Geo-registers : an abstraction for spatial-based distributed computing

Matthieu Roy, François Bonnet, Leonardo Querzoni, Silvia Bonomi, Marc-Olivier Killijian, David Powell

Geo-registers : an abstraction for spatial-based distributed computing

Matthieu Roy[1], François Bonnet[2], Leonardo Querzoni[3], Silvia Bonomi[3], Marc-Olivier Killijian[1], and David Powell[1]

[1] LAAS-CNRS; Université de Toulouse; F-31077 France
roy@laas.fr | mkilliji@laas.fr | dpowell@laas.fr
[2] IRISA; Université Européenne de Bretagne; Rennes, F-35042
fbonnet@irisa.fr
[3] Sapienza Università di Roma; Roma, I-00185
querzoni@dis.uniroma1.it | silvia.bonomi@dis.uniroma1.it

Abstract. In this work we present a novel abstraction that allows a set of distributed processes, aware of their respective positions in space, to collectively maintain information associated to an area in the physical world. This abstraction is a logical object shared between participating processes and provides two operations, namely read and write, as for traditional registers.
This paper provides a specification and implementation of such a shared object for non concurrent writes and multiple readers, then describes a possible extension to concurrent writes.
We also provide insights on how this kind of objects could be useful to program location-aware applications.

1 Introduction

1.1 Motivation

The advent of massively distributed systems in which every user carries an entity with full computing power, including communication and positioning capabilities, allows us to envision many new applications and services, truly decentralised by nature and tightly coupled to the position of entities. Nevertheless, the deployment of such systems imposes the de?nition of new formalisms that capture the new features of these systems and allow algorithms to be developed and reasoned about.

Indeed, for the first time ever, there exists a strong coupling between the graph of possible interactions between entities and the geographical distribution of nodes. The strength of the classical Internet model was to abstract all communication links into a complete graph, where any two nodes willing to cooperate could open a connection. This clique-based system modelling can no longer be applied to distributed dynamic systems where entities can communicate using short-range wireless technologies.

Two main issues arise when trying to deal with such large-scale and location-aware systems:

- the evolvable nature of such systems imposes any mechanism built for them to be resilient to mobility- and failure-induced changes to the composition and/or topology of the system,
- new features of such systems are not represented in traditional distributed models, namely their dynamicity and locality, and more specifically the geographical distribution of entities.

Our research efforts are focussed on providing suitable abstractions to reason on mobile, large scale and geographically aware systems. More specifically, in this paper we introduce a software abstraction, namely *geo-registers*, that can be used to associate some values to a geographic location. This abstract object is shared among all the processes that move inside a specified area in the surroundings of the object position and can be accessed, as a traditional register, through read and write operations. Since our primary motivation is to tackle the geographical aspects of such systems, we consider systems composed of correct entities that move in a geographical space. Unlike traditional failure mode assumptions, such as crash of processes or byzantine behaviors, we solely consider movements as the only source of uncertainty in the system. The paper provides specifications for both a *serial writes and concurrent reads* geo-register and a *concurrent reads and writes* geo register. A distributed algorithm implementing the former specification is also provided.

Outline of the paper The paper is made of five parts. This first section introduces the problem we solve and presents related works. In the next section, we define the formal model of the system, and identify the basic abstractions our work is based on. Section 3 presents the main contribution of our work. We define the geo-register abstraction in the case of non-concurrent write operations, provide an implementation in the case of one-hop ad-hoc networks, and prove that the implementation satisfies its specification. Section 4 proposes an extension to the concurrent writers model, and the last Section concludes the paper.

1.2 Related works

Shared storage objects are very attractive abstractions that can be used for indirect process communication and that simplify the development of applications. In mobile environments some solutions have been proposed to implement such shared objects.

In [4], an atomic memory implementation for mobile ah hoc networks is presented. This implementation is a quorum based one and implements an atomic register in a region of the space where nodes can move. The authors abstract the mobility of nodes by introducing a static concept: the *focal point*. A focal point is a region where nodes can move and can have a state, depending on the number of nodes inside the region. The authors point out that, in order to implement this shared object, some additional constraints must be put on these focal points, e.g. only a fraction of focal points are allowed to fail. As a result, this model introduces some limitations on node motions and density over time.

In [12] these constraints are reduced and only a small set of mobility constraints are considered to implement the atomic register. In particular, the main novelty with respect to Dolev's work is the possibility of tolerating unlimited number of focal point failures during the entire system lifetime. In fact, Tulone's mobility constraints only require a minimum coverage of the mobile nodes without additional constraint on movements.

