Abstract—The diffusion of mobile devices in the working landscape is promoting collaboration across time and space. Following through this development, we investigate opportunities for improving awareness in mobile environments, with a view to enable collaboration under power constraints and transitory network disconnections. We elaborate in particular on synchronous CSCW and expose with it significant details of group awareness, while we contribute a protocol for awareness support over large areas that strikes a balance between energy consumption and notification time. To avoid user disruption, this protocol notifies awareness information in a multicast fashion, while the bandwidth is allocated dynamically among notifications and data requests, minimizing thus the time needed by each one of them and ensuring the isochronous delivery of information to all clients. The efficiency and scalability of our protocol are evaluated with simulation experiments, whereby we compare various notification schemes and choose finally one that changes dynamically over time.

Index Terms—mobile computing, computer supported cooperative work, group awareness, reaction-diffusion metaphor, performance evaluation, reflective middleware.

1 INTRODUCTION
Computer-supported cooperative work (CSCW) is a term that indicates a variety of technologies, which enable teams of workers to cooperate electronically. The emergence of wireless networks and the proliferation of mobile devices have recently given rise to mobile CSCW, where teams can cooperate while on the move. Mobile CSCW is vital in various situations, such as the exploration of dangerous areas, the administration of ad-hoc meetings, the management of emergencies, and the guidance of tourists.

Yet mobile CSCW faces several challenges due to the limitations of mobile devices with regard to memory capacity, display size, and available power. The short life of batteries, for example, may cause unexpected interruptions with disastrous consequences for cooperative work. To extend this life, mobile devices operate usually in the sleep mode, in which they are connected to the network without doing any specific job. Otherwise they operate in the active mode, where they are allowed to perform some operation such as message delivery and data retrieval. If these operations last for long, however, mobile devices consume substantial amounts of energy. Besides that, wireless networks, over which these devices communicate, have low bandwidth and exhibit often significant delays due to temporary disconnections.

These challenges indicate a need for mechanisms that would allow CSCW to adapt to network and device limitations without sacrificing functionality. One of the main functions of CSCW is the provision of information to each worker (referred to as ‘user’ or ‘collaborator’ hereafter) on the presence and the activities of other group members. This information is called group awareness and has been a central topic of research, because it satisfies the need of collaborators to watch each other’s activities and coordinate accordingly their own work [12]. In fact, the situated nature of
CSCW does not favor the definition of static coordination mechanisms, so collaborators must coordinate their activities according to the situation prevailing each time.

Although the recognition of awareness as an important factor of CSCW has led to significant progress in this area, there is a common agreement that awareness does not denote yet a set of related practices [30]. In fact, there exist several definitions of awareness in the CSCW literature, which are not always mutually exclusive. Presence awareness for example refers to the knowledge of who is around [24], while workspace awareness refers to the knowledge of each other’s activities within a shared work context [15]. Peripheral awareness [13], in turn, refers to one’s ability to capture snapshots of the others’ activities while focusing on a different task. There exist also definitions for contextual awareness [17], passive awareness [12], and situation awareness [15], to mention a few. The aim of this paper is to propose mechanisms that can provide awareness to mobile collaborators in a time- and energy-efficient manner, regardless of their location.

1.1 Contributions
This paper is set within the context of mobile computing and CSCW and makes five contributions, i.e.

- It reviews the state of the art in mobile CSCW with regard to awareness support.
- To overcome the limitations of existing approaches, it proposes a new protocol that notifies awareness information to remote clients in a multicast fashion [37].
- To increase the efficiency of this protocol, it proposes an adaptive synchronization mechanism between servers and clients, which reduces energy consumption and accelerates the delivery of awareness information. This mechanism has evolved from extensive simulation experiments and determines when and for how long a mobile client should operate in the active mode (so as to save energy) as well as when a server should notify information to clients (so as to enable synchronous collaboration among them).
- Moreover, it presents a mathematical model for dynamic bandwidth allocation, which accounts for the isochronous delivery of awareness information to all clients within the same cell, providing thus a relaxed form of WYSIWYS [34].
- Last, it demonstrates the suitability of the reaction-diffusion metaphor [32] for representing awareness in mobile settings, by describing the implementation of the protocol in terms of this metaphor.

2 REVIEW OF RELATED WORK

2.1 Basic Concepts and Terminology
In this section we present some fundamental concepts of mobile computing which are relevant to our research. Throughout our paper we assume that cooperative work is carried out across infrastructure-based networks [33], i.e. wireless networks that involve one or more servers (base stations) with fixed location, which communicate with mobile devices (clients) via radio frequencies. Each base station acts as an intermediary between clients and exchanges with them information within a subdivision of the radio-coverage area that is known as a cell.

There is a major difference between systems running over wireless networks and those running over fixed ones, in that the latter strive for location transparency while in the former location
awareness constitutes a primary objective. This, in turn, requires techniques for location management so as to track clients while they are moving. For the purpose of our research we have adopted a simple location-management technique that is described in Section 3.2. This technique is based on the assumption that, whenever a client crosses cell boundaries, the new base station provides it with an idle channel in order to continue communication. This is known in the literature as handoff and is usually implemented at the data-link layer [33].

Another important difference between wireless and fixed networks is that in the former, clients can rarely transmit information at high speeds due to the power constraints on their batteries (limited upstream capacity). Yet clients can receive information in a broadcast fashion, since base stations are allocated much more bandwidth for transmission [1]. The duration of time that clients listen on the channel to receive broadcast data is known as tuning time and it is proportional to the energy they consume. The latter, unfortunately, has not been addressed by CSCW researchers so far. Instead, existing research focuses on prototype implementations, while there exist also some organizational studies that illuminate the impact of mobility on cooperative work. The following two sections survey this research and highlight its accomplishments in awareness support.

2.2 Awareness Support in CSCW
Awareness support is a great challenge for synchronous CSCW, where interactive responsiveness is the foremost goal. This implies that awareness information must be provided at a properly fast pace to convey the current status of cooperative work, without outstripping however the collaborators’ ability to perceive it. Moreover, awareness information must be consistent across the network, something that is reflected by the WYSIWIS requirement (What You See Is What I See). Besides, awareness must often be provided independently of user requests but without causing disruption. Hence the broadcasting method that we mentioned above is inappropriate for CSCW, as the delivery of information to non-interested users would disrupt their attention and constrain thus their ability to collaborate.

There have been various attempts to meet these requirements, which have resulted in different approaches to awareness support. Systems like MetaWeb [36], for example, take a distributed approach and provide awareness via event (message) notifications. Other approaches are based on media spaces [4], collaborative virtual environments [28], spatial models [29], and the reaction-diffusion metaphor we referred to in Section 1.1. Media spaces have been proven problematic due to their discontinuous nature [30], so the increased failure rate of mobile networks would make these spaces even more problematic. Indeed, mobile users expect awareness even while disconnected else they may loose the sense of fluidity that is prevalent when collaborating over fixed networks [22].

The current uncertainty about the utility of collaborative virtual environments [30, p.286], in turn, inhibits us from considering them for mobile CSCW. With regard to the spatial models which constitute generic abstractions of these environments, they also have some limitations. The model in [29] for example, which is based on the notions of space, medium and adapter, regards the mutual awareness of an observer and an object within a space as the result of the observer’s allocation of attention (focus) and the observability of the object (nimbus). The focus and nimbus define in turn the measure of awareness, which cannot represent network disconnections and their
effects on cooperative work. Hence this model is better suited to presence awareness rather than the awareness of context (i.e. the location and state of groups and computational objects [17]).

The reaction-diffusion metaphor, instead, bypasses these limitations by modulating awareness according to the current situation and the collaborators' needs. Unlike spatial models, this metaphor provides inter-application and peripheral awareness as well as awareness of context. Specifically, the state of each entity involved in cooperative work is expressed in this metaphor by its position and motion within an awareness space (diffusion) and its interaction with other entities (reaction), as well as by its sensitivity to internal and external fields. This sensitivity can be at different degrees each time, representing thus the notion of peripheral awareness which can be hardly represented by spatial models.

