Distributed Objects with Sense of Direction

G. v. BOCHMANN
University of Ottawa

P. FLOCCHINI
Université de Montréal

D. RAMAZANI
Université de Montréal

1 Introduction

An object system consists of a collection of objects and their relations; each object has a state (e.g., local variables) and a behavior (set of actions it may execute) and the global behavior of a system is described in terms of interactions between its objects.

When dealing with object systems, the manipulation of names is crucial to such en extent that an objects has been defined as "any entity in a system which deserves a name ..." [7]. Furthermore, some authors have stated that the discipline of object-oriented programming is all about manipulating object names (e.g., [8]).

In object systems, some of the main objectives of a naming scheme are: to *designate* an object, to distinguish between objects *independently of their state*; to *communicate* to other objects the knowledge about some objects; to *ease the design of algorithms* and the description of recursive data structures. Furthermore, a naming scheme should be *stable* and *reliable*, that is, names must remain valid, and keep denoting the same objects when the system changes its state.

In order to achieve these goals, the existing naming schemes either use an approach based on *global* names (e.g., the name of an object is an intrinsic property which distinguishes it from the other objects) or they use *hierarchical* names based on the location of the objects (e.g., email addresses).

On the other hand, locality of names and independence on the location are two very desirable characteristics of a naming scheme, and to date no proposal is available with such properties. Moreover, the existing naming schemes are not compatible; this fact gives rise to major problems when dealing with open systems in which the different components use different naming schemes. This is due to the lack of a unifying, formal specification of namings.

In this paper we propose a general approach to naming based on the notion of sense of direction [2], which can be used as a framework where to study and compare new naming schemes. Classical approaches like global naming can be modeled by this framework; our approach, however, is suitable for the creation of more flexible local naming sheme. More precisely, naming schemes based on sense of direction represent the first proposal where: objects use *local names* to refer to other objects, there are mechanisms for the non-ambiguous *communication* of names (i.e., an object receiving a name from another object understands who that object is referring to), and names are *location-independent*.

The paper is organized as follows: in Section 2 we introduce the ploblem of object naming, we discuss some of the approaches commonly used, and we identify the properties that a good naming scheme should have; in Section3 we introduce the notion of sense of direction; in Section 4 we propose a new general approach for creating naming schemes based on sense of direction; finally, in Section 5 we show how to model the classical global naming with sense of direction, and we propose a new good naming scheme.

2 Objects Naming

Object Systems. An object system consists of objects and interrelations between them; its behavior is described in terms of interactions between its constituant objects. Each object has a state denoted by the value of its attributes (local variables) and a behavior which denotes the set of actions it may execute. In addition, an object may have attributes which refer to other objects. Different models of object interactions exist. Two well-known models are:
1) asynchronous message passing, i.e. interaction by sending messages. In this model, an object "knows" the other object and it uses a referent to that object for sending a message to it.
2) synchronous communication by rendezvous: In this case there is no initiator for the communication. The objects involved in the communication agree on certain conditions, and they know each another when the communication occurs. This latter model is more abstract than the previous one.

In the following we shall consider the asynchronous message passing model, however our results holds also for the synchronous one.

From Global to Local Naming. As mentioned in the introduction, an object naming scheme should associate names to object independently of their state, it should allow objects to exchange knowledge about other objects, it should be stable and reliable.

An approach that has been followed to achieve these goals is to name objects based on their *identity*, where the identity of an object is an intrinsic property of that object that uniquely distinguishes it from the others. Such an approach has however a major drawbacks: it implies the need of maintaining global names. In distributed object systems, the management of global names is very expensive and often impossible when the system is large and dynamic. Furthermore, distributed object systems are often open systems thus it must be easy to connect them without changing the names of the objects; such a task cannot be achieved efficiently if the object naming is global. Thus the need of *local names*.

In order to have a local naming scheme, different alternatives have been proposed. As a first step towards finding adequate solutions to this problem, Wieringa and de Jonge [15] have formalized naming in object systems. However, their formalization is geared towards a single, specific property of naming which is the ability to count objects based on their names and ...

———————————————————————

??? COMMENT: Here I don't understand very well. Are they proposing local names or they still use a global approach ? In case it is global, in what sense it is a first step toward locality ?

