minimum time interval before electing a new coordinator or leaving the group.

We now explain the selection of the 4Δ time units guard time in the election procedure. When a new MGM sends a *join* message for an active group, the group's coordinator may be in transition. In this situation, the coordinator does not reply immediately. If the transition succeeds, then the new coordinator replies with the group's view, which takes a total of at most 3Δ time units. On the other hand, if the transition fails, i.e., the coordinator receives a *transition-denied* message, then it can take up to 3Δ time units for the coordinator to handle the buffered message, and up to 4Δ time units for the joiner to get a response. These scenarios are depicted in Figure 18.

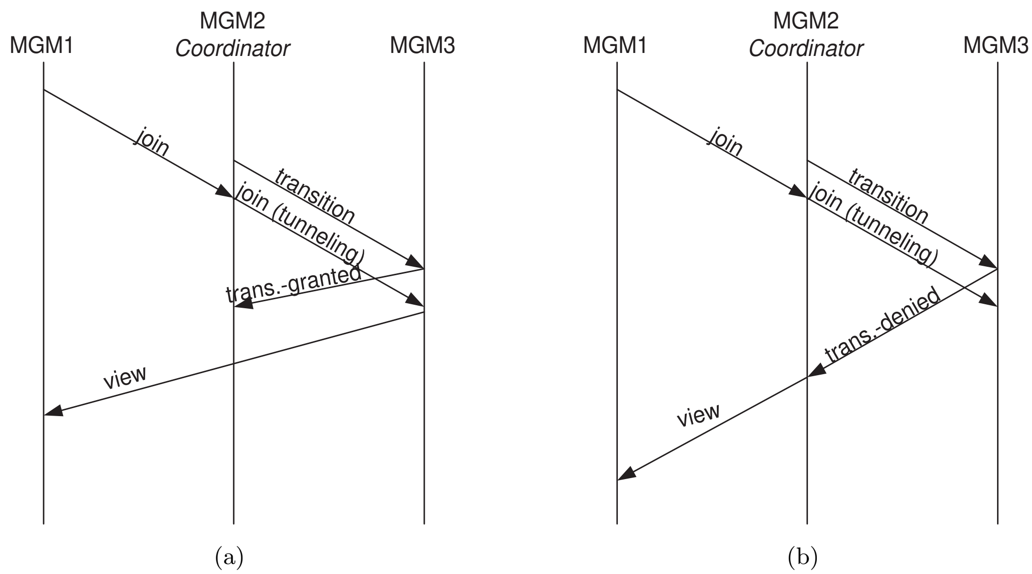


Figure 18: Two coordinator transition scenarios, where the group view is received within (a) 3Δ and (b) 4Δ time units.

According to this calculation, in MaGMA Multicast Overlay $T_{respond} = 3\Delta$. As $T_{guard} = 4\Delta$, the delay of the scenario depicted in Figure 5 Section 4.2.3, bounds MN's movement by 10Δ time units, as required in Section 3.1.

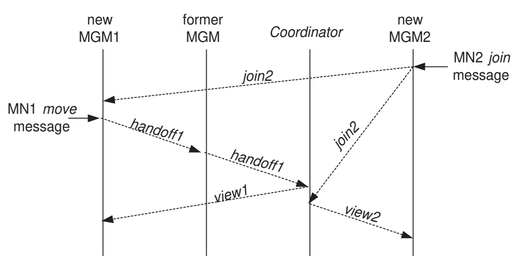
5.7 Ensuring View Consistency

The main goal of the control plane is keeping view consistency. The control plane must enable the MGMs to hold a coherent and up-to-date view of the MGMs participating in the group, as well as maintain an accurate view of the local MNs participating in the group. We intuitively describe several possible problems and their solutions, in [11] we provide a formal proof that the protocol keeps view consistency.

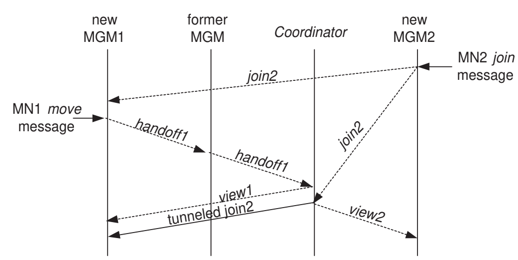
Due to the fact that MGMs can join and leave the group, a potential inconsistency can occur with careless handling of joins, when two or more MGMs join the group simultaneously.

Figure 19(a) illustrates a scenario that could have occurred if we would not have used tunnels. In this example, MN2 joins the group and MN1 moves from one domain to another. Neither MGM1 nor MGM2 are in the group before these events. Due to MN2's *join* message MGM2 broadcasts a *join* message. This message arrives to MGM1 before it receives MN1's *move* message, thus MGM1 ignores MGM2's message. Upon receiving MN1's message, MGM1 sends a *handoff* message to the former MGM. The message is forwarded to the coordinator, which receives MGM1's message before MGM2's message. Thus, MGM1 is unaware of MGM2 and holds an inconsistent view of the group.

MaGMA's solution for this problem is to tunnel new events from the coordinator to newly added group members. The group coordinator receiving a message (*join* or *move*) from a new MGM forwards to that MGM all incoming messages from other MGMs for Δ time units. Thus, new MGMs receive all group's events and maintain a coherent and up-to-date view of the group. The solution is illustrated in Figure 19(b).



