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
[2] IRISA; Université Européenne de Bretagne; Rennes, F-35042
[3] Sapienza Università di Roma; Roma, I-00185

Abstract. In this work we present an abstraction that allows a set of distributed processes, aware of their respective positions in space, to collectively maintain information associated with an area in the physical world. This abstraction is a logical object shared between participating processes that provides two operations, namely read and write.

1 Introduction

Motivation The advent of massively distributed pervasive systems in which every user carries a powerful computing device with communication and positioning capabilities, allows us to envision many new intrinsically decentralized applications and services that are tightly coupled to the position of entities. Two main issues arise when trying to deal with such dynamic and location-aware systems:

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

Our research efforts are focussed on providing suitable abstractions to reason about mobile, large-scale and geographically-aware systems. More specifically, in this paper, we introduce a software abstraction, called a *geo-register*, that can be used to associate some values to a geographic location. Unlike traditional failure mode assumptions, such as process crashes or byzantine behaviors, we solely consider movement as the only source of uncertainty in the system. The paper provides the specification for a *serial writes and concurrent reads* geo-register. A distributed algorithm implementing this specification is provided.

Related work 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, a few solutions have been proposed

* The full version of this work is available as a research report [4]. This work is partially funded by FP6 NoE ReSIST, contract nr. 026764.

to implement such shared objects. In [1, 3], atomic memory implementations for mobile ad hoc networks are presented. Both approaches differ from the one presented in this paper because their aim is to build a register maintained by a set of geographic regions 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 of the remainder of the system.

2 Architecture and model

Formal system model 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 and do not crash, and *anonymous*, i.e., execute the same algorithm and do not own a unique identifier.

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

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 only by additions or removals of entities. 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.

Execution model 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 time, as perceived by an external observer, at which the caller p_i invokes Op, and $\mathsf{End}(Op)$ is defined by the end of the operation Op from the system's point of view, i.e., the time at which the last action of the Op invocation protocol terminates. Two operations Op_1 and Op_2 in an execution are *non-concurrent* if $\big((\mathsf{End}(Op_1) < \mathsf{Begin}(Op_2)) \vee (\mathsf{End}(Op_2) < \mathsf{Begin}(Op_1)\big)$, else Op_1 and Op_2 are *concurrent*.

Geo-Reliable Broadcast To abstract away physical parameters of the system, we suppose that the system is equipped with a *geo-reliable broadcast*. A geo-reliable broadcast is a communication primitive that guarantees that all processes located in an area A receive messages broadcasted to that area. From an implementation point of view, this primitive is built on top of wireless communication and positioning capabilities.

Definition 1 ($(\delta, A)-$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* $\texttt{broadcast}(m)$
- *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 processes that stay in the area for the whole duration of a broadcast.

Definition 2 (Core region). *Let A be a valid area equipped with a $(\delta, A)-$geo-reliable broadcast. A* core region *A' associated with 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 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 is guaranteed to be in A at time $t + \delta$.

3 Non Concurrent Write Geo-Registers

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

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 area. 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.

Inspired by the seminal paper of Lamport [2], we provide a specification of a *non-concurrent regular* register. More complex semantics, like multi-writer ones, are explored in [4].

The semantics of a non concurrent write geo-register[4] is defined with respect to 1) the most recently completed write operation and 2) the write operations possibly concurrent with a read operation, that can be defined as follows:

Definition 3. *Let Op be an operation performed on the register. The most recently completed write operation before Op is by definition W_{Op} such that*
$\mathsf{End}(W_{Op}) = \max\{\mathsf{End}(W_x) : \mathsf{End}(W_x) < \mathsf{Begin}(Op)\}$

Definition 4. *Let R be a read operation, and W_R 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 $\mathcal{CW} \cup \{W_R\}$. V is the set of possible outcomes of the read operation R.*

Definition 5 (geo-register). *Let A be an area, and A' an associated core region. A* geo-register *for (A, A') provides* read *and* 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 $\mathtt{active}_{A'}(t)$*
- *Let a* read *operation R be issued by a process in $\mathtt{active}_A(\mathsf{Begin}(R))$. Let W_R be the most recently completed write operation before R and V be the set of possible outcomes of R. The value returned by R satisfies:*

[4] Notice that non concurrent write implies that, whenever multiple processes call write operations, no two write operations occur concurrently.

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

Implementation for 1*-hop communication* We provide an implementation for the simple one-hop broadcast-based system. In this solution, 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. The proof is omitted and can be found in [4].

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** ($R_p \neq void$)
  return(R_p)

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

Fig. 1. Implementation

4 Conclusion

In this paper we provided the specification and a simple implementation of a geo-localized storage service for mobile systems. Unlike other similar work, we are interested in providing a local-only abstraction that can be used by applications that require information to be stored only when entities populate the area. Future research directions include the introduction of process failures, and the possibility of providing stronger semantics such as write concurrency.

References

1. S. Dolev, S. Gilbert, N. Lynch, A. Shvartsman, J. Welch, *GeoQuorums: Implementing Atomic Memory in Mobile Ad Hoc Networks*, PODC 03, pp. 306–320
2. L. Lamport. *On Interprocess Communication.* Distributed Computing (1985)
3. D. Tulone *Ensuring strong data guarantees in highly mobile ad hoc networks via quorum systems* in Ad Hoc Networks Volume 5 , Issue 8 (Nov. 2007).
4. M. Roy , F. Bonnet , L. Querzoni , S. Bonomi , M-O. Killijian , D. Powell. *Geo-registers: an abstraction for spatial-based distributed computing.* LAAS report Nr. 08487. Oct 2008.