Both approaches mentioned above differ from the one presented in this paper because their aim is to build a register maintained by a set of geographic regions (the focal points) while our aim is to build a register in a given geographic region. While previous works focus on using geographic dispersion of nodes to tolerate failures, we are interested in the orthogonal problem of defining a shared storage in an area, in isolation with the remainder of the system.

As a result, our work considers a different semantics for the geo-localized register; we remove any constraint on nodes movements and allow the register to loose its state if no node is in the area covered by the register.

In order to build such shared objects, nodes need to communicate. In a mobile environment, algorithms have been introduced [6, 7, 9, 2] to provide a geocast with probabilistic guarantees. In Dolev's and Tulone's approachs, a deterministic geocast is required in order to implement such atomic memories. Deterministic guarantees are given in [11, 5, 3] for multicast and in [10] for broadcast in MANET. In [1] are shown some impossibility results defining under which assumptions it is possible to implement a geocast. In our work, we concentrate our attention on a single and limited area and, since our model is similar to the one considered in [10], we can assume to have a reliable communication primitive to implement our geo-localized register.

1.3 Application example motivation

Current personal navigation systems based on the use of GPS calculate a route that optimizes journey duration only based on static data, like the maximum allowed speed for each road segment. In the best case the navigation system can take into account historical data to match your journey with trip times experienced in the past by other users on the same journey (e.g. TomTom IQ Routes technology). Traffic alert services are sometimes integrated in these devices, but they are provided through a centralized infrastructure.

A decentralized and geographically aware storage service could be used to store dynamic information on traffic that can be used by the navigation system to plan journeys taking into account current traffic conditions on public roads.

The shared geo-localized register proposed in this paper can be used as a basic block to build a completely distributed application for traffic jam reduction. In particular, users can access the geo-located register to

gather information on the traffic state in a certain zone when approaching it and can report their experience with the traffic, i.e. the detection of a traffic congestion state, when leaving the area covered by the register. Using the information contained in the register, the navigation system of the cars can adjust the planned route in order to avoid congested roads. The novelty of this approach is in the complete distribution of the traffic detection state that can be achieved only with the resources provided by the users.

2 Architecture and model

2.1 Formal system model

Entities The system is composed of entities $(p_i)_{i=1,2,...}$ of an infinite set Π that evolve in a 2-dimensional space, or geographic space. The entities are *correct*, i.e. execute correctly the algorithms and do not crash, and *anonymous*, i.e. execute the same algorithm and do not own a unique identifier.

Communication and positioning All entities are equipped with a positioning device and wireless network capabilities. The entities are aware of their position at every time with infinite precision. They can move in the space continuously with a bounded maximal speed V_{max}.

Area of interest An area A is a geographic surface, i.e. a continuous subset of the space.

At every instant t, let $\texttt{active}_A(t)$ be the set of processes in A. Since processes are correct and move continuously, $\texttt{active}_A(.)$ evolves by additions or removals of entities :

$$\forall t, \exists \epsilon : \begin{cases} \texttt{active}_A(t+\epsilon) & \subset \texttt{active}_A(t) \quad \text{leave operation} \\ \vee\ \texttt{active}_A(t) & \subset \texttt{active}_A(t+\epsilon) \quad \text{join operation} \end{cases}$$

Definition 1. *The area A is valid if $\forall t, \texttt{active}_A(t)$ is a clique w.r.t. communication capabilities, i.e. any two processes in the set can communicate.*

Notice that this definition is very general, and can represent either an area that is a disc of radius equal to half the communication radius, or a multi-hop network equipped with a routing algorithm that guarantees a complete communication graph.

2.2 Execution model

In this paper we are only interested in operations that take place in a particular area, say A. As a result, an execution for an area A corresponds to a set of operations performed in this area. Since all entities are supposed to be correct, we also assume that every operation takes a finite amount of time.

To simplify reasoning, in the following we will refer to the starting and the ending of a given operation Op using two operators, $\mathsf{Begin}(Op)$ and $\mathsf{End}(Op)$. By definition, $\mathsf{Begin}(Op)$ corresponds to the (external observer) time of invocation by the caller p_i of the operation Op, and $\mathsf{Begin}(Op)$ is defined by the end of the operation Op from the system's point of view.