(a)



(b)

Figure 19: (a) An examples for potential view inconsistency with a simplified control plane and (b) the tunneling solution.

Due to the role of the coordinator as the forwarder of some but not all of the MGMs' messages, in some cases, an MGM can receive the message of another MGM out of order. For example in Figure 20(a), a new MGM is joining the group and broadcasts a *join* message to the MGMs in the network. The coordinator sends to the new MGM the group's view. Meanwhile, an existing MGM sends a *leave* message. This message is sent prior to the reception of the *join* message. Upon receiving the *leave* message, the coordinator tunnels the message to the new MGM. Meanwhile, the leaving MGM re-joins the group by broadcasting a *join* message. The *join* message is received by the new MGM before the reception of the previous *leave* message. If this situation is not adequately handled, as a result the new MGM may hold an inconsistent view of the group, due to the fact that it removed the existing MGM from its *groupView*.

To address this difficulty, each MGM maintains an increasing *Local Event Counter* (LEC) for every group, initialized to 1. Whenever an MGM receives a *join*, *leave*, or *move* message from a local MN, it increments the appropriate LEC. In addition, the MGM keeps, for every group, a *LECvector*, holding the highest known LEC for each MGM in this group. The LEC solution for the scenario described above is illustrated in Figure 20(b).

In every message sent from one MGM to another, the sender's LEC is included. When an MGM receives a packet, it checks the LEC. If the LEC in the message is lower than the known one, then it ignores the message. When the coordinator sends the group's view to new joining MGMs, it includes the LECs corresponding to the view it is sending to the new MGM.

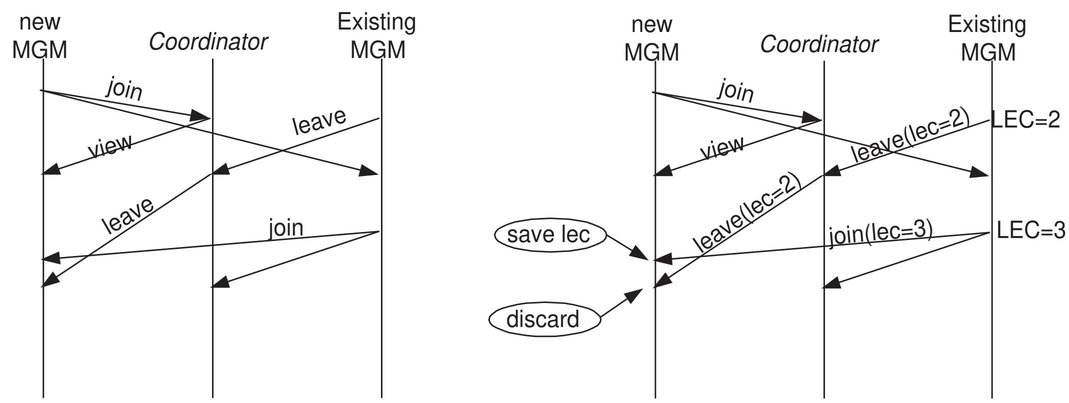


Figure 20: (a) An example for potential view inconsistency with a simplified control plane and (b) the LEC solution.

We now describe another scenario, illustrated in Figure 21(a), where out-of-order reception of messages might have caused view inconsistency. In this scenario, MGM1 and MGM2 join a group. The group's coordinator receives MGM2's *join* message before receiving MGM1's *join* message. Therefore, when it receives MGM1's *join* message, it sends back to MGM1 the group's view and tunnels the *join* message to MGM2. MGM1 receives the group's view almost immediately and then it leaves the group by sending *leave* message to MGMs in its *groupView* (the coordinator and MGM2). MGM2 receives the message immediately and therefore removes MGM1 from its *groupView*. Right after that, MGM2 receives the tunneled *join* message previously sent by MGM1, and adds MGM1 to the group's view. In addition, MGM1's *leave* message is received by the coordinator after it closes the tunnel towards MGM2, and therefore the coordinator does not forward the message to MGM2. Consequently, MGM2 thinks that MGM1 is a member of the group. We note that the LEC mechanism cannot prevent this inconsistency as MGM2 removed MGM1 from its *groupView* and therefore does not have any information regarding MGM1. MaGMA's solution to this problem is a guard time maintained after receiving the group's view, requiring that an MGM can not leave the group less than Δ time units after receiving the group's view from the coordinator, as illustrated in Figure 21(b) (see Figure 11 line 4 and Figure 10 line 4). This way, we prevent out-of-order reception of consecutive *join* and *leave* messages.