The features of the reaction-diffusion metaphor have been incorporated in the so-called Model of Modulated Awareness (MoMA), in which the value of a field at each point in the awareness space is determined by a distribution function, while field-sensitivity functions determine to what fields an entity is sensitive and in what states. These functions allow representing the effects of client mobility and network disconnections, since these phenomena act as external fields to the behavior of collaborators and they allow also for more flexible representations of the focus and nimbus. The awareness space reflects the semantics of collaborative applications and, although the set of possible states must be defined in advance for each entity, the awareness space can be enhanced with extra states so as to account for client mobility and resource constraints. MoMA provides also reaction rules, which determine synchronous state changes in interacting entities and enable so the representation of the production and delivery of awareness information.

2.3 Awareness Support in Mobile CSCW

The earliest study of awareness support in mobile settings appeared in [10], where the author considered awareness in terms of connectivity information about the network and feedback on the activities of group members. One of the earliest prototypes, in turn, was StudySpace [31], which could support document sharing over mobile networks under hardware and network failures. Another early prototype was MOST [9] that could support awareness in multimedia-based communication among mobile workers.

Kristoffersen and Ljungberg [21] examined later awareness in handheld devices and concluded that direct-manipulation interfaces are not suitable for this purpose, suggesting moreover new interaction styles with little visual attention and substantial audio feedback. Another significant work on handhelds was the development of IPADs (portable Inter-Personal Awareness Devices) [16], which provided presence awareness among people situated in the same place or virtual environment. Meanwhile, the Pebbles project [25] at Carnegie Mellon had exploited PDAs to provide concurrent input to single displays, support mobile meetings, and control remotely such applications as MS-Word. Later, RoamWare [39] enabled mobile meetings via chat and e-mail messages and managed also to minimize user disruption.

The first informed study of the implications of mobility on group awareness was done by a group at Lancaster University [6], which conducted empirical evaluations of MOST and concluded that collaborators should be aware of the constraints of mobile networks so as to adjust their
behaviour accordingly. Another group at Lancaster addressed awareness with a spatial model [11], which could provide contextual awareness as well as group awareness while on the move. In a similar spirit, the Awarenex prototype [35] could provide collaborators with awareness cues for coordinating their actions across devices with diverse interface characteristics.

Over the same period a number of more elaborate systems appeared, such as NETMAN [20], DACIA [22], MOTION [19] and FUSE [17]. NETMAN is a wearable system that can support collaboration of mobile teams through audio and video conferencing over wireless channels. DACIA, in turn, adapts dynamically to user mobility and resource constraints through object relocation, allowing thus applications to change hosts at runtime in order to maintain connectivity with remote users. Associated with each application is a monitor, which gathers awareness information and relocates objects. Yet DACIA does not support presence awareness. As concerns MOTION, it provides access control, distributed data search, platform independence, presence awareness, as well as identification of data location. MOTION is based on a peer-to-peer architecture, in which some of the peers host services and others act as mere clients. Awareness support is based on event notifications, which are conveyed with SMTP messages.

All these systems share however two major drawbacks, i.e. substantial energy consumption and sensitivity to network delays in wide-area communications. Such delays can affect awareness seriously [14], so the above systems may not be always efficient. Energy, in turn, is important for ensuring the fluidity of cooperative work since it enables clients to tune into the network whenever they want. The FUSE platform does ensure fluidity to some extent, by queuing client requests when the network fails and notifying to clients previously enacted events upon reconnection. In addition to the feedback on past events, FUSE provides also presence awareness and awareness of context. A major disadvantage of FUSE though is its inability to support synchronous collaboration over long distances. Moreover, fluidity is not always guaranteed, since the feedback on past events is not effective in synchronous collaboration [17, p.184] while tuning may not be possible if clients have exhausted the power of their batteries. To provide thus continuous awareness support and ensure the sense of fluidity, we must reduce not only the notification time but also the energy consumed by the clients.

2.4 Conclusions

The above survey reveals several key requirements for awareness support in mobile CSCW, which are summarized in Table 1 below:

| i | presence (and activity) awareness of other group members |
| ii | contextual awareness that enables collaborators to arrange their behaviour according to their current location and the limitations of their environment |
| iii | minimization of user disruption |
| iv | support of heterogeneity among mobile devices |
| v | support of synchronous collaboration over long distances |
| vi | access to awareness information and shared data regardless of location |
| vii | resilience to disconnections and device constraints |
| viii | provision of the fluidity sense that is prevalent in collaboration over fixed networks |
| ix | reduction of energy consumption |
| x | minimization of notification delays |

TABLE 1

Key Requirements for Awareness Support in Mobile CSCW
Requirement [v] differs from [vi] in that the former implies the latter but not vice versa (e.g. an application that allows users to access remote information may not enhance awareness if this information is old). All these requirements are accommodated in the protocol we propose below. For example, requirement [i] is satisfied by the first two rules of the protocol which mandate that the location and predilections of each client should be forwarded to all his peers via the base stations in the intermediate cells. In the following, the term ‘client’ refers both to collaborators and the devices they use, so the reader should distinguish between them from the context.

3 A NEW PROTOCOL FOR AWARENESS SUPPORT IN MOBILE C.S.C.W.

3.1 Basic Assumptions

Given the above requirements, we observe that the last two are mutually dependent. Hence an awareness-support protocol for mobile CSCW must gain a balance between energy consumption and notification time, as the smaller the time needed to notify information the lower will be the energy consumption. On the other hand, if clients switch to the active mode frequently to refresh their awareness, they spend more energy but receive the information faster. The effectiveness of immediate notification of information has been questioned in [27], where it is claimed that if servers provide awareness information immediately after its generation, they may cause network congestion with consequent delays in awareness. Hence they suggest that the regulation of information flow should be taken over by clients. Because the latter cannot predict the arrival of this information however, proper synchronization mechanisms must be established between servers and clients so as to achieve short notification times.

Concerning requirement [iii], since broadcasting is likely to cause user disruption (as explained in Section 2.2), awareness information must be notified with multicasting at the expense of the technical difficulties associated with this method. In fact, the protocol we propose in this paper alerts users to specific messages or events according to their preferences, and notifies them to all the users synchronously. When however the users move to another cell, the messages are stored temporarily in the base stations and notified to the users upon the completion of handoff. This arrangement satisfies requirements [v] and [vi], and it is made because if the base station in the new cell has no spare channels, it will drop the handoff request and terminate the delivery of information, which is less desirable than a delayed delivery. Anyway, the user devices must be able to tune into two channels for a short period so as to communicate with both the current and the new base station (as shown in Figure 1). Since this is only possible with third-generation devices (3G) [26], our protocol can work effectively if users own such devices only.

![Fig. 1. A mobile client crosses the boundary between two cells and communicates with both servers for a short while.](image-url)
As we mentioned before, finally, the protocol conforms to the reaction-diffusion metaphor and hence it is implemented in terms of spaces, states, fields, rules, and field-specific functions (Section 5). Physical spaces consist of mobile clients, base stations, and cells, while awareness spaces consist of interacting entities and various fields.

3.2 Protocol Rules

Table 2 presents an informal description of the protocol rules, whereas a formal description in terms of the reaction-diffusion metaphor is given in Section 5 along with implementation details.

| TABLE 2 |
| Protocol for Efficient Awareness Support in Mobile CSCW |

1. Whenever a client wishes to express predilection for some events, s/he must register a relevant interest with the server invigilating her/his own cell. The server broadcasts this to the other servers within its reach so as to make them aware of the location and predilection of that client (provided of course that the servers maintain location databases for this purpose). This also aims to inform interested clients of the actions of their peers whenever the latter are located several cells apart.

2. When a server receives awareness information that is intended for a client in another cell, it forwards this information to the recipient through the servers in the intermediate cells. If, instead, this information is intended for a client within its own cell, the server starts emitting alert signals at the frequency of that client. In both cases, each server creates first a time-stamped copy before forwarding the information to another recipient.

3. If the information interests multiple clients, the server must start emitting alert signals in a multicast fashion. Multicasting is repeated periodically until acknowledgements are received from all clients. If however no acknowledgement is received from a client after a number of alert signals (which we specify in Section 3.4), the server realizes that some disconnection has occurred and pauses temporarily the signal emission. If this disconnection is due to the client’s relocation to another cell, the new server informs accordingly the previous one and the flow of execution jumps automatically to Rule 8. If this is not the case though, when another message (or event) must be notified later to the same client, the server hooks it to the previous one and notifies both of them as a single message.