Definition 2 (Execution). *Let A be an area, and $\mathcal{O}$ be the set of possible operations available in A. An execution in A is a set of operations $\mathbb{E}$ such that $\forall Op \in \mathbb{E}, \mathsf{Begin}(Op) < \mathsf{End}(Op)$*

Definition 3 (Concurrency). *Two operations Op_1 and Op_2 in an execution $\mathbb{E}$ are* non-concurrent *if*

$$\big((\mathsf{End}(Op_1) < \mathsf{Begin}(Op_2)) \vee (\mathsf{End}(Op_2) < \mathsf{Begin}(Op_1))\big)$$

If the above statement does not hold, Op_1 and Op_2 are concurrent.

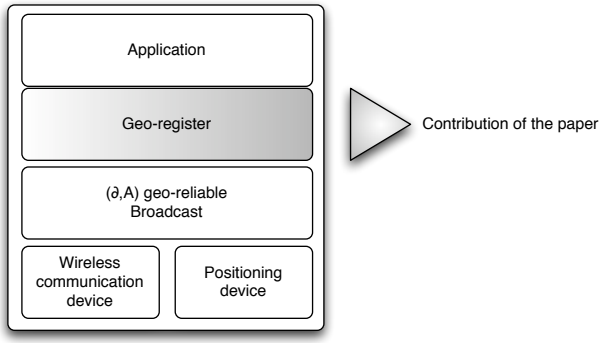

Fig. 1. Architecture for a given area A

2.3 Architecture of the System

In this paper, we are interested in providing an abstraction, or building block, that can be directly used by an application. The shared geo-localized storage service, or *geo-register*, will be built on more simple building blocks.

Even though the systems we consider are large-scale, we focus on an area of interest A for which some basic abstractions are available, as shown in Figure 1.

Recall that we suppose that a perfect positioning service is available, and that wireless communication is possible within the whole area A. On top of these two basic, hardware-oriented service, we suppose that a *geo-reliable broadcast* service is implemented. The goal of this service, presented in the next subsection, is to merge geographical and communication-based information to produce a reliable message passing service in the area A with known maximum transfer time.

Our contribution lies in the definition, implementation and proof of the geo-register service, that is presented in the next section, and could be used directly by an application to handle storage in an area A.

2.4 Geo-Reliable Broadcast

A geo-reliable broadcast is a communication primitive that guarantees that all processes[4] located in an area A receive the broacasted message. This primitive is built on top of wireless communication described in the previous section.

More formally the geo-reliable broadcast is defined as follows:

Definition 4 ($(\delta, A)-$geo-reliable broadcast). *Let δ be a positive number and A be an area. A $(\delta, A)-$geo-reliable broadcast enjoys the following properties:*

- *every process $p \in A$ can issue a* `broadcast`(m)
- *if m is a message broadcasted at time t by a correct process p that is in the area A from time t to time $t + \delta$, then all correct processes remaining in A between t and $t + \delta$ deliver m by time $t + \delta$.*

This definition is relatively weak, since it does not take into account the processes that may enter or leave the area during the broadcast, and only focuses on entities that stay in the area for the whole duration of a broadcast. The δ period of time can be either fixed and known in advance or not. In this paper we consider that δ is fixed for a given A, and known by processes[5].

Such a primitive is implementable when the area A remains valid (the communication graph is a complete graph) for the whole period of time δ.

[4] Recall that, in our model, all processes are correct

[5] Notice that, in practice, δ is related to the diameter of A. A larger area needs a larger δ to ensure a reliable dissemination

Definition 5 (Core region).
Let A be a valid area equipped with a $(\delta, A)-$geo-reliable broadcast.
A core region *A' associated to A is a subset of A such that every message m sent at time t by any process p in A' using $(\delta, A)-$geo-reliable broadcast will be delivered (by time $t+\delta$) by every process q that was in A' at time t.*

Notice that this definition abstracts away some physical parameters of the system. In particular, the definition implies that a process that is in A' at time t will be in A at time $t+\delta$.

3 Geo-Registers

A *geo-register* is the abstraction of a storage mechanism attached to a particular area, that can be used to store and retrieve collectively pieces of information.

Since no assumption is made neither on the number of entities nor on their density, the area of interest may sometimes be populated, and sometimes be empty. As a consequence, data persistence cannot be guaranteed for the whole execution.

Intuitively, a geo-register implements a temporally ordered sequence of (traditional) registers. Every element of the sequence corresponds to a temporal interval where entities populate the node. As soon as the area becomes empty, the state of the storage is lost, and when entities reenter the area, a new instance has to be created.