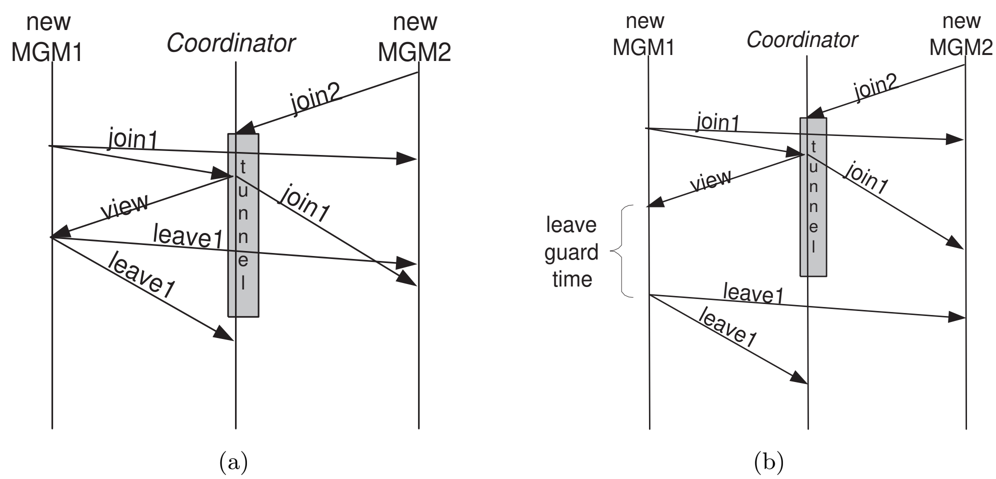


Figure 21: (a) An example for potential view inconsistency with a simplified control plane and (b) the guard-time solution.

6 Control Plane Evaluation

6.1 Simulations and Analysis Scenario

We now evaluate the overhead associated with the control protocol. We simulate the following uniform network model:

- 10 domains (Domains 1-10), 1 MGM in each domain;
- 10-100 receiving MNs, initially uniformly distributed in Domains 1-10, then moving among these domains;
- a single group, where every receiving MN participates in this group;
- Constant delay between MGMs, $\Delta = 0.4sec$.

6.2 Comparing MaGMA's Two Solutions

As described in Section 3.2, we assume that *move* messages dominate the control traffic. Therefore we simulate MaGMA's control protocols in this setting, and measure the average control overhead associated with a single *move* message in both models. The average is calculated over 1000 events for each number of MNs. In each event, a random MN moves to a new random domain. The results are depicted in Figure 22, with 95% confidence intervals for MaGMA Multicast Overlay. We also mathematically analyze the expected control overhead. The detailed analyses of MGMLeader and the MaGMA multicast overlay solution are presented in [11]. For MGMFlood, this is straightforward. Since each control message is sent to all MGMs, and there are nine receiving MGMs, together with the MN's *move* message the overhead is exactly ten messages per *move* event. Not surprisingly, the analysis and simulation results for this protocol accurately match each other (see Figure 22).

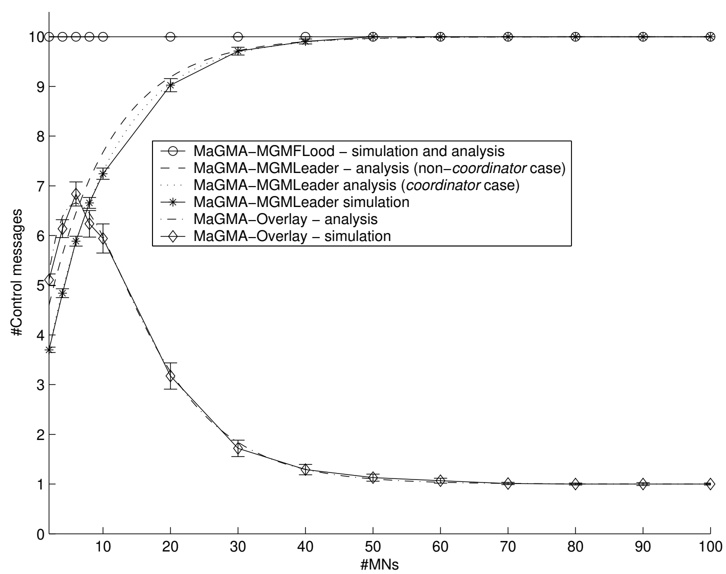


Figure 22: Average number of control messages per movement, uniform system, varying number of MNs, fixed number of domains: analysis vs. simulations.

For MGMLeader and the MaGMA overlay solution, we give two separate analyses covering two different situations. Recall that in these protocols, a new MGM joining a group communicates with the MN's former MGM, which forwards the message to the coordinator. We distinguish the case that the former MGM is the group's coordinator (the *coordinator case*), from the case that the former MGM is not the coordinator (the *non-coordinator case*). The analyses of these cases are depicted separately in Figure 22. In MGMLeader, one more control message is sent in the non-coordinator case as compared to the coordinator case— the message from the former MGM to the coordinator.

In MaGMA's multicast overlay solution, the situation is reversed: if the group becomes empty following the move, then in the non-coordinator case, the former MGM can leave the group without asking for permission, whereas in the coordinator case, the coordinator needs to find a new coordinator to replace it and to inform other MGMs of the transition. Therefore, the overhead in the coordinator case is larger.

With MGMLeader, the overhead increases with the number of MGMs that have members in the group. In sparse groups, few MGMs are involved, and hence few control messages are sent. With MaGMA's multicast overlay solution, the overhead is similar to MGMLeader in sparse groups (2-10 MNs), but then the overhead *decreases* with the number of MGMs that have members in the group. In dense groups (50-100 MNs), most movements do not cause any changes in MGM-level membership, and therefore MaGMA's overlay solution sends only two messages per movement: from the moving MN to the new MGM and from the new MGM to the former MGM. The remaining MGMs do not need to be informed of the move.