4. Alternatively, instead of emitting alert signals the server may await a client to switch to the active mode and, upon receiving from it (i.e. the client) a signal that denotes this, start immediately the notification of information. In that case, each client (i.e. mobile device) must tune into the network periodically. If a client does not receive anything within a period $S_t$ (whose value is also determined in Section 3.4), it must send a query to the server and a relevant message appear on its interface.

5. If there exist undelivered messages for a client (due to a past disconnection), the server notifies them immediately according to Rule 6 below else it replies to the client’s query with a negative message, which appears on the client’s interface too. If no reply is sent, the owner of the device must switch to the sleep mode and move to another cell.

6. Upon realizing that a client has switched to the active mode, the server starts multicasting the awareness information, whose end is denoted by an End-of-Message flag. Each client remains in the active mode throughout multicasting and, even if one client changes cell, the notification is not interrupted since that client will be able to tune into both frequencies for a short period.

7. After seeing the End-of-Message flag, a client must send an acknowledgement to the server so that the latter can remove the message’s copy from its memory. The clients remain in the active mode for one tuning period after the acknowledgement, and then they switch to the sleep mode. After one sleep period, they tune into the channel so as to receive new messages. If they want to retrieve data though, they must remain active and switch to the sleep mode only after the server(s) have responded to their queries. Data are disseminated with multicasting, like the messages. Moreover, clients must remain active after the receipt of the End-of-Message flag in order to ensure that consecutive messages separated by a small delay are notified as if they were one continuous message.

8. If a client has moved to another cell and the handoff there has been done, the new server informs the previous one accordingly and requests the undelivered information in order to forward it to its new guest (after synchronizing with it of course). Upon receiving the message’s copy, the new server forwards it to the client according to Rule 3 or 4 by reserving first the necessary bandwidth.

Figure 2 below illustrates schematically how awareness is provided according to these rules:
3.3 Discussion

The protocol rules in Table 2 satisfy most of the requirements of Section 2.4. Indeed, let us suppose that several users who are located in different cells want to establish an ad-hoc meeting while they are moving. Systems like RoamWare, FUSE, and Awarenex do not afford this capability, since they can only locate mobile devices within small areas. The assumptions we have made in Sections 2.1 and 3.1, instead, extend beyond the frontiers of a single cell and are meant to enable awareness in a way similar (but not identical) to data broadcasting across several cells. This is why we assumed that servers multicast information instead of assuming a discovery mechanism in each client, something that would confine collaboration within small areas. No one else has proposed something similar so far, and hence very few CSCW systems satisfy currently requirement [v] (i.e. DACIA that allows object relocation at run time and, to some extent, MOTION as well). Other systems, like FUSE, which provide access to Web servers, do not enable synchronous collaboration in that case.

We must point out nevertheless that the aforementioned discovery mechanisms aim to facilitate client-to-client communication, which is used in synchronous CSCW to increase responsiveness [17, p.174]. Extending collaboration towards large areas will increase communication delays but, as we show in Sections 3.4, 3.5 and 4.2, this increase can be minimal if clients are properly synchronized with the servers and the available bandwidth is allocated to them fairly. In this regard, Rule 1 of our protocol does allow synchronous collaboration across large areas (i.e. it satisfies requirement [v]) by achieving a responsiveness comparable to the responsiveness of client-to-client communications within a single cell. It also satisfies requirement [i] with regard to presence awareness by broadcasting location information, as well as requirement [vi] since an enacted event can be notified to every interested client once her/his location is known.

Requirements [i], [v] and [vi] are satisfied by Rule 2 as well, which allows the notification of messages (or events) to remote clients, while the creation of timestamped copies guarantees that these messages will be sent to the recipients anyway, regardless of temporary disconnections in the network (hence requirement [vii] is also satisfied). Considering again the example with the ad-hoc meeting, the awareness information that Rule 2 refers to may be a user’s request to join the...
meeting or a message that s/he is leaving it. That message may not reach all the recipients instantly, so if some of them are located several cells apart, the message will need to hop through the servers in the intermediate cells. If there is a disconnection between a server and a client, the message will be buffered in the sender until communication is re-established. This is in sharp contrast with the migration of collaborative sessions that FUSE performs when users change location [17, p.171]. Obviously, this migration must be costly and inefficient when several users change location simultaneously.

Moreover, as we mentioned earlier, the protocol minimizes user disruption with multicasting, which is mandated by Rule 3. Hence this rule satisfies requirement [iii], while the acknowledgement of alert signals guarantees that no notification will be lost due to a disconnection (so requirement [vii] is also satisfied by that rule). No other system allows acknowledgements at the moment, since most of them are intended for short-range communications where disconnections are rare, and thus acknowledgements are not needed. Rule 4, in turn, is an alternative to Rule 3 and follows the authors' suggestion in [27] to let the clients schedule the notifications by themselves so as to reduce the likelihood of congestion. The latter complies with requirement [x], as congestions incur delays. Returning again to the ad-hoc meeting example, Rule 4 entails that a user may learn that another one has joined the meeting only after his device has switched on and not immediately after the expedition of the relevant message by the other user. We note also that none of the current systems forces user devices to switch between modes, because none of them aims to reduce energy consumption. Whenever a message is sent it is captured immediately by its recipient, since his/her device operates continuously in the active mode during cooperative work.

Rule 5 continues Rule 4 and it is the only one in our protocol that requires user intervention. The ability to intervene satisfies requirement [ii], since users can adapt their behavior to the limitations of the environment. Concerning Rule 6, it is meant to reduce energy consumption (i.e. satisfy requirement [ix]) and provide also a sense of fluidity (requirement [viii]). With regard to the latter, no one from the existing systems incorporates handoff with CSCW. FUSE and DACIA try to provide the fluidity sense by relocating objects at run time; thereby FUSE manages to provide fluidity over wireless LANs only [17, p.178], while the extent to which DACIA manages this is not exactly known.

Rule 7 also satisfies requirement [viii], by keeping clients active for one tuning period after an acknowledgement as well as after the End-of-Message flags (in fact, no re-synchronization is needed in these cases). Moreover, this rule satisfies requirement [vi] with respect to shared data. This is very significant, since data retrieval from shared workspaces often occurs simultaneously with notifications, so the protocols that support awareness must account for data sharing as well. Rule 8, finally, complements Rules 3 and 6 and satisfies requirements [v], [vi], and [viii].

It is evident therefore that most of the requirements of Section 2.4 are indeed satisfied. Requirement [iv] is implementation-dependent so we discuss it in Section 5. As concerns requirements [ix] and [x] that determine the efficiency of our protocol, their satisfaction is implied by Rules 3 through 7. Because this is not so evident from the above discussion however, we present below a detailed analysis that supports this claim by determining when and for how long a client should remain in the active mode, as well as when a server should multicast information to clients.
3.4 Synchronizing Servers with Clients

Minimizing energy consumption can only be possible if servers and clients operate synchronously. The protocol we described earlier involves two types of synchronization between servers and clients, which are mandated by Rules 3 and 4. Since the notification of events (or messages) must be time- and energy-efficient, the emission of alert signals seems superior at first glance. In fact, events are notified immediately upon their occurrence, while the energy consumption is reduced since clients tune into the network only at certain points in time. The other approach (where the servers save the events temporarily in their memory until the clients wake up) incurs higher energy consumption, since a client may tune repeatedly without receiving any event (i.e. if no event has been saved by the server in the meantime). It would be advantageous therefore if clients had the ability to predict event arrivals and tune into the network at the right moment but, since cooperative work is usually opportunistic [3], this seems impossible.

On the other hand, the superiority of the first approach with regard to time-efficiency should not be taken as granted, given the narrow bandwidth of wireless networks and the increased likelihood of congestion (indeed, the pace of CSCW may be such that the immediate notification of messages may compound congestion and increase so delays [27]). We need thus to investigate whether there are situations where the second approach outperforms the first in terms of time-efficiency (i.e. notification time). Since this approach requires some sort of synchronization between servers and clients, we examine below appropriate mechanisms for this purpose. In Section 4.2, we compare these mechanisms against active notification. As an aside, we note that these mechanisms are applicable to certain existing systems too (e.g. FUSE and RoamWare), provided that the latter could force user devices to switch automatically between modes.