3.1 Non Concurrent Writers Geo-Register

Intuitively, a geo-register is an abstraction of shared storage attached to a surface in space which provides a `write` operation and a `read` operation. Inspired by the seminal paper of Lamport [8], we provide a specification of a *non-concurrent safe* register, then extend it to the case of concurrent writers in the next Section.

`write` and `read` operations are not instantaneous, and require a finite amount of time. The register is said to be *non concurrent writers* if it does not support concurrent writes. Notice that it does not require that a single process is allowed to perform a write operation. It only states that, whenever multiple processes call write operations, no two write operations can occur concurrently.

The semantics of the non concurrent writers geo-register is defined with respect to 1) the most recently completed write operation and 2) the write operations possibly concurrent with a given read operation, that can be defined as follows:

Definition 6 (most recently completed write operation). *Let Op be an operation performed on the register. The most recently completed write operation before Op is by definition W_y such that*

$$\mathsf{End}(W_y) = \max\{\mathsf{End}(W_z) : \mathsf{End}(W_z) < \mathsf{Begin}(Op)\}$$

Definition 7 (possible outcomes of a read operation on a non concurrent writers register). *Let R be a read operation performed on the register, and W_y the most recently completed write operation before R. Let $\mathcal{CW}$ be the set of write operations that are concurrent with R, and V the set of values written by operations in $\mathcal{CW} \cup \{W_y\}$. V is the set of possible outcomes of the read operation R.*

Definition 8 (non concurrent writers geo-register). *Let A be a valid area, and A' its associated core region. A* safe geo-register *for (A, A') provides a* read *and a* write *operations such that*

- *a* read *operation can be issued at time t by processes in* $\mathtt{active}_A(t)$
- *a* write *operation can be issued at time t by processes in* $\mathtt{active}_{A'}(t)$[6]

[6] an operation Op called by p_i could also be said to be *valid* if p_i remains in A from $\mathsf{Begin}(Op)$ to $\mathsf{End}(Op)$, but this alternative model would require further assumptions such as movements control

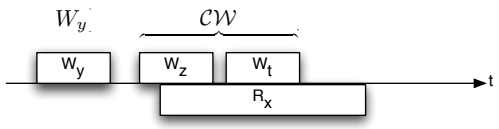

Fig. 2. Example of possible outcomes for a read operation. Here, $V = \{y, z, t\}$

– *Let a* read *operation R_x be issued by a process in* $\texttt{active}_A(\mathsf{Begin}(R_x))$. *Let W_y be the most recently completed write operation before R_x and V be the set of possible outcomes of R_x. The semantics of the read operation R_x is as follows:*

(Partial Amnesia): *if, since* $\mathsf{End}(W_y)$, *there exists an instant t_1 s.t.* $\texttt{active}_{A'}(t_1) = \emptyset$, *it returns either a value in V or $\perp$.*

$$\mathsf{End}(W_y) \xrightarrow[\exists t \in [\mathsf{End}(W_y), \mathsf{Begin}(R_x)] : (active_{A'}(t) = \emptyset)]{} \triangleright\ \mathsf{Begin}(R_x) : x \in V \cup \{\perp\}$$

(Safety) *if* $\texttt{active}_{A'}$ *has never been empty from* $\mathsf{End}(W_y)$ *to* $\mathsf{Begin}(R_x)$, *it returns a value in V.*

$$\mathsf{End}(W_y) \xrightarrow[\forall t \in [\mathsf{End}(W_y), \mathsf{Begin}(R_x)] : (active_{A'}(t) \neq \emptyset)]{} \triangleright\ \mathsf{Begin}(R_x) : x \in V$$

3.2 Implementation for 1-hop communication

In this section we propose an implementation of a geo-register that complies with the specification defined above.

Target system Our solution implements the geo-reliable broadcast by a simple wireless broadcast, that is a 1-hop communication broadcast. This primitive implements a (δ, A)-geo-reliable-broadcast only if any pair of processes in A are in wireless range of communication, i.e. the diameter of A must be smaller than R, the (common) wireless range of communication[7]. The parameter δ is a known period of time fixed by the geo-reliable-broadcast primitive from lower parameters; it abstracts the implementation details of the primitive that may include more than one broadcast due to message collisions.

We define A', the core region of A, as the maximal region (included in A) such that any entity in A' cannot leave A in less than 4δ time. Since entities have a bounded maximal speed V_{max}, the area A' can also be defined geographically as the set of points in A that are at least at distance $4\delta * V_{max}$ from A's boundaries.