———————————————————————-

In Open Distributed Computing, and CORBA systems, the need of local naming is crucial. In both systems the approaches to this problem are characterized by the idea of grouping object names based on certain criteria giving rise to hierarchical naming schemes.

In particular, in Open Distributed Computing [9], it has been introduced the notion of naming context. A naming context is a relation between objects and names; when an object is introduced into a naming context, a name is bound to it in such a way that objects in the same naming context have all distinct names. Thus, global naming is required only within a naming context, while objects in different naming contexts are allowed to have the same name. A naming context is an object in itself and, as such, it can be referenced (i.e., it has a name) and it can belong to one or more naming context provided that different naming context belonging to the same naming context are distinguishable. When an object needs to reference an object in the same context it just uses its name; otherwise the reference to the object must contain its name and the list of context which have to be "traversed" in order to get to the object.

——————————————————-

???COMMENT: I know, this sentence is not clear, I was trying to explain in other words Dunia's sentence about the federation (Objects from a naming context must can reference objects in other naming contexts; in order to do that the two naming contexts must be "federated" in a third naming context in which they are denoted by distinct names.) which was not too clear either ....

——————————————————

A similar approach is taken in CORBA systems [14] and DCE implementations [1].

————————————

???COMMENT: What is DCE ? before using the abbreviation we should say what it stays for (same for ODP ...etc.)

————————————

In CORBA, we have the same concept of naming context and names are structured. They consist of a handle and a remainder. The handle usually represents a naming context while the remainder represents the path from the handle to the object. In DCE, the concept of naming context is replaced by that of a cell. Cells can be combined like federating naming contexts. In addition, in DCE, there is a unique universal object identifier (UUID) associated to each component of DCE.

The results of all these approaches is a hierarchical scheme where an object must use a “path” of naming contexts (or ...) to denote another object. Even if local names are allowed within a certain area, locality is only apparent and the resulting scheme is still global. Moreover, names are highly sensitive to location.

Anothe major problem related to these approaches is the following: consider a distributed system consisting of an ODP component, a CORBA component, and a DCE component. Although the naming of these components could be compatible, we still have different interpretations of naming in these components.

??? COMMENT: I don’t understant what you mean. From what is written it looks to me that the approaches are exactly the same, in one case they speak about naming context, in another of cell .... but it’s not clear why the schemes cannot be combined in the same system. Dunia, could you maybe add a sentence to better explain that ?

This is caused by the lack of a formal specification of these namings. In order to devise an adequate solution for the above problem, we need a framework in which to answer to deep semantic questions concerning naming, and to propose solutions based on semantic coherence and theoretical feasibility. For instance, if the retained solution has to be integrated in a language such as C++ or Java, it has to be formalized. Finding such a framework is not an easy task. It requires examining the impact of object naming on the definition of objects and how objects interact. Ultimately, it affects how we use the object paradigm. However, such a framework would be necessary for demonstrating the adequacy and consistency of a naming mechanism.

Good Naming Schemes. From the study of the existing techniques for object naming it clearly emerges the need of a general approach. Furthermore, a good naming scheme should have the following properties:

1. *Locality*.
Objects use local names to refer to other objects. To avoid local ambiguity, each object uses different names for referencing different objects.

2. *Mechanisms of name communication.*
The scheme provides mechanisms for the non-ambiguous communication of names; i.e., an object receiving a name, understands with no ambiguity what is the object denoted by that name.

3. *Location Independent*.
The naming scheme does not change when objects are physically moved.

3 Sense of Direction

In this section we introduce the notion of sense of direction.

Sense of direction is a well known property of labeled graphs that has been extensively studied in the context of distributed computing (e.g., [2, 6, 5, 10, 4, 11, 12, 13, 16]) and that can be used to provide good object naming schemes.

Let $G = (V, E)$ be a graph where nodes correspond to objects and edges correspond to reference links between objects. Let $E(x)$ denote the set of edges incident to node x.

Every node associates a label to each incident edge; let $\lambda_x(x, z)$ denote the label associated by x to the edge $(x, z) \in E(x)$; such a label denotes the name that x uses when referring to y. We denote with λ the set of all λ_x. The system can thus be described by the pair (G, λ). In the following we will assume that each node can distinguish among its incident edges; i.e., $\forall x \in V$, $\forall e_1, e_2 \in E(x)$, $\lambda_x(e_1) = \lambda_x(e_2)$ iff $e_1 = e_2$.