We assume in principle that clients tune into the network periodically. To appreciate the advantages of periodic tuning, consider the tuning patterns of Figure 3. The solid intervals in this figure represent tuning periods, while the spaces between them represent sleep periods. The thin vertical lines denote the time points at which events arrive at the servers, while the bold ones denote the time points at which these events are notified to the clients. The very thick vertical line at the bottom, finally, denotes the simultaneous notification of two events.

Fig. 3. Different tuning patterns.

Periodic tuning allows for small delays in notification and results in a moderate consumption of energy (which, in turn, will allow clients to keep collaborating with each other for long periods). Exponentially decaying tuning, on the other hand, incurs long delays if the notification is late relative to the arrival of the event. In Figure 3, in fact, the second and the third event are
notified with substantial delay, so clients may have performed several activities in the meantime. In general, if we assume that each tuning period lasts $t_c$ time units and that the first sleep period lasts $t_p$ units while the tuning frequency decays at a rate of $e^{-a}$ (where $a \geq 0$), then the notification of the first event will occur after $\lambda \cdot t_c + t_p \cdot \int_{0}^{\lambda-1} e^{a} \cdot da$ time units (where $\lambda$ denotes the number of tuning trials before notification occurs). In periodic tuning instead, the notification of the first event would occur after $\lambda \cdot t_c + t_p \cdot (\lambda - 1)$ time units. Finally, a tuning frequency that grows exponentially may decrease the notification time, but if an event arrives after several tuning trials, it will incur higher energy consumption.

Periodic tuning is thus superior to the other patterns because it achieves a better balance between notification time and energy consumption. Still there are several variations of periodic tuning, such as those shown in Figure 4:

![Fig. 4. Variations of periodic tuning.](image)

Part (a) of this figure shows a variation where the tuning and sleep periods are equal in length, whereas in part (b) the sleep periods are longer. Parts (c) and (d) in turn show variations in which the length of the tuning period is constant but the sleep period varies according to a Gaussian and an exponential distribution, respectively. These distributions are repeated periodically, so for example if each part of Figure 4 corresponds to 100 time units, the shape of the Gauss distribution reappears after 44 units while the shape of the exponential reappears after 22 units.

Nevertheless, both the tuning and sleep periods must be such so as to reduce the consumption of energy and the notification time. There are again various ways to do this, as we show below.

### 3.4.1 Mathematical Analysis of Periodic Tuning

Assuming a discrete-time model, let’s consider a client who tunes into the network every $t_p$ time units and remains active for $t_c$ units. The following cases are then possible, as shown in Figure 5:

![Fig. 5. Various synchronization patterns with constant sleep periods.](image)
[1st] $t_c \geq t_p$, which corresponds to parts (a) and (c) of the above figure. The proportion of time that a client remains in the active mode is in this case $\frac{t_c}{t_c + t_p}$, which is greater than or equal to $\frac{1}{2}$, entailing therefore significant energy consumption. Moreover, the smaller the $t_p$ the bigger will be the energy consumption. Indeed, if we denote by $X$ the time between two successive notifications and by $f$ the energy consumed by an active client in one time unit, then the total energy consumption will be $E_0 = \lambda \ast t_c \ast f$, where $\lambda$ represents the number of tunings in this period, i.e.

$$
\lambda = \left\lceil \frac{X}{t_c + t_p} \right\rceil \text{ or } \lambda = \text{Int} \left( \frac{X}{t_c + t_p} \right) + 1
$$

Since $t_p$ is in the denominator, it is inversely proportional to the consumption of energy $E_0$. The value of $t_c$, instead, affects energy consumption differently. Indeed, if we analyze further the product denoted by $E_0$ we have

$$
E_0 = \left\lceil \frac{X}{t_c + t_p} \right\rceil \ast t_c \ast f \approx \left( \frac{t_c}{t_c + t_p} \right) \ast X \ast f
$$

Since $t_c$ is in both the numerator and the denominator, it does not affect $E_0$ so much as $t_p$. Instead, $E_0$ is affected more by the ratio $\frac{t_c}{t_c + t_p}$ which can only be decreased if $t_p$ is substantially larger than $t_c$.

[2nd] $t_c < t_p$, which corresponds to parts (b) and (d) of Figure 5. The proportion of time that a client is in the active mode is less than $\frac{1}{2}$ in this case, entailing therefore a lower consumption of energy. Yet the delay in notification may be significant if an event arrives after the client has switched to the sleep mode.

In both cases, the smaller the $t_p$ the faster will be the notification. In fact, if an event’s arrival coincides with a tuning period, then the notification will take $Tr$ time units, where $Tr$ denotes the time required for the transmission of the event via the wireless channel. If instead an event’s arrival coincides with a sleep period, then at most $t_p$ time units will elapse until the clients switches to the active mode, so the notification delay will be at most $t_p + Tr$. We are facing hence a problem with contradictory objectives, since the value of $t_c$ can increase energy consumption and reduce simultaneously the notification time (or vice versa). Unfortunately, we cannot solve this problem with existing methods of operations research, because the value of $X$ changes dynamically.

Besides this difficulty, there is also a difficulty with parts (c) and (d) of Figure 4 which are not represented in Figure 5 but constitute possible alternatives for the synchronization of servers with clients: the conclusions we derived above (namely that $t_c$ does not affect substantially energy consumption and $t_p$ is proportional to notification time and inversely proportional to energy consumption) also hold in these parts, but they do not indicate whether the repetitive tuning patterns with the random sleep periods are superior to the patterns with constant sleep periods that we analyzed above. In fact, whether a Gaussian or an exponential distribution of sleep periods can further reduce the energy consumption and the notification time is an issue that
requires extensive probabilistic analysis, which would only be valid if we could model client activities with a statistical distribution. Given the opportunistic nature of CSCW, however, this would be risky and error prone.

To get around the above difficulties, we performed simulation experiments whereby we tested various combinations of parameter values for the synchronization process, something that could not be done with pure mathematical analysis. Moreover, we contrasted the results obtained from these combinations against the approach with the alert signals, using different collaborative scenarios so as to measure their effects on network congestion. There were though two parameters that were treated analytically before running the experiments. One was \( S_t \) (referred to in Rule 4 of Table 2), which was given a constant value since it cannot exceed \( \max(t_p) \) if a client remains in the same cell. If a client changes cell, then \( S_t \) can become at most \( \max(t_p) + H_t \), where \( H_t \) denotes the time needed by the new base station to assign a band of frequencies to that client. The other parameter was \( T_r \), which contributes to the notification time. This parameter is inversely proportional to bandwidth, as it is suggested by Shannon’s law [33, p.27].

By allocating therefore the bandwidth dynamically, we can configure \( T_r \) (and consequently the notification time as well) according to the needs of clients. It is important to note that dynamic bandwidth allocation emerged as an option due to multicasting, since broadcasting would notify events to everyone and thus the bandwidth would be allocated up front in a static manner.

