Demonstrator III (Distributed Diary)

July 8, 1997

1 Introduction

The Diary application allows a set of users, each equipped with his/her own handheld, to arrange for meetings. This application is an example of *coordination* among a set of mobile users and it is structured to take into account aspects peculiar of mobile computing:

- Disconnection may be a primary mode of operation.
- Movements of disconnected users are not predictable.
- Disconnections may be unannounced.
- Users may remain disconnected for long times (compared to the time-outs employed at the transport level).
- The user must be in control of any decision regarding when to download data, when to participate in a computation and similar.
- Concurrent updates operated by distinct users may be inconsistent.
- The user must be in control of any decision about changes to the shared data, in particular, upon conflicting updates.

Users shall be able to coordinate their decisions even if they *never* communicate directly to each other. In contrast, most distributed algorithms for non-mobile computing that solve similar problems (i.e., consensus) assume, perhaps implicitly, that the participants may communicate directly with each other for a "sufficient" amount of time. Such an assumption is not realistic in the scenario considered in this demonstrator.

Since updates to the Diary involve scheduling of activities in the real world, we assume that the ultimate decision about every update must be always taken by the users; the application shall merely help users in taking such decision. In particular, in the presence of conflicting updates, the Diary first detects that a conflict occurred. Then, it may either declare that (some of) the conflicting updates are to be rejected, or it may attempt to "modify" these updates so that they do not conflict with each other anymore. In any case,
all involved users are notified about the conflict and their approval is necessary before making a "modified" update indeed permanent.

This strategy must not be seen as a limitation. It is in fact a worst-case scenario: all involved users must be contacted upon each update (whether conflicting or not); to some extent, they must be given the freedom of "negating" the approval to a given update (i.e., they might not be able to attend a meeting that they previously approved); they must be notified about every conflict possibly occurred. Automated resolution of conflicting updates depends essentially on the semantics of the shared data and the functionality that one wishes to embed in the application. As such, these rules are largely independent of the issues of major interest in this demonstrator.

The document is structured as follows. First we shall describe the overall structuring and functionality of the Diary (section 2). Then we shall describe the interface presented to users (section 3) and the environment in which the demonstration will take place (section 4). In section 5 we shall describe the salient aspects of the implementation and in section 6 we shall discuss the major design alternatives. Finally, we shall discuss the relation between the demonstrator and the consistency architecture in the Phase I final report. Unless stated otherwise, we shall use the term computer to mean "computer permanently connected to the stationary network", i.e. any machine in the infrastructure that provides support to mobile users equipped with hand-helds. The part of the demonstrator that runs on computers is denoted as stationary part, whereas the part that runs on hand-helds is denoted as mobile part.

2 Overall structuring

A meeting is essentially: (i) a set of participants; (ii) an indication of where the meeting will take place and of its topic; (iii) starting date, starting time and expected duration. The collection of all meetings may be thought of as an abstract object that we shall denote as the Diary. The Diary is shared among all users and is kept on the stationary network. In particular, in the permanent storage of a computer.

Each hand-held maintains an application-controlled (actually, user-controlled) cache of the Diary. Updates to a cache are reflected onto the Diary when the user is connected to the stationary network and decides to transfer the updates. Similarly, updates to the Diary are reflected to a cache when the user of the corresponding hand-held is connected and decides to. The user has thus substantial control over the use of the potentially costly wireless links.

The contents of the Diary must satisfy a number of constraints, that may involve the information in either the same meeting (i.e., the meeting shall not start in the past) or in different meetings (i.e., a given person cannot participate in two or more meetings at the same time). In the following, we shall refer only to constraints among different meetings, as the constraints relevant to a single meeting can be handled easily. The only kind of constraint that has been taken into account is the following: any two meetings with overlapping time frames must have disjoint sets of participants. Two or more meetings
that do not satisfy the defined constraints are said to arise a *conflict*. In this case, the Diary detects that a conflict has occurred, notifies all involved users about the conflict and leaves the meeting information unmodified.

Each hand-held may connect to and disconnect from the stationary network unpredictably. When an hand-held is connected, the user may *fetch* information from the Diary and/or *flush* information onto it. In particular, the user may fetch all updates to the Diary that "involve" the user himself and that occurred while the user was disconnected; moreover, the user may flush all the activity performed while disconnected onto the Diary. A user is "involved" in any meeting that requires his presence.

Updates to the Diary include *insertion* of a proposed meeting and *voting* for a proposed meeting. When a meeting has been just inserted in the Diary, it is marked as *Pending*. All participants in a *Pending* meeting are required to give a vote, either *Yes* or *No*. If all participants have voted *Yes*, then the meeting is marked as *Committed*, whereas a single *No* causes the meeting to be marked as *Aborted*. The various state transitions are summarized in Fig. 1.