A *path* in G is a sequence of edges in which the endpoint of one edge is the starting point of the next edge. Let $P[x]$ denote the set of all paths with x as a starting point, and let $P[x, y]$ denote the set of paths starting from node x and ending in node y. Let Λ be the extension of the labeling function λ from edges to paths.

We now introduce the notion of coding and decoding function.

A *coding function* is a function that maps a sequence of labels corresponding to a path in G into a value, in such a way that two sequences corresponding to different paths starting from the same node are mapped in the same value iff the ending point of the paths are the same.

Definition 1 Coding Function
A coding function f *of a graph* (G, λ) *is a function such that:* $\forall x, y, z \in V$*,* $\forall \pi_1 \in P[x, y]$, $\pi_2 \in P[x, z]$ $f(\Lambda_x(\pi_1)) = f(\Lambda_x(\pi_2))$ iff $y = z$

For example, consider a 2-dimentional mesh where the edge labels are from the set $\{north, south, east, west\}$ and are assigned in the natural globally consistent way. Intuitively, in such a system the coding function is the one that allows us to understand that the sequences $(north, south, north, east)$ and $(east, north)$ lead to the same destination when starting from the same node.

A *decoding function* h for f is a function that, given a label and the coding of a string (a sequence of labels), returns the coding of the concatenation of the label and the string. More precisely,

Definition 2 Decoding Function
Given a coding function f*, a decoding function* h *for* f *is such that* $\forall x, y, z \in V$*,*

such that $(x, y) \in E(x)$ *and* $\pi \in P[y, z]$, $h(\lambda_x(x, y), f(\Lambda_y(\pi)) = f(\lambda_x(x, y) \circ \Lambda_y(\pi))$, *where* $\circ$ *is the concatenation operator.*

We can now define sense of direction:

Definition 3 *[3]*
A system (G, λ), *has a* Sense of Direction *($\mathcal{SD}$) iff the following conditions hold:*
1) there exists a coding function f,
2) there exists a decoding function h *for* f.

We shall also say that (f, h) is a sense of direction in (G, λ). Several examples of sense of directions (e.g., chartographic, chordal, neighbouring) have been described in [3].

4 Naming Schemes based on $\mathcal{SD}$

In this Section we show that sense of direction provides a general approach to the construction of naming schemes.

Let (G, λ) be an object system with sense of direction (f, h). Each node associates a local name to the other nodes in the systems; let $\beta_x(y)$ denote the name that node x locally associates to y and let us denote with β the set of all β_x.

By definition of labeling λ, we have that $\lambda_x(x, y) = \beta_x(y)$; so, names that nodes gives to their neighbours are defined by the labeling λ.

Object naming schemes can be constructed based on the existence of the coding function f; more precisely, the name that an object gives to another object is either the label of the link between them (if there is such a direct link), or the coding of a sequence of labels leading to it.

Object naming scheme β.

Given a coding function f, the local object naming scheme β_x for x
is constructed as follows:

$$\forall x, y, \; \beta_x(y) = \begin{cases} \lambda_x(x, y) & \text{if } y \in E(x) \\ f(\alpha) & \text{otherwise} \end{cases}$$

where α is the sequence of labels corresponding to an arbitrary path between x and y

The collection of all local object naming schemes $\beta = \{\beta_x : x \in V\}$
constitutes the (global) object naming scheme.

Given a graph representing an object system, many different labelings could be constructed to have different senses of direction (e.g., see Figure 1), each one providing a different naming scheme.

All the naming schemes based on sense of direction are clearly location-independent since the name an object uses for another object just depends on the labels of a path between them and not on their location. In the following we will show that all naming schemes based on sense of direction are not ambiguous (i.e., an object uses different names for different objects), and that they all have mechanisms for the communication of names (i.e., an object receiving a name understands who ...).