We conclude that in the subscription model, MGMLeader is preferable for sparse groups, whereas the much simpler MGMFlood may be adequate for dense groups in which all or most MGMs participate. It is also clear that for large groups, MaGMA's multicast overlay greatly outperforms MGMFlood and MGMLeader, and this solution is therefore preferable where MGMs that support this functionality can be deployed.

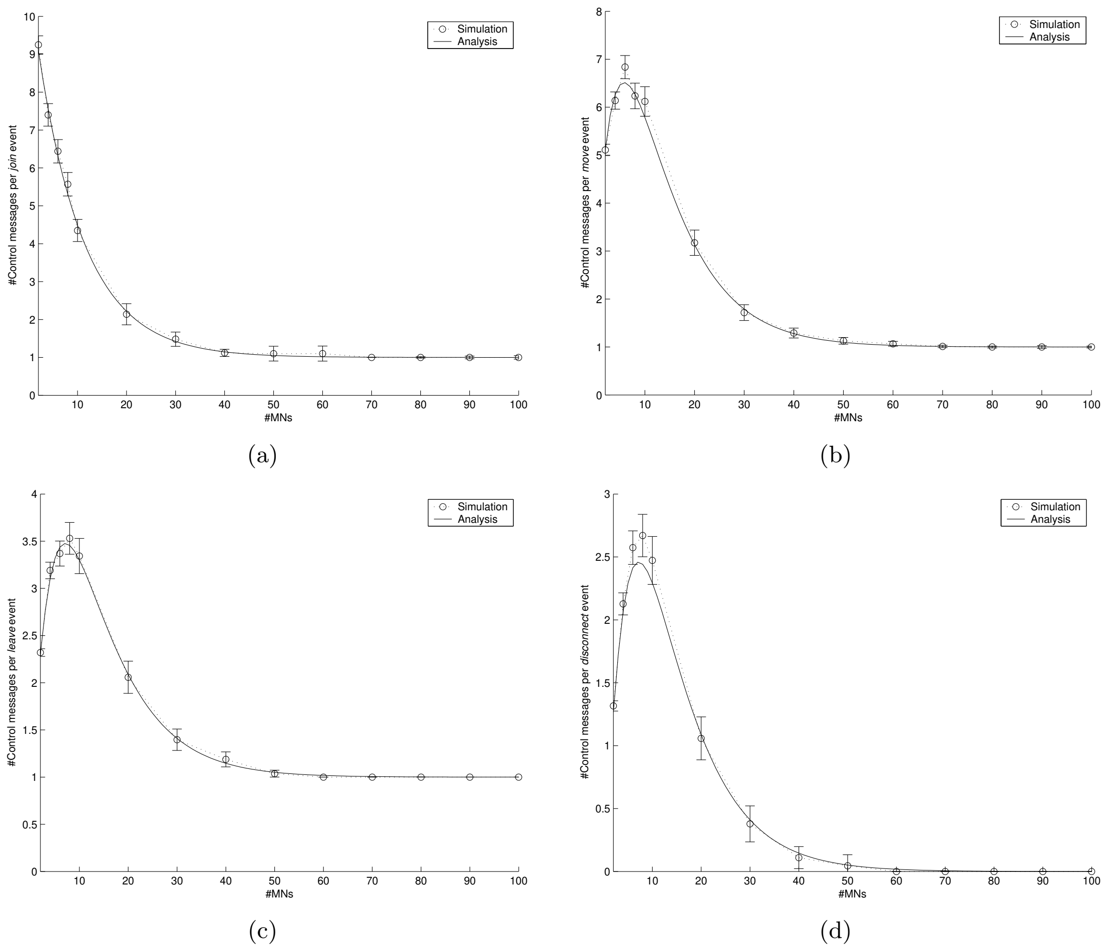
6.3 Evaluating MaGMA Multicast Overlay

6.3.1 Control Overhead

We now evaluate the control overhead of join, leave, movement and disconnection of an MN in the MaGMA multicast overlay model. The average number of control messages sent per event are depicted in Figure 23. The detailed analyses of the join, move, leave and disconnect events are presented in [11].

From Figure 23(a) we see that in a sparse group when a new MN joins the probability that the group does not exist in the domain is high and therefore the MGM has to broadcast the *join* message to the rest of the MGMs. In dense a dense group (40-100 MNs) when the MGM already participates in the group and therefore only a single message is sent (the MN's *join* message). From Figure 23(b), we see that in a sparse group, due to the handoff mechanism, the number of messages sent due to a move event is smaller than the number of messages sent due to a join event.

From Figures 23(c) and 23(d) we see that the only difference between the results is the *leave* message sent by the MN. We also see that in a dense group there is no need to inform the participating MGMs, as there are additional MNs in the domain that participate in the group.



(a) (b) (c) (d)

Figure 23: Average number of control messages per (a) join, (b) movement, (c) leave and (d) disconnect event, uniform system, varying number of MNs, fixed number of domains: analysis vs. simulations.