Algorithm With these definitions of A and A', we propose an implementation of a geo-register in Figure 4. In our implementation, each entity in A maintains a local register R_p of the geo-register.

The main part of the code (first control thread) consists in the initialization of this local register when an entity enters in A, and freeing the local register when the entity leaves A.

Explainations of the first wait for 2δ time The first wait for 2δ is needed to ensure that the initialization of the local register R_p is correctly made. Otherwise it may happen that a process p initializes its local register to an old value that has been replaced (by a more recent write). In the following we give an example of such a bad initialization:

[7] From this observation it appears that the maximal available zone A for our geo-register implementation corresponds to a disk of radius $\frac{R}{2}$, which corresponds to the constraints of our simplistic geo-reliable broadcast implementation

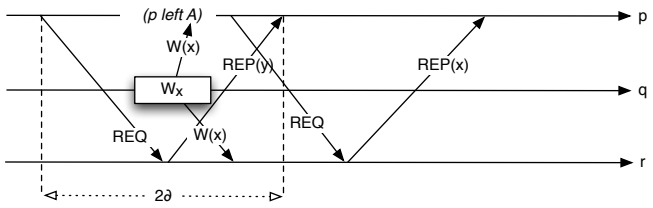

Fig. 3. Example of race condition for $REP(y)$ messages

Let us consider a process p that arrives in A just after a `write` operation W_x has been initialized by some process q. Without waiting 2δ, p is not guaranteed to receive the value x, as shown in Figure 3. If p does not wait for 2δ, it may receive (before 2δ) a reply $REP(y)$ corresponding to a REQ (made by him or another process) with a value $y \neq x$; indeed this reply may be sent by a process r just before the end of q's `write` operation (so just less than δ time after the entrance of p in A). Since both the REQ and the reply may respectively take up to δ time to arrive p has to wait for 2δ.

Note that this delay of 2δ can be reduced to δ if we suppose that requests include identity of the process that makes the request and a sequence number of requests made by this process. The id allows to differentiate requests from different processes, whereas the sequence number is needed to differentiate two requests from the same process.

Geographically controlled thread:

when p enters A:
  $R_p \leftarrow void$;
  wait for
    □ ($W(x)$ is received) : $R_p \leftarrow x$; exit;
    □ (2δ time delay elapsed)
  RB_send(REQ)
  wait for
    □ ($REP(v)$ is received) : $R_p \leftarrow v$;
    □ ($W(x)$ is received) : $R_p \leftarrow x$;
    □ (2δ time delay elapsed) : $R_p \leftarrow \perp$;
when p leaves A:
  free(R_p);

Communication controlled thread:

upon reception of (REQ) : **if** ($R_p \neq void$) **then RB_send**($REP(R_p)$)
upon reception of ($W(x)$) : $R_p \leftarrow x$

Read and Write operations:

When p is in A:
`read`() : **wait until** ($R_p \neq void$) **then return**(R_p);

When p is in A':
`write`(x) : **RB_send**($W(x)$);

Fig. 4. Implementation

3.3 Proof of Correctness

Duration of read/write operations The `write` operation consists only in one geo-reliable broadcast; then any `write` operation will terminate in at most δ time. The `read` operation is most of the time instantaneous, since it consists mainly in a local memory access; however if a process p initiates a `read` soon after entering A, the operation may be delayed upto at most 4δ (maximum time needed to initialize the local register R_p). Both operations take then always a finite amount of time to complete.

Semantics of the `read` *operation* With respect to the specification, it is obvious from the implementation that the `read` operation always returns either a written value, or $\perp$. Then we prove that (1) in case of non concurrent `write` operation, a `read` operation returns either the value corresponding to the most recently completed `write` or $\perp$, (2) in case of non concurrent `write` operation and if A' remains non-empty from the end of the most recently completed `write` to the beginning of a `read` operation, then this `read` can not return $\perp$ (and then returns the value of the most recently completed `write` operation), (3) in case of a concurrent `write` operation then in both previous cases, the value being currently written can also be returned by the `read` operation.