Theorem 1 *Let (G, λ) be a system of distributed objects with a sense of direction (f, h). The object naming scheme β is locally non-ambiguous, and it provides mechanisms for the communication of names.*

Proof.
To prove that there is no *local ambiguity*, we have to show that: $\forall x, y, z, \beta_x(y) = \beta_x(z)$ iff $y = z$. This property follows from the definition of coding function. In fact, let $x, y, z \in V$, $y \neq z$, $\pi_1 \in P[x, y]$ and $\pi_2 \in P[x, z]$. By contradiction let $\beta_x(y) = \beta_x(z)$; this is impossible because, by definition of f $f(\Lambda_x(\pi_1)) \neq f(\Lambda_x(\pi_2))$.
We will show that the decoding function h provides the mechanisms for the non-ambiguous *name communication*. By definition of h we have that: $\forall \langle x, y\rangle \in E(x),\ \forall \pi \in P[y, z]\ h(\lambda_x(x, y), f(\Lambda_y(\pi))) = f(\lambda_x(x, y) \circ \Lambda_y(\pi))$.
By definition of β we have that $\lambda_x(x, y) = \beta_x(y)$, $f(\Lambda_y(\pi)) = \beta_y(z)$ and $f(\lambda_x(x, y) \circ \Lambda_y(\pi)) = \beta_x(z)$, thus it follows that $h(\beta_x(y), \beta_y(z)) = \beta_x(y)$. This means that a node x, receiving an information about a remote node z from a neighbour $\beta_x(y)$i, can "translate" that information to understant how this object is called in its own local view. □

Object oriented applications can thus be viewed as evolving labeled graphs (nodes corresponding to objects and edges to references between objects). At each point in time, an object x "knows" the names of a subset of the other objects; the objects that are directly linked to it in the corresponding graph. During the application, x aquires knowledge about other objects by receiving information about them from objects already known. Every time x receives an information about a remote object r which was not known, it can "translate" this information into its local name $\beta_x(r)$ and possibly reference it thus creating a new link (x, r) labeled with $\beta_x(r)$.

Sense of Direction is thus a general framework where to study and compare naming schemes, in fact different naming schemes can be obtained depending on the choice of the labeling and of the coding function. In particular, some classical naming scheme used in the literature (for example, the global one), can be now described in terms of sense of direction; moreover, new, more appealing ones can be defined.

In the following Section we will show two particular instances of sense of direction. The first corresponds to the classical approach of global names, the second provides a new, truly local naming scheme.

5 Examples

5.1 Global Names: Neighbouring Naming

The classical solution that uses global names for the objects in the system can be modeled in our framework as follows.

Given a graph (G, λ), λ is a *Neighboring labeling* iff: $\forall \langle x, y\rangle \in E[x]$, $\langle z, w\rangle \in E[z]$,

$$\lambda_x(\langle x, y\rangle) = \lambda_z(\langle z, w\rangle) \text{ iff } y = w$$

That is, in a neighboring labeling, all the links ending in the same node x are labeled with the same label (see Figure 1 b)). It is easy to see that this labeling is a sense of direction [2]. The coding function is the following: $\forall \pi \in P[x_0],\ \pi = (\langle x_0, x_1\rangle, \dots, \langle x_{m-1}, x_m\rangle)$

$$f(\Lambda_x(\pi)) = \lambda_{x_{m-1}}(\langle x_{m-1}, x_m\rangle)$$

and the decoding function:
$\forall \langle x_0, y_0\rangle \in E(x_0),\ \forall \pi \in P[y_0],\ \pi = (\langle y_0, y_1\rangle, \dots, \langle y_{m-1}, y_m\rangle)$

$$h(\lambda_{x_0}(\langle x_0, y_0\rangle), f(\Lambda_{y_0}(\pi))) = f(\Lambda_{y_0}(\pi))$$

We can then construct our naming scheme:

$$\beta_x(y) = f(l_1, l_2 \dots, l_n) = l_n$$

This scheme is only apparently local, on the contrary it corresponds exactly to the situation where objects have all distinct names; in fact it is easy to see that $\beta_x(z) = \beta_y(z)$, $\forall x, y, z$.

5.2 A Good Naming Scheme: Chordal Naming

A *chordal* labeling of a graph $G = (V, E)$, with $|V| = n$, is defined by fixing a cyclic ordering of the nodes and labeling each incident link (x, y) of x by the distance (modulo n) in the above cycle (see Figures 1 a)).