3.5 Dynamic Bandwidth Allocation

In this paper we view dynamic bandwidth allocation as an optimization problem, because we aim to minimize by it the transmission time \( T_r \). The relation between bandwidth and transmission time is indicated by a fundamental law of communications, according to which the speed of transmission is directly proportional to the band of frequencies [33, p.17]. For the purpose of our research, we express this analogy with the formula

\[
B_{u_k} = \frac{y}{T_{r_k} \cdot w} \quad \forall k \in \{1, 2, \ldots, N\}
\]

where \( B_{u_k} \) denotes the bandwidth utilized to transmit to client \( k \), \( y \) denotes the size of transmitted information in Bytes, and \( w \) is a symbol used to associate the bandwidth with the transmission speed in Shannon’s law. Our aim is to minimize the time for notifying awareness information, which in turn depends on the minimization of the transmission time, i.e.

\[
\min(T_{r_k}) = \min\left(\frac{y}{w \cdot B_{u_k}}\right) \quad \forall k \in \{1, 2, \ldots, N\}
\]

The above objective is subject to the constraints \( B_{u_k} > 0 \ \forall k \in \{1, 2, \ldots, N\} \) and \( \sum_{k=1}^{N} B_{u_k} \leq \mathcal{B} \), where \( \mathcal{B} \) denotes the total available bandwidth. By computing therefore optimal values for each one of the \( B_{u_k}’s \), we can minimize \( \frac{y}{w \cdot B_{u_k}} \) or, equivalently, maximize \( \frac{w \cdot B_{u_k}}{y} \) subject to the same constraints. An easy solution is to assign the value \( \frac{\mathcal{B}}{N} \) to each \( B_{u_k} \) but this solution is
by no means optimal. Consider for example two clients who register their interest in the same event, but when this event is being notified one of the clients is also retrieving data. That client will get the event later than the other and thus WYSIWIS will be violated. Hence, in addition to notifications, we must allocate bandwidth for data sharing as well. Since these operations occur in parallel oftentimes, the bandwidth must be allocated to them evenly. If it suffices for both, then the notification and retrieval times will be minimal, else the bandwidth must be allocated in a different manner.

3.5.1 Quantifying Bandwidth Allocation

Let’s denote by \( s_n \) the size of information retrieved by client \( n \) (where \( 1 \leq n \leq M \)), by \( I_{rt_n} \) the time required for this retrieval, and by \( B_{bu_n} \) the utilized bandwidth. Then \( B_{bu_n} = \frac{s_n}{w \cdot I_{rt_n}} \) or \( I_{rt_n} = \frac{s_n}{w \cdot B_{bu_n}} \). We wish to minimize \( I_{rt_n} \) \( \forall n \in \{1, 2, \ldots, M\} \), bearing in mind that we wish also to minimize \( T_{rk} \) \( \forall k \in \{1, 2, \ldots, N\} \). These minimizations are equivalent to maximizing \( B_{bu_n} \) and \( B_{uk} \) \( \forall n \in \{1, 2, \ldots, M\} \) and \( \forall k \in \{1, 2, \ldots, N\} \) subject to the constraint

\[
\sum_{k=1}^{N} B_{uk} + \sum_{n=1}^{M} B_{bu_n} \leq TAB
\]

where \( TAB \) denotes the total available bandwidth. The above problem can be simplified by imposing a “fairness” constraint on the retrieval and notification times. In fact, collaborators may opt for faster awareness than data sharing in some applications, so the server can assign relative weights to the bandwidth allocated to each operation as follows: if the desired relation between the retrieval and notification times is for example \( T_{rk} \cdot y_1 \leq I_{rt_n} \leq T_{rk} \cdot y_2 \) for any clients \( k \) and \( n \) (where \( y_1 \) and \( y_2 \) are close to 1), and the size of the data requested by client \( n \) is \( s_n \) while the size of the awareness information notified in parallel with that data is \( y \), then the allocated bandwidth must satisfy the relations \( B_{bu_n} = \frac{s_n}{w \cdot I_{rt_n}} \) \( \forall n \in \{1, 2, \ldots, M\} \) and \( B_{uk} = \frac{y}{T_{rk} \cdot w} \) \( \forall k \in \{1, 2, \ldots, N\} \), as well as the relation

\[
\frac{s_n}{y \cdot y_2} \leq \frac{B_{bu_n}}{B_{uk}} = \frac{s_n \cdot T_{rk} \cdot w}{y \cdot I_{rt_n} \cdot w} \leq \frac{s_n \cdot y_1}{y} \forall k \in \{1, 2, \ldots, N\} \text{ and } \forall n \in \{1, 2, \ldots, M\}.
\]

Based on the last relation, the constraint of the optimization problem \( \max(B_{uk}) \) \( \forall k \in \{1, 2, \ldots, N\} \) becomes

\[
\sum_{n=1}^{M} B_{un} \cdot s_n \cdot \frac{y_1}{y} + \sum_{k=1}^{N} B_{uk} \leq TAB
\]

Since the bandwidth for notifications must be the same for all clients, the required maximum is

\[
\frac{TAB}{N + \left(\frac{y_1}{y}\right) \cdot \sum_{n=1}^{M} s_n}
\]
∀ k ∈ \{1, 2, \ldots, N\} (where \[\lfloor \cdot \rfloor\] denotes truncation). The value of γ₁ can be revised over time of course, depending on the needs of the clients. This agrees with the authors’ suggestion in [27] to control the delivery of awareness information at the application level and, as we show in Section 4.2, it can exert an impact on the protocol’s efficiency.

Concerning data retrieval, in turn, the maxima \(\max(Bbu_n)\) subject to the constraint

\[
\sum_{k=1}^{N} Bbu_k \times \left( \frac{s_k}{y \times y_2} \right) + \sum_{n=1}^{M} Bbu_n \leq TAB
\]

are not always the same for every client. This problem can be simplified as before if we assume a “uniform rate of retrieval”, which implies that larger quantities of data must incur longer retrieval times. For example, if each pair \((Bbu_k, Bbu_n)\) satisfies the relation \(\frac{Bbu_k}{Bbu_n} = \gamma_3 \times \left( \frac{s_k}{s_n} \right)\), where \(\gamma_3\) is close to 1, then each \(Bbu_k\) can be expressed in terms of \(Bbu_1\) since all the \(s_k’s\) are already known. Moreover, there will be \(M\) different \(\gamma_3’s\) in that case, namely \(\gamma_{31}, \gamma_{32}, \ldots, \gamma_{3M}\). \(\gamma_{32}\), for example, will be equal to \(\frac{Bbu_{32} \times s_1}{Bbu_{2} \times s_2}\). Under this assumption, the constraint of the optimization problem \(\max(Bbu_i)\) becomes

\[
\sum_{k=1}^{N} Bbu_i \times \left( \frac{\gamma_{3k} \times s_k}{y \times y_2} \right) + \sum_{n=1}^{M} Bbu_i \times \gamma_{3n} \leq TAB
\]

and the obvious solution is \(Bbu_i = \frac{\frac{\gamma_{3i} \times s_i}{y \times y_2} \times \sum_{k=1}^{N} \gamma_{3k} \times s_k + \sum_{n=1}^{M} \gamma_{3n}}{\sum_{n=1}^{N} \gamma_{3k} \times s_k + \sum_{n=1}^{M} \gamma_{3n}}\).

The above analysis implies that if a client has requested data of size \(s_i\) but is also interested in awareness information of size \(y_i\), then the bandwidth that must be allocated to him is

\[
Bbu_i = \frac{\frac{TAB}{\sum_{n=1}^{M} s_n \times \left( \frac{Y_1}{y} \right) + N}}{\left( \sum_{k=1}^{N} \frac{\gamma_{3k} \times s_k}{y \times y_2} \right) + \sum_{n=1}^{M} \gamma_{3n}} \times \left( \frac{\gamma_{3i} \times s_i}{s_1} \right)
\]

4 EVALUATION OF THE PROTOCOL
In Section 3.4 we questioned the suitability of active notifications because of the increased likelihood of congestion and the ensuing delays in cooperative work. For this reason we examined various synchronization mechanisms, which can reduce congestion by allowing clients to tune into the network at different time periods. Yet these mechanisms are not always amenable to mathematical analysis. To test therefore their efficiency and determine whether they prevail over active notification in real collaborative situations, we performed several simulation experiments. In
addition to energy consumption and notification time, we tested each mechanism also with respect to its scalability over various client populations. The simulation results provided evidence that these mechanisms scale well over populations of medium size and that they prevail over active notification in certain situations. Our experiments were run using the Arena simulation package [18] and were based on the simulation model below.