A participant may vote *Yes* and then vote *No* on the same meeting. On the other hand, any *No* vote is an irrevocable decision. A participant that has voted *No* and then wishes to change his mind, shall propose another "slightly different" meeting. However, the two phases of voting *No* and proposing a new meeting are automated by the graphical interface.

A new meeting cannot be inserted in the Diary if it conflicts with other either *Pending* or *Committed* meetings. In case of conflict, the originator of the new meeting is informed that the meeting is *Conflicting*.

To summarize, the user may perform four major actions: (i) defining new meetings; (ii) voting for pending meetings; (iii) transfer data from the mobile part to the stationary part; (iv) transfer data from the stationary part to the mobile part. Actions (i) and (ii) correspond to updating the cache. Action (iii) propagates these updates to the Diary. Action (iv) reflects to the cache updates originated by other users.

## 3 Graphical interface

The user performs the above operations through a graphical interface that is enclosed in the mobile part. The main window of this interface is shown in Figure 2. The major components of this window are:

- A calendar (left side of the central part). The month displayed ("current month") can be changed by clicking on the arrows ("<<" and ">>"). The days in the current month that contain meetings are displayed in a different color. The "current day" is selected by clicking on a day in the current month.

---

1. This is not really a "state" because the meeting is not inserted in the Diary, thus it is not shown in figure 1.
A participant changed his mind

![Diagram of state transitions of a meeting in the Diary]

**Figure 1**: State transitions of a meeting in the Diary.

- The list of meetings in the "current day" (right side of the central part). This list contains a description of each meeting and of the set of votes that have been given so far. The interface displays with different colors meetings that are PENDING, COMMITTED, CONFLICTING or ABORTED. It also adequately emphasizes those meetings whose updates have not been reflected yet onto the stationary part, and are thus still unknown to the other users.

- The Control menu allows shutting down the interface.

- The All Meetings menu contains three items. One of these items creates a dedicated window listing all COMMITTED meeting. The two other items create a similar window for the meetings that are PENDING and those that are CONFLICTING.

- The New Meeting button allows defining a new meeting and is discussed in more detail below. The Flush flushes on the stationary part all updates to the hand-held that have not been propagated yet. The Get button fetches from the stationary part all updates to the Diary that have not been reflected to the hand-held yet. Double-clicking on a day is equivalent to the two actions: (i) defining that day as the current day; and (ii) clicking on the New Meeting button.

- The bitmap to the right of the Get button emphasizes the presence in the hand-held of meetings for which the user has not voted yet: The semaphore next to the bitmap is red when there is at least one such meeting and gets green when there is none.
By clicking on the semaphore, a window containing the list of the corresponding meetings pops up.

- The four bitmaps below the buttons provide synthetic indications about the differences between the information in the mobile part and in the stationary part. Each bitmap is associated with a semaphore and is dedicated to a different kind of information, as described below. Intuitively, a red semaphore indicates that a **Flush** operation is necessary (left-most bitmap) or that the last **Get** operation fetched new updates from the stationary part (remaining bitmaps). We describe each bitmap from left to right.

  - The left-most bitmap is devoted to updates done into the mobile part and not reflected yet into the stationary part. The semaphore is red when there is at least one such update and gets green when these updates have been transferred into the stationary part.

  - The second bitmap is devoted to updates just fetched from the stationary part. These updates may be: (i) new meetings; (ii) **PENDING** meetings whose set of votes has changed; (iii) **PENDING** meetings that were already known but whose place and/or timeframe has been changed. The semaphore is red when there is at least one such update and it may change color only upon completing a **Get** operation.

  - The third and fourth bitmaps are also devoted to updates just fetched from the stationary part: meetings that have become **COMMITTED** and that have become **ABORTED**, respectively. In other words, these two bitmaps are associated with a subset of the updates (ii) of the second bitmap. The color of the semaphore is governed by the same rule as the second bitmap.

In each case, by clicking on the semaphore, a window containing the list of the corresponding updates pops up.

- The **Status** indication tells the user whether the mobile part is communicating with the stationary part or it is idle\(^2\). It also tells the user whether connectivity was lost during an interaction with the stationary part.

- The **Coverage area** indication notifies the user about the identity of the nearest computer, if any. This indication is meant to simulate similar information that is likely to be provided in practice, by means of a proper beaconing protocol, for instance.

By clicking on the **New Meeting** button, a dedicated window pops up that allows defining a new meeting in the current day. An example is given in Figure 3. The title bar shows that the window is taken from the mobile part of user "Giacomo". The participants in the meeting (or **guests**) are indicated in the right part and are selected from the list on the left

---

\(^2\) With the terminology of sections 5 and 6.1, it tells whether a session is in progress.
Demonstrator III: Distributed Diary