6.3.2 Time to Stabilization

Another performance parameter of MaGMA that we examine is time-to-stabilization (TTS). This parameter measures the amount of time it takes an event to propagate through the network/group. For example, when an MN joins or leaves the group or when an MN moves to another domain, it sends a message to the local MGM indicating this event. As a result, further messages are sent by the local MGM and other MGMs. TTS measures the time from the beginning of the event until the time where the last message sent due to this event is received. We consider two measurements of TTS, one from the MN's perspective denoted as TTS_{MN}, where we measure TTS from the time where the MN sends a message or from the time of its disconnection, and the second from the MGM's perspective, denoted as TTS_{MGM}, where we measure TTS from the time the MGM receives the MN's message or detects the MN's disconnection.

Table 2 summarizes the maximum possible TTS_{MN} and TTS_{MGM} of the four events. The maximal TTS for a join event occurs when the MN sends a *join* message to the local MGM and the MGM needs to join the group, for a leave event, the maximal TTS occurs when the leaving MN's MGM functions as the coordinator and need to initiate a transition. Similarly, for the move and disconnect events the maximal TTS is receives when there is a transition procedure due to the MN's event.

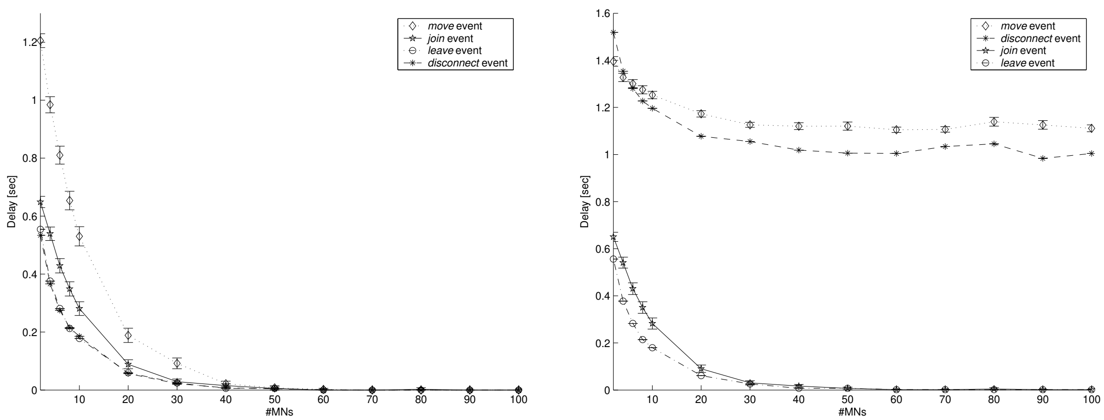
	max TTS_{MN}	max TTS_{MGM}
join	3Δ	2Δ
leave	3Δ	2Δ
move	$\tau + 3\Delta$	3Δ
disconnect	$\tau + 2\Delta$	2Δ

Table 2: Maximum TTS according to the four events.

We evaluate the average TTS of the four events using simulations based on the uniform network. Figure 24 shows the results; 95% confidence intervals are shown. The results of the TTS_{MGM} are depicted in Figure 24(a) and the results of the TTS_{MN} are depicted in Figure 24(b).

From Figure 24(a) it is clear that when the network becomes denser the TTS_{MGM} is equal to zero for all events, this stems from the fact that MGMs do not need to send further messages upon an event, for example if an MGM receives a *join* message it does not need to join the group as it already participates in the group. In sparse networks, we note that for move events the TTS is equal to the maximal TTS, see Table 2, and that the TTS of leave and disconnect events is similar, this stems from the fact that MGMs do not distinguish between these events.

From Figure 24(b) we see that the average TTS of the move events decreases with the number of the MNs, and in dense network (more than 20 MNs) the TTS is around $\tau = 1.1sec$, similar to the disconnect event. The main difference between TTS_{MGM} and TTS_{MN} is with disconnect and move events, due to the fact that detection time is larger than the delay of a message, and that join and leave events are based on message passing.



(a) (b)

Figure 24: Time to stabilization per event (a) from MGM detection (b) from the beggining of the event, $\Delta = 0.4sec$, $\tau = 1.1sec$, uniform system, varying number of MNs, fixed number of domains: simulation.

7 MaGMA's Data Plane

Until this point, we have focused on MaGMA's control plane. We now examine the data plane, i.e., the transport of multicast data among group members.

7.1 Comparing MaGMA's Two Solutions

We now compare the data planes of the two MaGMA solutions. In the subscription model, a sender wishing to initiate a multicast session retrieves the group view and creates unicast streams to all the group members in order to send data to them. In the multicast overlay solution, the sender sends data messages to its MGM indicating the target group, and the MGM forwards the messages via the overlay to other MGMs listed in its group view. Upon receiving data messages, the MGMs forward the messages to their local MNs participating in the group (using multicast or unicast communication within that domain). The latter approach reduces the load on the IP infrastructure and supports larger groups better.

To illustrate the differences between the two models, we analyze the average number of incoming streams per domain. The detailed analysis is available in [11]. We use the following uniform network in our calculations:

- 11 domains (Domains 0-11), 1 MGM in each domain;
- 10-200 receiving MNs, uniformly distributed in Domains 1-10;
- 8 groups, where every receiving MN participates in a single group chosen uniformly at random;
- a fixed number of sources in Domain 0;
- all groups are active.