4.1 Simulation Model

Like the protocol we proposed in Section 3.2, the simulation model we built adopts the notification style of [36] while it comprises six components, i.e. shared data, servers, cells, producer and consumer clients, as well as a wireless network. The values of the simulation parameters were drawn either from the literature (e.g. [33] for handoff time and [40] for client mobility) or from observations of cooperative tasks, and they are presented in Table 3 below:

<table>
<thead>
<tr>
<th>TABLE 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simulation Parameters</td>
</tr>
</tbody>
</table>

- **Distributions of event generation by producer clients**
  - Poisson
  - Gauss
  - None (i.e. real scenarios)

- **Mean of event generation**
  - 1 event every 3 to 60 seconds

- **Value of parameter w in Shannon’s law**
  - 2

- **Number of “producer” clients**
  - 1 - 10 (in the random generation of events)
  - 3 - 10 (in the real scenarios)

- **Distributions of sleep periods**
  - Uniform
  - Periodic Gaussian
  - Periodic Exponential

- **Mean of sleep periods**
  - 1 - 9 seconds

- **Number of “consumer” clients**
  - 2 - 15

- **Length of tuning intervals**
  - 1 - 5 seconds

- **Available bandwidth**
  - 38.4 KHz - 4.0 MHz

- **Distribution of data requests by consumer clients**
  - Poisson
  - None (i.e. real scenarios)

- **Mean of data requests**
  - 1 request every 10 seconds to 2 minutes

- **Power consumption by each client**
  - In the sleep mode
  - 0.00005 Watt
  - When tuned into the network
  - 1 Watt

- **Service rate of client queries**
  - 1 query per second

- **Mean of client queries**
  - 1.2 second

- **Tuning patterns during synchronization**
  - Uniform
  - Periodic Gaussian
  - Periodic Exponential
  - Non-periodic (Adaptive)

- **Bandwidth-allocation coefficients**
  - \( \gamma_1 \)
  - \( \gamma_2 \)
  - \( \gamma_3 \)

- **Range of values**
  - 1.0 - 1.5

- **Size of object store**
  - 5 - 500 MBytes

- **Client-mobility distribution**
  - Exponential

- **Mean of mobility distribution**
  - 1 movement every 1 to 3 minutes

- **Distribution of failure probability of the network**
  - Poisson

- **Mean of failure distribution**
  - 1 failure every 30 to 60 seconds

- **Failure duration**
  - 1 second

- **Wireless links**
  - Upstream Capacity
  - 19.2 Kbps
  - Downstream Capacity
  - 2.0 Mbps

- **Transmission speed of each link**
  - 20 events per second

The shared data component, for example, was modelled by a set of objects whose size varied from 5 to 500 MBytes (representing such tasks as cooperative editing and drawing). Associated with each server were three queues, i.e. an event (message) queue, a notification queue, and a query queue. Whenever an event was generated by a producer client, it was joining the event queue whose average waiting time was the time needed by the client to send the event plus the time needed by the server to create a timestamped copy. Once an event was becoming available for notification it was joining the notification queue, in which the average waiting time was the total notification time. The query queue, finally, was formed by the data-retrieval requests of the
consumer clients and was modeled by a Poisson distribution with mean varying from 1 query every 10 seconds to 1 every 2 minutes.

To the best of our knowledge, there is not any published work that describes CSCW with statistical distributions nor are there any measurements that could be used for the establishment of such distributions. Only the work of Baeza-Yates and Pino [2] suggests the use of Poisson for modelling cooperative tasks, and thus we incorporated it into our experiments. Specifically, we formed a distribution by recording the intervals between successive requests of graphical objects by 15 clients in a drawing session. Using then the $A^2$ statistical test [8] with significance level of 5%, we tested and verified the hypothesis that the above distribution is Poisson with mean 10 seconds. By repeating this test with various numbers of clients, we verified the same hypothesis with a different mean each time (e.g. for 2 clients the mean was 119.6 seconds).

As concerns the producer-client component, it was modelled by a different number of processes in each experiment (ranging from 1 to 10), which were sending events to the server asynchronously. Specifically, we represented client events with a Poisson and a Gauss distribution, while we used also three real cooperative scenarios in which the generation of events did not conform to any known distribution. The Poisson distribution was used for the reason we mentioned above. The use of Gauss, in turn, was motivated by the assumption that when the user population is sufficiently large, their activities are likely to conform to a Gauss distribution (following the central-limit theorem, as suggested by one of the reviewers). As concerns the real scenarios, finally, they included the drawing of an automobile (which we referred to above), the concurrent editing of an ASCII document, and the accomplishment of a mechanical task by remote workers. The events in the first two scenarios included the creation, deletion, and copy of figures and text, while in the third scenario they included the exchange of SMS messages and JPEG images.

The consumer-client component in turn was modelled by a number of processes that varied between 2 and 15, since, according to our experience such groups usually are engaged in cooperative tasks. The energy consumption over the tuning periods was calculated by the formula

\[
\text{ENERGY (measured in Joules)} = \text{POWER (measured in Watts)} \times \text{TIME (measured in seconds)}
\]

The power consumption per second was 1 Watt in the active and 0.0005 Watt in the sleep mode, matching thus typical device specifications. The servers and the clients communicated via the wireless network component, which was modelled by a failure-prone link that was carrying information in both directions. As shown in Table 3, such failures were occurring according to a Poisson distribution with mean ranging between 30 and 60 seconds [5], while the duration of each failure was 1 second. In the notifications coinciding with a failure, however, the overall delay was 2 seconds. Following Rule 3 of the proposed protocol (Section 3.2), we accompanied each failure with client relocations which were modelled by a uniform distribution implying either 0 or 1 relocation every 10 failures. Concerning the speeds of the uplink and downlink channels, they were 19.2 Kbps and 2Mbps, while the corresponding bandwidth was 38.4 KHz and 4 MHz, respectively, implying thus the value 2 for the $w$ parameter (Section 3.5).

The cells’ component, finally, was used to represent client mobility. We considered three cells, each one with its own server. All the clients were located initially in one cell but they started moving
progressively to the others according to an exponential distribution with mean ranging from 0.33 to 1 (i.e. 1 to 3 client relocations every 3 minutes). The adoption of the exponential distribution in this case was based upon a relevant model in the literature [40]. Moreover, we assumed that the handoff time during relocations was between 0.1 and 0.3 sec [33].

4.2 Simulation Results

Our experiments considered both the case of active notification and the case of synchronization between clients and servers. Each case was divided into two parts, namely $A$ in which clients were not changing cells (hence network failures were not causing client relocations), and $B$ in which clients were changing cells. Each part, in turn, involved five types of event generation, namely one corresponding to Poisson (type 1), another corresponding to Gauss (type 2), and another three corresponding to the real scenarios (i.e. type 3 corresponding to the cooperative drawing, type 4 to the document’s editing, and type 5 to the mechanical task). So there were totally 20 types, in which the mean of each distribution varied within the range shown in Table 3. Due to space limitations we do not present schematically the results for every type, but we comment on the variations we observed in the corresponding experiments.

4.2.1 Experiments $A_1$

Figures 6a and 6b show the average notification time $ant$ per client in seconds (blue curves) and the average energy consumption $aec$ per client in Joules (yellow curves), as they were computed by the simulator after 12 minutes of experimentation. Figure 6a shows the results of active notification and Figure 6b the results of the synchronization approach. It is evident from these figures that active notification prevails clearly over the other approach in both respects.

Concerning Figure 6b, the tuning pattern that conforms to the periodically repeating Gauss distribution (PGD) (which we showed in Figure 4) prevails in terms of energy consumption over the uniform distribution and the periodically repeating exponential (PED). This phenomenon persisted as the distributions’ means were increasing and it was independent of the number of clients. The average notification time, in turn, was nearly the same for all the distributions while its variance (that is not shown in Figure 6b) ranged from 0.93 to 2.11 seconds.

![Fig. 6(a). Performance of the proposed protocol using active notification.](image1)

![Fig. 6(b). Performance of the proposed protocol using synchronization.](image2)
Regarding the effects of bandwidth allocation, we observed that notifications were isochronous to six clients in the first experiment and varied slightly in the others. Finally, the conclusion we derived with regard to scalability was that the three distributions scale linearly with respect to notification time, but with respect to energy consumption the uniform distribution seems to be independent of client populations.

4.2.2 Experiments A2

The experiments with the Gauss distribution of event generation yielded different results, as the variations we observed in the notification time among different populations of clients were sharper than those in the Poisson distribution. Still the superiority of active notification was again evident in both respects, and its scalability was linear as the number of clients increased.