(1) Suppose there is no concurrent `write` operation during a given `read` operation R performed by a process p. Let W_x be the most recently completed `write` operation before R as defined in the previous section. Two cases have to be considered:

- p was in A at $\mathsf{Begin}(W_x)$ and remains in A until $\mathsf{Begin}(R)$. p has then received the value x, has updated its local register R_p to x, and thus returns x for the operation R.
- p has entered A (for the last time) at time $t_p > \mathsf{Begin}(W_x)$. p may have received the written value x, that is then similar to the previous case, or it may have missed the written value x. In the latter case, before p can terminate its `read` operation R, it has to update its local register R_p. This update can only be performed by the first control thread (since there is no concurrent `write` operation), that implies the update can occur only 2δ time after t_p. Any broadcast received by p after $t_p + 2\delta$ can not have been initialized before $t_p + \delta > \mathsf{Begin}(W_x) + \delta > \mathsf{End}(W_x)$, i.e. such a broadcast has been initialized after $\mathsf{End}(W_x)$; thus any process (if any) that may reply to p's REQ has received $W(x)$. Consequently p receives a reply $REP(x)$ and initializes its local register either to x or to $\perp$ (if no process was able to transmit the current value of the register).

(2) Let us suppose that A' remains non-empty since the most recently completed `write` operation W_x, we want to prove that the value $\perp$ cannot be returned by the `read` operation R performed by p. Let us prove first, by induction, the following property $\mathcal{P}$: any process that spends some time in A' between $\mathsf{End}(W_x)$ and $\mathsf{Begin}(R)$ has its local register set to x during (at least) all the time spent in A'. Secondly we will deduce that p has then necessarily updated its own local register to x before its `read` operation R.

- Any process that is in A' at time $t = \mathsf{End}(W_x)$ has received $W(x)$ and then has updated its local register. Indeed if a process is in A' at t, it was already in A at time $t - \delta$ and remained in A from $t - \delta$ to t, which implies from the definition of the geo-reliable-broadcast that it has received the `write`.
  Suppose the property $\mathcal{P}$ is true until some time t_q with $\mathsf{End}(W_x) < t_q < \mathsf{Begin}(R)$ when a process q is entering A'. At that time q has been in A for at least 4δ (due to the definition of A'), thus its local register has then been initialized ($R_q(t_q) \neq void$). The last update has been done (a) either by receiving a `write` operation, (b) or by receiving (or not) a REP in the first control thread. The case (a) implies that q has received $W(x)$ since this is the most recent one. The case (b) implies that q enters in A after $\mathsf{Begin}(W_x)$ (since it missed $W(x)$). q made a request 2δ period of time after entering A; the request is then made at time $t_{req} > \mathsf{Begin}(W_x) + 2\delta > \mathsf{End}(W_x) + \delta$. If q receives a reply, it cannot have been initialized before $t_{req} - \delta > \mathsf{End}(W_x)$. Then any reply received by p contains necessarily the value x. It remains to prove that p receives indeed a reply before entering in A'. This is due to the fact that $t_{req} < t_q - 2\delta$[8] and by

[8] This inequality is obtained by the fact that A' has radius 4δ smaller than A

hypothesis A' remains non-empty from $\mathsf{End}(W_x)$ to t_q; it implies that at least one entity has received p's request (between t_{req} and $t_{req} + \delta$), then this entity sends a reply with value x that is received by p at most at time $t_{req} + 2\delta < t_q$. Thus q updates its local register to x before entering in A'. By induction $\mathcal{P}$ remains true until $\mathsf{Begin}(R)$.
- p's `read` operation R waits for p to update its local register; the reasoning done for q in the previous point can be applied to p provided p remains in A, from which we can conclude that p updates its local register to x, the most recently written value and then returns x.

(3) Suppose now there is one concurrent `write` operation $W(y)$ during a given `read` operation R performed by a process p. In that case, the previous reasonning remains correct except that now p can also update its local register with the communication controlled thread if $W(y)$ is received by p before the end of R.

4 Concurrent Writers Version

In the previous sections we specified the properties of a geo-register that support multiple readers and multiple writers with the restriction that write operations are not concurrent.

Now we extend that specification to take into account multiple concurrent writes that act on the same geo-register. Exactly like in the previous case, we assume each write operation W_x to be characterized by $\mathsf{Begin}(W_x)$, the instant in time when the write operation begin, and $\mathsf{End}(W_x)$, the instant in time when the write operation ends. The actual write of the value x in the register can generically happen at any instant of time $t \in [\mathsf{Begin}(W_x), \mathsf{End}(W_x)]$

Without loss of generality, as done by Lamport in [8], we assume that there is an initial operation $W_\perp$ that is not concurrent with any other operation.