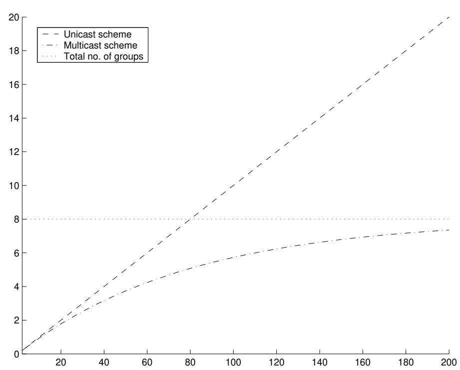


Figure 25: Average number of incoming streams per domain in uniform system with 8 groups, varying number of MNs, fixed number of domains: analysis.

Figure 25 shows that using the unicast scheme, the average number of incoming streams is the average number of participating MNs within the domain, whereas using the multicast scheme, it is the average number of groups in the domain.

7.2 Comparison with Mobile IP

The use of Mobile IP to forward traffic to a mobile user traversing the network suffers from poor performance due to triangle routing. In contrast, in MaGMA, traffic is sent either directly to the destination nodes (in the subscription model) or using the MGM's overlay. In the latter case, if MGMs are located in every domain, or at least in strategic central points in the core, then network traffic is also sent using an almost direct route.

To illustrate this advantage of MaGMA, we simulate a constant bit rate (CBR) UDP session from a static source to a moving receiver. We measure the performance of both MaGMA and Mobile IP. We simulate a network with four domains. The source is located in Domain 0, the receiver is initially located in Domain 1 and then moves toward Domain 3 through Domain 2. Domain 1 functions as the home domain of the receiver in the Mobile IP simulations. Figure 26 illustrates the simulated network.

We measure the average end to end delay in three architectures: Mobile IP, MaGMA multicast overlay model, and MaGMA subscription model. Figure 27 shows the result of the simulations. It is clear that when the receiver is located at its home domain, the three architectures perform similarly. When the receiver moves outside its home domain, the triangle routing causes the packet delay to increase by a factor of 3, whereas using MaGMA the delay does not increase due to the use of the optimal route.

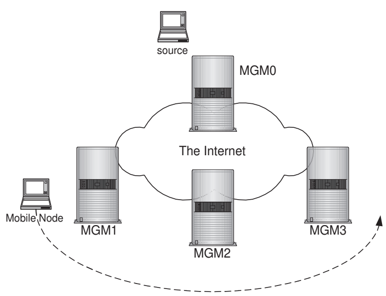


Figure 26: Scenario simulated in Figure 27.

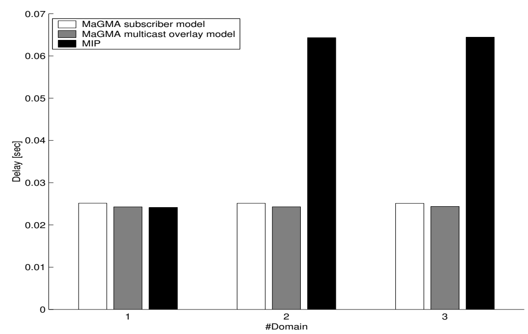


Figure 27: Average end-to-end packet delay for system depicted in Figure 26: simulation.

Another simulation that we conducted, emphasizes MaGMA's advantages in packet loss comparing to MIP. We measured the packet loss of a CBR UDP session where the source is static and the receiver is an MN. The source is located in Domain 1, the receiver is initially located in Domain 1 and the moves towards Domain 4 through Domains 2 and 3. Domain 1 functions as the home domain of the receiver in the MIP simulations. Figure 28 illustrates the simulated network.

We measure the packet loss during the movement of the receiver between domains. Figure 29 shows the result of the simulation. The results emphasize the advantages of MaGMA, while in MIP the packet loss depends on the distance of the MN from its home domain, in MaGMA the packet loss depends on delay between the new and previous MGMs. In MIP, when an MN

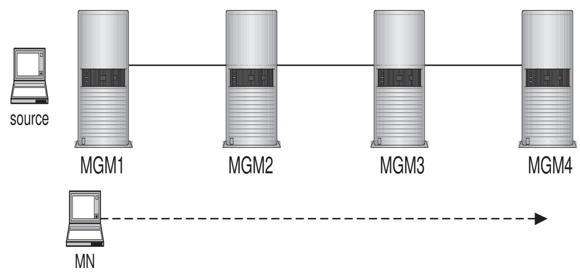


Figure 28: Scenario simulated in Figure 29.

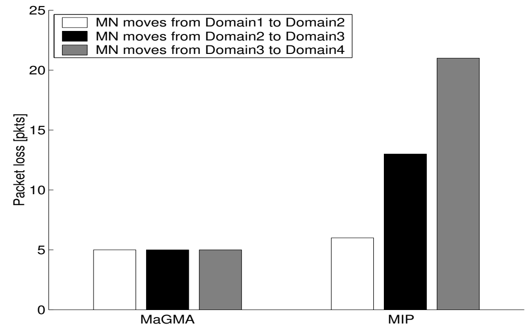


Figure 29: Packet loss for system depicted in Figure 28: simulation.

moves into a foreign domain, it informs the HA regarding its new address. After receiving this update, the HA can tunnel the data messages to the current location of the MN, therefore the delay of the update message, i.e., the distance of the MN from its home, is critical to the number of packet loss. On the other hand, in MaGMA, when a MN moves into a new domain the new MGM establishes a connection with the previous MGM, thus it enables the delivery of data messages prior to the ending of the joining procedure.