In general, the results we obtained indicated that [1st] the active notification of events is far better than synchronization in both respects (i.e. energy consumption and notification time) and also scales linearly over large populations of clients, [2nd] in the approach with periodic tuning the scalability of all the distributions is linear over medium-size populations with respect to notification time, and [3rd] the PGD outperforms the other distributions in energy consumption except when the sleep period is very large. These observations suggest that for medium-size groups the PGD is more suitable than the other tuning distributions but inferior to active notification. Based on this conclusion, we tested the suitability of the synchronization approach in the real scenarios using only PGDs, whose means varied from 1 to 5 seconds.

4.2.3 Experiments A3 - A5

The average energy consumption in the real scenarios exhibited in both approaches a shape similar to the shape of PGD in Figure 6b. Still the superiority of active notification was not so evident this time, since the differences with the PGD were in most experiments lower than 10%. The average notification time of PGD, in turn, did not increase linearly with the length of the sleep period but the simulator produced a fractal shape, due to the fact that the events had formed a mix of distributions with a different mean each. Moreover, the deviations from each mean were higher this time, indicating thus a violation of WYSIWIS.

What stroke to us however was that active notification was proven inferior to synchronization in these experiments. To reduce the large deviations we repeated the experiments but, instead of a PGD, we considered this time a synchronization approach whose sleep periods were changing dynamically based on previous values of notification time. More specifically, whenever a client was switching to the sleep mode, the length of the sleep period was changing on-line according to the following algorithm (Algorithm-1): if there were \( k \) clients \( cc_1, cc_2, \ldots, cc_k \) whose recent notification times were \( nt_1, nt_2, \ldots, nt_k \), respectively, while there had been before another \( Y \) notifications, and the initial sleep period was \( t_{p0} \), then if \( \text{var}_Y > \text{var}_{Y-1} \) the new sleep period \( t_{p(Y+1)} \) was given the value \( t_{pY} - \frac{\left| t_{pY} - t_{p(Y-1)} \right|}{2} \) (where \( | | \) denotes absolute value and \( \text{var}_Y \) is the variance of the notification times at step \( Y \)). Otherwise, \( t_{p(Y+1)} \) was assigned the value of \( t_{pY} \). For
example, if the sleep period was 2 seconds at step $Y$ and 3 seconds at step $Y-1$, at step $Y+1$ it should become 1.5 second.

The first experiment with this algorithm run the cooperative design scenario and lasted 12 minutes, while most of its parameters were the same as in the experiment of Figure 6b except the average sleep period and the mean of data requests. The notification times of this algorithm were lower than the times of the PGD and exhibited also substantially smaller variance than before. As concerns energy consumption, it was higher with this algorithm than it was with the PGD in the previous experiments. Figure 7 below shows the results of the last experiment, where the dotted lines represent the new synchronization algorithm and the continuous ones the active notification (which was also included in this experiment so as to enable comparison).

![Figure 7](image.png)

**Fig. 7.** Performance of the two approaches when the second involves adaptive tuning.

Overall, we concluded that as the number of clients increases, adaptive synchronization yields shorter times than active notification and preserves WYSIWIS better than any tuning mechanism conforming to a specific distribution. Whether we can further reduce the notification time by another formula for $t_{pY}$ than the formula of Algorithm-1, remains however an open issue.

### 4.2.4 Experiments B1 - B5

The experiments involving client relocations revealed significant differences in notification time due to the handoff requests, and the differences were even more significant in the consumption of energy. Yet the consumption incurred by the PGD was lower this time compared to Algorithm-1, while the notification time was almost the same in all cases.

This prompted us to trace down user actions and detect a strange phenomenon owing to the network failures. These failures forced clients to move quite often to another cell, but in some cases there was another failure in the new cell which was causing additional delays. Because clients were operating in the sleep mode during relocation, they were not consuming energy. When we modified the value of the failure rate and repeated the experiments, we observed higher energy consumption and lower notification times. We made therefore the conclusion that the on-line algorithm we presented earlier should have considered not only the variance in notification time but also the variance in energy consumption.

Thus we repeated the experiments with the real scenarios (B3 - B5) by transforming Algorithm-1
as follows (Algorithm-2): if \( \var{Y} > \var{Y-1} \) and the relative reduction in notification time was greater than the relative increase in energy consumption, i.e.

\[
\frac{\mu_Y - \mu_{Y-1}}{\mu_{Y-1}} > \frac{E_Y - E_{Y-1}}{E_{Y-1}}
\]

where \( \mu_Y \) denotes the average notification time in step \( Y \) and \( E_Y \) the average energy consumption in that step, then the sleep period was changing according to Algorithm-1 else the sleep period was conforming after step \( Y \) to a PGD with mean \( t_{pY} \). After a time period \( z \) these two conditions were tested again in order to derive a new value for \( t_p \). We run the modified experiments several times with different client populations and \( z \) values, and we observed small improvements in the notification time but significantly lower values of energy consumption. Hence we concluded that the savings in energy obtained by the PGD could not be obtained otherwise. The combination of this distribution with Algorithm-2 achieves therefore a better balance between energy consumption and notification time when the pace of collaboration is fast. When this pace is slow active notification performs better, but since there are considerable time gaps between successive events in that case, the gains in time and energy do not improve awareness substantially.

4.3 Summary of Results
To gain insight from the above results into the efficiency of our protocol, we summarize them as follows: in the experiments that did not consider client relocations, active notification prevailed over synchronization both in energy consumption and notification time and also scaled linearly with client populations. In the real scenarios though, active notification prevailed only in energy consumption but not in notification time. The subsequent replacement of PGD by an on-line algorithm made this difference more evident. So choosing between active notification and adaptive synchronization seemed to be a tradeoff between energy consumption and notification time. This tradeoff was avoided in the experiments that involved client relocations by extending adaptive synchronization so as to incorporate also the variance in energy consumption. This extension yielded in fact lower energy consumption and notification time. Thus we concluded that our protocol should always follow Rule 4 instead of Rule 3 and that the clients’ tuning should be done according to the extended synchronization process. As concerns multicasting, it should only start if synchronization has been established with each client. These conclusions have been incorporated in the protocol’s implementation that we describe below.

5 PROTOCOL IMPLEMENTATION
Mobile code requires platform independence as well as adaptation to available resources and environment restrictions. For these reasons, the implementation of our protocol was based on Java and computational reflection [23]. At the time of writing the bandwidth-allocation model was not implemented, as it involves low-level communication issues that were addressed separately.

In general, our implementation is stratified into three layers, i.e. communications middleware, groupware services, and user interface. A schematic view of this implementation is given in Figure 8:
As shown in this figure, the user interface consists of a number of awareness widgets, which realize the functionality entailed by Rules 4 and 5 and provide also interoperability among devices with diverse interface features. As concerns the other two layers, we examine them below in detail and justify the dependencies among various components which are shown in the figure.

5.1 Communications Middleware

The lowest layer of our implementation is the communications middleware, which provides to the layers above it event handling, connection establishment, and location management services. In general, middleware mediates interaction among applications and infrastructures (i.e. operating systems and networks) and enables also interoperability among heterogeneous components of distributed systems. By using thus middleware we aim to satisfy requirement [iv] of Section 2.4, which is not addressed directly by the protocol rules.

Because traditional middleware (e.g. CORBA, Jini, Microsoft’s .NET) cannot handle the dynamic aspects of mobile environments, we used reflective middleware [7]. Reflection denotes the ability of a system to monitor and change its execution [23]. As concerns the communications middleware of our implementation, it monitors and changes the protocol’s execution through three meta-objects, i.e. event, connection, and location. Instances of the first two of them exist in both the clients and the servers, while instances of the location meta-object exist only in the clients. Associated with each event meta-object is a thread that handles event and message queues, as well as a method that extends Java object serialization in order to handle buffered messages. Event meta-objects also trap alert signals and pass them to the groupware service layer in order to execute Rule 2. Moreover, they trigger acknowledgment events whenever the groupware service layer raises the End-of-Message flag.