If we consider an execution on a geo-register, we can separate it in subset of operations where no concurrent writes occur. Within these subsets we can identify sequences of operations that can be totally ordered.

Definition 9 (write sequence). *Let $\mathbb{W}(t)$ be a set of write operations executed on a geo-register. We define a write sequence w as an ordered set $[W^0 \rightarrow \ldots \rightarrow W^j]$ of non concurrent write operations:*

- $\forall W^i \in w : W^i \in \mathbb{W}(t)$,
- $W^0 = W_\perp$,
- $\forall i \in [0, j-1] : \mathsf{End}(W^i) < \mathsf{Begin}(W^{i+1})$.

These sequences of writes are intersting only when they are maximal, in the sense that no write operation can be added to sequence without adding concurrency:

Definition 10 (maximal write sequence). *Let $w = [W^0 \rightarrow \ldots \rightarrow W^j]$ be a write sequence on $\mathbb{W}(t)$. w is maximal w.r.t. $\mathbb{W}(t)$ if any write operation of $\mathbb{W}(t) \setminus w$ is concurrent with w:*

- $\nexists \overline{W} \in \mathbb{W}(t) \setminus w,\ \exists i \in [0, j-1] : End(W^i) < Begin(\overline{W}) \wedge End(\overline{W}) < Begin(W^{i+1})$.
- $\nexists \overline{W} \in \mathbb{W}(t) \setminus w,\ End(W^j) < Begin(\overline{W})$

Consider the example depicted in Figure 5 where distinct entities execute concurrent writes on a same geo-register. In this scenario the set of maximal write sequences in $\mathbb{W}(\mathsf{Begin}(R_x))$ includes the following entries:

$$W_\perp \rightarrow W_1 \rightarrow W_3$$
$$W_\perp \rightarrow W_2$$

Definition 11 (most recent write in sequence). *Given a write sequence w we define the most recent write operation in the sequence as the last operation in the ordered set.*

Note that, given the set $\mathcal{W}(t)$ of all the maximal write sequences in $\mathbb{W}(t)$, the set constituted by the most recently completed write operations from all the sequences contains write operations that are all concurrent.

With the help of maximal write sequences we can now define the new possible outcomes of a read operation. It follows directly the definition of concurrent writers geo-register that is the same as the non-concurrent one, except on the set of possible outcomes values of a read operation.

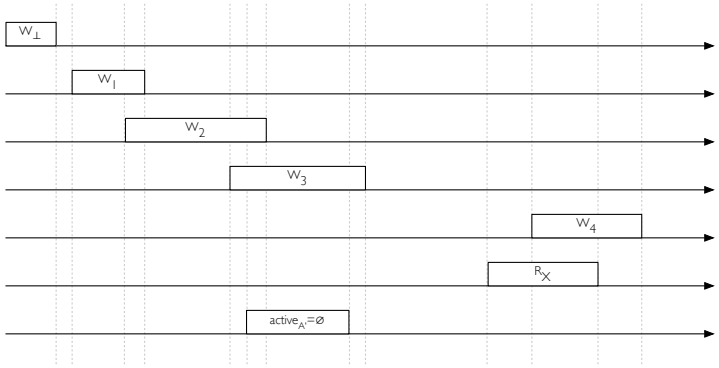

Fig. 5. An execution scenario with concurrent writes from distinct processes.

Definition 12 (possible outcomes of a read operation on a multi writer register). *Let R be a read operation performed on the register. Let $\mathcal{RW}$ be the set of most recent write operations from the maximal sequences in $\mathbb{W}(\mathsf{Begin}(R_x))$, and $\mathcal{CW}$ be the set of write operations that are concurrent with R. Let V be the set of values written by operations in $\mathcal{CW} \cup \mathcal{RW}$. V is the set of possible outcomes of the read operation R.*

Definition 13 (concurrent write/read geo-register). *Let A be a valid area, and A' its associated core region. A* safe geo-register *for (A, A') provides a* `read` *and a* `write` *operations such that*

- *a* `read` *operation can be issued at time t by processes in* $\texttt{active}_A(t)$,
- *a* `write` *operation can be issued at time t by processes in* $\texttt{active}_{A'}(t)$,
- *Let R_x be a* `read` *operation. Let W_y be the most recently completed write operation before R_x and V be the set of possible outcomes of R_x. The semantics of the read operation R_x is as follows:*