Another important parameter is the overhead of the headers added to every data packet. In a simple unicast session, the header is composed of the UDP header, 8 bytes, and the IP address, 20 bytes. In MIP, where the MN is located in its home domain and there is no need to tunnel packets, the overhead is the same as in the simple unicast session. When the MN moves into a foreign domain the HA tunnels the packets to the FA using IP encapsulation, thus 20 extra bytes are added to the header. On the other hand, in MaGMA subscription model, no matter where the MN is located, the overhead is as in the simple unicast session. This is due to the fact that the communication between MNs is conducted using direct unicat sessions.

In the multicast related protocols we compare the multicast extension to MIP with MaGMA multicast overlay model. In MIP with multicast, the HA forwards the multicast messages to MNs located in foreign domains. It uses the IP-in-IP encapsulation, two additional IP headers are added by the HA: one to tunnel the message to the FA, and another to tunnel the message from the FA to the MN. In MaGMA multicast overlay model, the data is forwarded over the overlay. MGMs receiving the data messages and forward the messages to local MNs need to know the group ID. Thus, 4 extra bytes are used to identify the group, as in class D addresses. Figure 30 illustrates the headers in each of the aforementioned methods.

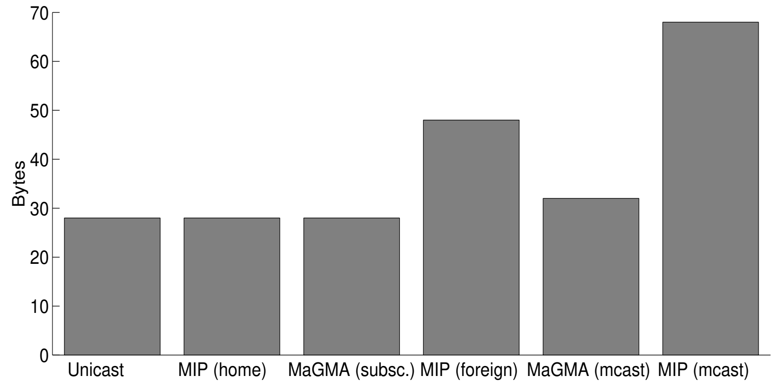


Figure 30: Header sizes in unicat and multicast communication methods.

8 Prototype Implementation

As a proof-of-concept, we have built a SIP-based prototype of MaGMA's MGM. The MGM uses the NIST-SIP [15] implementation. It is implemented in JAVA, and runs on a standard Intel Pentium 4 PC. Our MNs are WiFi enabled iPAQ PDAs. The MaGMA client software is implemented in C++ on top of the Microsoft Real Time Communication (RTC) client [12]. We have demonstrated MaGMA by running an instant messaging application on the iPAQs.

We have implemented MaGMA Multicast Overlay, where the MGM serves a message duplicator. The MGM consists of three components. One component is a SIP parser for processing the received SIP messages. The second is a database that holds information about the current groups and the MNs subscribed to each group. The third part is an MaGMA processor, which updates the database according to the received *move, join, leave* messages, and duplicates and forwards data messages to the group members.

We illustrate MaGMA's flexibility and general applicability by using a standard wireless environment composed of WiFi access points, regular PC-based servers, and iPAQs, and by basing our prototype on an existing SIP implementation. This work is ongoing and we plan to extend it to larger scale infrastructures.

9 Future Work

MaGMA is a general, open, and flexible architecture. We anticipate that the MaGMA-like architectures and similar services will be major components in 3G and B3G networks. As such, we believe that there is plenty of room for proposing enhancements, optimizations, and service extensions beyond the basic services and protocol suite introduced in this paper. We now briefly outline possible service extensions.

Extending MaGMA to support crashes and dynamic membership changes of MGMs is a natural extension to this kind of architecture. As explained above, this extension is not trivial and incorporates additional control overhead and more control mechanisms.

Another interesting direction for future work would be supporting hybrid networks, which are composed of both ad-hoc networks and access-point based networks [4]. In such networks, some MNs may be unable to directly communicate with any MGM. Therefore, one would need to delegate parts of the MGM's functionality to one of the ad-hoc nodes. This node would function both as the group manager representing the group members located in the ad-hoc network and as a relay forwarding the group's session packets to the group members it represents.

A third interesting future extension would be providing advanced services for group applications, e.g., floor control services, as specified in [29]. Since MaGMA is a distributed and dynamic architecture, supporting floor control in this kind of an environment can be more difficult than in traditional centralized groupware servers. Moreover, providing cross-platform session management for converged cellular/wireless networks can be challenging when nodes on the different networks have different characteristics, e.g., different compression standards, different delays, bandwidths, and jitter.

10 Conclusions

We have presented MaGMA, an architecture for supporting real-time group services in integrated WiFi/cellular networks. MaGMA is the first comprehensive solution for groupware with seamless mobility and QoS support. MaGMA is very flexible and can co-exist with current as well as emerging wireless network technologies, it can support vertical handoff between WiFi/WiMAX and cellular modes of operation, and it is suitable for incremental deployment.