The connection meta-object, in turn, abstracts the details of the WAP SI standard of wireless communications [38], providing thus to the upper layers a form of network transparency. For example, in order to bypass limitations of handheld devices regarding the message size, it splits large messages into smaller WAP SI’s. This meta-object provides also other services that cannot be analyzed in detail due to space limitations, but they are similar to the services that are usually provided by middleware to communication networks. The location meta-object, finally, raises an exception whenever the time gap between two successive End-of-Message flags exceeds a threshold value. In that case, the location meta-object provides a new method that tries to capture signals
within a different band of frequencies and informs accordingly the groupware service layer so as to execute Rule 8. The threshold value is smaller than the usual time gap between successive events, so the clients are kept continuously aware of their current location.

5.2 Groupware Services

The next layer in our implementation is another reflective module that implements groupware services, which include notification and synchronization, session management, and data sharing. Regarding the first, it has been implemented in terms of the reaction-diffusion metaphor.

For example, as suggested by MoMA, events can be represented by internal fields, while sensitivity functions (which determine what entities are sensitive to what fields and in what states) can be defined relative to the event types in which the clients have registered their interests. In this sense, the implementation of Rule 1 consisted of a definition of sensitivity functions for several entities, followed by a field-diffusion rule that determined where (i.e. to which entities) an event could be propagated, based on its type and the sensitivity function of each one of the corresponding entities. According to [32], diffusion implies “the existence of a space where the involved entities are situated and can move” (p.509). In mobile CSCW there are two such spaces, i.e. the physical space and the awareness space. So in our protocol we mapped the former to the latter by assuming that the server of a new cell represents just another entity in the awareness space. In other words, we considered diffusion as “the ability of clients to cross cell boundaries and become thus sensitive so to more fields (i.e. receive information from more than one server)”.

Rule 2 also denotes diffusion and can be represented by a trigger rule [32], according to which a server changes its state upon the receipt of an event. In our implementation, events are attached to shared objects and, whenever the latter change state they trigger the events, forcing thereby the communications middleware to react. In fact, depending on the value of the sensitivity function for each event, the server relays the event to a neighboring server or performs two actions in sequence, i.e. saves a copy of that event in its memory and then starts emitting alert signals (these two actions indicate that the server is also sensitive to that event). As concerns the diffusion itself, it is manifested by the emission of alert signals that represent fields originated at the server. The sensitivity of all clients to these fields is positive. Moreover, these fields are external because they are not part of the collaboration semantics but have been incorporated in our protocol in order to facilitate synchronization.

The synchronization imposed by Rule 3 denotes reaction, since more than one entity changes state. The interruption of signal emission, in turn, defines another trigger rule according to which the server changes its state upon a network disconnection (in terms of implementation, a typical case of computational reflection). As Rule 3 mandates, if this disconnection owes to a client’s relocation to another cell, the new server informs accordingly the previous one and the protocol’s execution jumps automatically to Rule 8. This denotes another reaction, since disconnection affects more than one entity. As concerns finally the hooking of an event to a previous one (which can be repeated several times), it was implemented as another trigger rule in which the interruption of emissions (state) and the arrival of a new event (field) resulted in another state (state2), which was denoted by the simultaneous propagation of multiple events (fieldk, fieldk-1, fieldk-2, etc).
The synchronization imposed by Rule 4, on the other hand, can be expressed in MoMA by a
diffusion rule since, upon receiving an external field (client’s signal), the server starts propagating
its own internal fields (namely awareness information). Regarding the submission of a query when
a client does not get anything within a period $St$, it was emulated in our implementation by
redefining first the sensitivity function of clients so as to make them insensitive to alert signals
when they operate in the sleep mode or when a network disconnection occurs. We see also that the
diffusion (propagation) of an external field affects the conditions of client reaction, something
that cannot be expressed in the spatial model. In MoMA however, it can be expressed by transport
rules that determine how a field can affect the position of an entity in the awareness space. As
concerns our protocol, an interference that causes disconnection for example forces a client to
change its position in the awareness space by becoming insensitive to alert signals. Using reflection,
we emulated such situations by allowing the changes that were made to the event meta-object to
incur other changes to sensitivity functions.

As concerns moreover the switching between the active and sleep mode, it has been realized
through the combination of two transport rules. Specifically, we regarded each entity as the
source of a field (i.e. a clock variable) to which this entity was sensitive; if the entity was not
receiving anything over an entire tuning period (namely whenever the clock variable was reaching
a threshold value [32, p.509]), then the entity was changing position in the awareness space by
becoming insensitive to alert signals. Similarly, if the entity had remained in the sleep mode for a
certain period (which was denoted by another threshold value), it was changing position again by
becoming sensitive to alert signals. We regarded clock variables as internal fields, since in
synchronous CSCW they are directly relevant to collaboration semantics.

Rule 5, in turn, was implemented as an ordinary trigger rule (except the user’s intervention),
while Rule 6 was implemented through a combination of reaction and field-diffusion rules. We
assumed moreover that the field-distribution function is assigned proper values in the handoff
area, since entities located in that area can receive information from two servers simultaneously,
which implies that they are sensitive to more than one field.

Rule 7 denotes another transport, since an entity that is sensitive to the End-of-Message flag
reacts by changing its position in the awareness space, i.e. it switches to the sleep mode after a
short period and is no longer sensitive to events. Rule 8, finally, can be enforced by another
trigger, according to which the current server becomes sensitive to the external field propagated
by the new server and reacts accordingly. This state change owes to the client’s relocation, which
represents a situation where the spatial model in [29] falls short (in fact, the focus of the current
server may have no intersection with the nimbus of the new one, since that nimbus is peripheral
to cooperative work). Because however we have not realized yet the bandwidth-allocation model
of Section 3.5.1, Rule 8 is not fully operational at the moment.

6 CONCLUSIONS AND FUTURE RESEARCH
Mobile CSCW breaks some assumptions that are implicit when collaborators work over fixed
networks, such as continuous power supply and network reliability. This observation calls for a mix
of mechanisms that can make CSCW effective in mobile settings. In this paper, we have tried to
address this requirement by a combination of groupware design principles and mobile computing mechanisms, which we have assessed with simulation experiments.

More specifically, we have developed a notification protocol and a bandwidth-allocation model which strike a balance between energy consumption and notification time. Our contribution is the first in the CSCW literature that accounts for these factors, and it aims to assist collaboration across large areas by preserving synchronous interaction as much as possible. A conclusion that we derived from the simulations, is that active notification is not always superior in terms of time efficiency, due to the likelihood of network congestion which is high in that case. To reduce this likelihood, our notification protocol utilizes a synchronization process that changes according to an on-line algorithm so as to incorporate the effects of client relocations and minimize the consumption of energy by them. The adaptive character of this process indicates a good fit of the reaction-diffusion metaphor for representing awareness in mobile settings, since the process can be directly implemented in terms of this metaphor. The bandwidth-allocation model, in turn, enables the delivery of awareness information to all clients simultaneously, by allocating the bandwidth to them according to the number of multicast messages and data requests.

We have incorporated the above protocol into a groupware prototype, whose implementation has prompted us to investigate further into a number of issues. As we mentioned in Section 4.2, for example, we must extend our simulations to check whether different synchronization algorithms yield lower energy consumption and notification time. Moreover, we must formalize our protocol and verify its correctness with respect to interface consistency, as the latter can be affected seriously by network disconnections and client movements. Finally, because awareness information is often provided simultaneously with data, we want to develop a data replication strategy for mobile environments that will preserve data consistency across networks and will also enable rapid responsiveness to user requests.

REFERENCES


Constantinos Papadopoulos (M’96) holds a PhD in Computer Science from Lancaster University and currently he works as a Lecturer in the School of Informatics of the University of Manchester.

He is an experienced teacher in Computer Science with over three thousand hours of lecturing at the American College of Greece (Deree) and the University of Manchester, while his professional experience includes twelve years in Control Data - Athens and the Greek Ministry of Finance (KEPYO), where he was developing software applications for transactional information systems. Moreover he has been the main architect of the TAXIS system, whereby the Greek Ministry of Finance has managed to confine the tax evasion in the country. As concerns his research interests, they include currently Mobile Computing, Computer Supported Cooperative Work, Information Security, and Formal Methods. Moreover, he has had considerable expertise in Discrete-Event Simulation, which he uses extensively in his research work.

Dr Papadopoulos is a Member of the IEEE and the ACM and he devotes his spare time to Byzantine art and philosophical reasoning.