(Partial Amnesia): *if, since W_y, there exists an instant t_1 s.t.* $\texttt{active}_{A'}(t_1) = \emptyset$, *it returns either a value in V or $\perp$.*
(Safety) *if* $\texttt{active}_{A'}$ *has never been empty from* $\mathsf{End}(W_y)$ *to* $\mathsf{Begin}(R_x)$, *it returns a value in V.*

Applying this definition to the example depicted in figure 5 the read operation R_x should return one of the values written by the write operations W_2 and W_3 (because these operations are in $\mathcal{RW}$) or the value written by the write operation W_4 (because this operation is in $\mathcal{CW}$). The value $\perp$ cannot be returned because the area A' is always non-empty between the end of the operation W_3 and the beginning of the read R_x.

5 Conclusion

In this paper we provided the specification, implementation and proof of a geo-localized storage service for mobile systems. Differently from other research works on similar systems, we are interested in providing a local-only abstraction that can be used by applications that require to store information only when entities populate the area where data are to be stored.

As an example, we briefly explained how this kind of register could be used to improve personal navigation systems, but we believe this abstraction can be useful to any application that enjoys the following property: *"When no user is in the area A then the state of the storage service can be lost"*. This kind of behavior can be found in many geo-localized, collaborative applications such as augmented reality games or localized service offering (e.g. car seat sharing offering).

To provide an implementation and a proof of the register, we based our work on a reliable, timed broadcast. Although this assumption seems strong, it can be implemented in a mobile system provided the area of interest considered is relatively small.

We also provided insights on the behavior of our implementation in the case of concurrent writes, and showed that it enjoys reasonably strong semantics, similar to the *safe* registers of Lamport[8].

Future research directions we focus on include the introduction of process failures, and the possibility of providing stronger semantics.

References

1. R. Baldoni, K. Ioannidou, A. Milani *Mobility versus the Cost of Geocasting in Mobile Ad-Hoc Networks*, International Symposium on Distributed Computing, DISC07, LNCS, 48–62, 2007, 2007
2. J. Boleng, T. Camp, and V. Tolety. *Mesh-based geocast routing protocols in an ad-hoc network.* In Proceedings of the 15th International Parallel and Distributed Processing Symposium (IPDPS), pages 184?193, April 2001.
3. R. Chandra, V. Ramasubramanian, and K. P. Birman. *Anonymous gossip: Improving multicast reliability in mobile ad-hoc networks.* In Proceedings of the 21st International Conference on Distributed Computing Systems (ICDCS), pages 275?283. IEEE Computer Society, 2001.
4. S. Dolev, S. Gilbert, N. Lynch, A. Shvartsman, J. Welch, *GeoQuorums: Implementing Atomic Memory in Mobile Ad Hoc Networks*, in: Proceedings of the 17th International Conference on Distributed Computing, October 2003, pp. 306–320
5. S. K. S. Gupta and P. K. Srimani. *An adaptive protocol for reliable multicast in mobile multi-hop radio networks.* In Proceedings of the 2nd Workshop on Mobile Computing Systems and Applications (WMCSA), page 111. IEEE Computer Society, 1999.
6. Y. Ko and N. H. Vaidya. *Geotora: a protocol for geocasting in mobile ad hoc networks.* In Proceedings of the 8th International Conference on Network Protocols (ICNP), page 240. IEEE Computer Society, 2000.
7. Y. Ko and N. H. Vaidya. *Flooding-based geocasting protocols for mobile ad hoc networks.* Mobile Network and Application, 7(6):471?480, 2002.
8. L. Lamport. *On Interprocess Communication.* Distributed Computing (1985)
9. W. Liao, Y. Tseng, K. Lo, and J. Sheu. *Geogrid: A geocasting protocol for mobile ad hoc networks based on grid.* Journal of Internet Technology, 1(2):23?32, 2001.
10. M. Mohsin, D. Cavin, Y. Sasson, R. Prakash, and A. Schiper. *Reliable broadcast in wireless mobile ad hoc networks.* In Proceedings of the 39th Hawaii International Conference on System Sciences (HICSS), page 233.1. IEEE Computer Society, 2006.
11. E. Pagani and G. P. Rossi. *Reliable broadcast in mobile multihop packet networks.* In Proceedings of the 3rd Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom), pages 34?42. ACM Press, 1997.
12. D. Tulone *Ensuring strong data guarantees in highly mobile ad hoc networks via quorum systems* in Ad Hoc Networks Volume 5 , Issue 8 (November 2007) Pages 1251–1271