We have presented MaGMA's control and data planes, and have demonstrated MaGMA's efficiency using simulations and mathematical analysis. We have presented two solution types: a subscription model, suitable for lightweight servers and small groups, and a multicast overlay solution, which is more scalable and better supports large and dense groups. Finally, we described a proof-of-concept prototype implementation. We believe that MaGMA or similar services will play an important role in 3G and B3G environments.

Acknowledgments

We thank Emmanuel Elder and Tzvi Keisar for their help in developing the MaGMA prototype.

References

[1] 3GPP. http://www.3gpp.org.

[2] 3GPP2. http://www.3gpp2.org.

[3] T. Anker, D. Breitgand, D. Dolev, and Z. Levy. CONGRESS: connection-oriented group address resolution services. In *Proceedings of SPIE on Broadband Networking Technologies*, pages 89–100. SPIE Press, 1997.

[4] E. M. Belding-Royer, Y. Sun, and C. E. Perkins. Global Connectivity for IPv4 Mobile Ad hoc Networks. Internet draft, IETF, draft-royer-manet-globalv4-00.txt, Nov. 2001.

[5] Y. Chawathe. Scattercast: an adaptable broadcast distribution framework. *Multimedia Syst.*, 9(1):104–118, 2003, DOI: http://dx.doi.org/10.1007/s00530-002-0082-z.

[6] V. Chikarmane, C. L. Williamson, R. B. Bunt, and W. L. Mackrell. Multicast support for mobile hosts using mobile IP: Design issues and proposed architecture. *Mob. Netw. Appl.*, 3(4):365–379, 1999, DOI: http://dx.doi.org/10.1023/A:1019101521453.

[7] Flarion. 11 key design requirements for a mobile broadband network. Whitepaper, Feb. 2003.

[8] J. F. Huber. Mobile Next Generation Networks. *IEEE Multimedia*, 11(1):72–83, 2004, DOI: 10.1109/MMUL.2004.1261110.

[9] Jabber. http://www.jabber.com.

[10] D. Johnson, C. Perkins, and J. Arkko. Mobility Support in IPv6. RFC 3775 (Proposed Standard), June 2004.

[11] N. Lavi. Supporting Groupware in Mobile Networks. Master's thesis, also available as CCIT TR #528, The Technion - Israel Institute of Technology, April 2005.

[12] Microsoft. Microsoft Real Time Communication (RTC) Client. msdn.microsoft.com/downloads/list/clientapi.asp.

[13] SIEMENS mobile. Push to Talk over Cellular White Paper, 2004.

[14] J. Mysore and V. Bharghavan. A new multicasting-based architecture for internet host mobility. In *Proceedings of the 3rd annual ACM/IEEE international conference on Mobile computing and networking*, pages 161–172. ACM Press, 1997.

[15] National Institute of Standards and Technology. NIST SIP. http://snad.ncsl.nist.gov/proj/iptel/.

[16] Nextel. http://www.nextel.com/services/directconnect.shtml.

[17] Nokia. Push to Talk over Cellular Real-time always-on voice service. http://www.nokia.com/poc/PoC_WP_A4_net.pdf.

[18] ns2 homepage. http://www.isi.edu/nsnam/ns.

[19] Open Mobile Alliance. http://www.openmobilealliance.org.

[20] C. Perkins. IP Mobility Support for IPv4. RFC 3344 (Proposed Standard), August 2002.

[21] C. E. Perkins. IP Mobility Support for IPv4. RFC 3344, IETF, Aug. 2002.

[22] C. E. Perkins and D. B. Johnson. Mobility support in ipv6. In *MobiCom '96: Proceedings of the 2nd annual international conference on Mobile computing and networking*, pages 27–37. ACM Press, 1996.

[23] C. E. Perkins and D. B. Johnson. Route Optimization in Mobile IP. Internet draft, IETF, draft-ietf-mobileip-optim-11.txt, Sep. 2001.

[24] A. C. Snoeren and H. Balakrishnan. An end-to-end approach to host mobility. In *Proceedings of the 6th annual international conference on Mobile computing and networking*, pages 155–166. ACM Press, 2000.

[25] L. Subramanian, I. Stoica, H. Balakrishnan, and R. H. Katz. OverQoS: offering internet QoS using overlays. *SIGCOMM Comput. Commun. Rev.*, 33(1):11–16, 2003, DOI: http://doi.acm.org/10.1145/774763.774764.

[26] C. Tan and S. Pink. Mobicast: a multicast scheme for wireless networks. *Mob. Netw. Appl.*, 5(4):259–271, 2000, DOI: http://dx.doi.org/10.1023/A:1019125015943.

[27] The Yankee Group. NG Push-to-Connect Provides Simple UI and Enriches User Experience, Sep. 2003.

[28] Togabi. http://www.togabi.com.

[29] International Telecommunication Union. *Generic conference control. Recommendation T.124*. Standardization Sector of ITU, Geneva, Switzerland, Febuary 1998.

[30] Verizon Wireless. http://news.vzw.com/news/2003/08/pr2003-08-14.html.

[31] E. Wedlund and H. Schulzrinne. Mobility support using SIP. In *Proceedings of the 2nd ACM international workshop on Wireless mobile multimedia*, pages 76–82. ACM Press, 1999.

[32] WinterGreen. Push To Talk (PTT) Market Opportunities, Market Forecasts, and Market Strategies 2003-2008, Dec. 2003.