Let λ be a chordal labeling. The system (G, λ) has a sense of direction [3]. Infact, the coding function f is a function that maps the sequence of labels (integer numbers) into their sum modulo n: $\forall x, y$, for any path $\pi = ((x_0, x_1), \dots, (x_{m-1}, x_m)) \in P[x_0]$

$$f(\Lambda_{x_0}(\pi)) = \sum_{i=0}^{m-1} \lambda_{x_i}(x_i, x_{i+1}) \bmod n$$

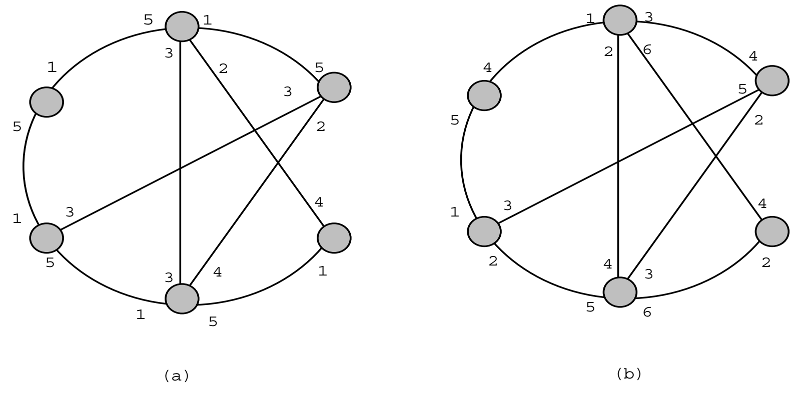

Figure 1: Graph with (a) Chordal and with (b) Neighboring sense of direction

It is easy to see that $f(\Lambda_{x_0}(\pi))$ is the distance, in the fixed cyclic ordering, between the starting node x_0 and the ending nodes x_m of the path. Clearly, given a different path $\pi' \in P[x_0, x_m]$, we would have that $f(\Lambda_{x_0}(\pi')) = f(\Lambda_{x_0}(\pi))$, thus, f is a coding function.
In other words, a sequence of integers corresponding to a path from x to y is coded into the distance between x and y in the cyclic ordering. Different sequences corresponding to different paths staring from and ending in the same nodes will be thus mapped to the same value.
The corresponding decoding function is the following: $\forall \langle x_0, y_0\rangle \in E(x_0),\ \forall \pi \in P[y_0]$

$$h(\lambda_{x_0}(x_0, y_0), f(\Lambda_{y_0}(\pi))) = \lambda_{x_0}(x_0, y_0) + f(\Lambda_{y_0}(\pi))$$

$$= \lambda_{x_0}(x_0, y_0) + \sum_{i=1}^{m-1} \lambda_{y_i}(y_i, y_{i+1}) \bmod n = \lambda_x(x, y) + \sum_{i=1}^{m-1} \lambda_{y_i}(y_i, y_{i+1}) \bmod n$$

As opposed to the example of the previous Section, in this case we have a truly local scheme. Consider, for example, node x in Figure **??** ..... it is references as ??? by etc .... exemple of communication of names: ...... etc ....

6 Conclusions

....

References

[1] Guide to writing DCE applications, digital equipment corporation.

[2] FLOCCHINI, P., AND MANS, B. Optimal election in labeled hypercubes. *Journal of Parallel and Distributed Computing 33*, 1 (1996), 76–83.

[3] FLOCCHINI, P., MANS, B., AND SANTORO, N. Sense of direction: definition, properties and classes. *Networks*. To appear. Preliminary version in *Proc. of 1st Colloquium on Structural Information and Communication Complexity*, 9-34,1994.

[4] FLOCCHINI, P., MANS, B., AND SANTORO, N. On the impact of sense of direction on message complexity. *Information Processing Letters 63*, 1 (1997), 23–31.

[5] FLOCCHINI, P., RONCATO, A., AND SANTORO, N. Complete symmetries and minimal sense of direction in labeled graphs. *Discrete Applied Mathematics*. to appear. Preliminary version in *Proc. of 27th SE Conference on Combinatorics, Graph Theory and Computing*; Congressus Numerantium 121, 3-18, 1996.

[6] FLOCCHINI, P., AND SANTORO, N. Topological constraints for sense of direction. *International Journal on Foundations of Computer Science*. To appear. Preliminary version in *Proc. of 2nd Colloquium on Structural Information and Communication Complexity*, 27-38, 1995.

[7] GOSCINSKI, A. *Distributed Operating Systems: The Logical Design*. Addison-Wesley, 1991.

[8] HOGG, J. Islands:aliasing protection in object-oriented languages. In *Proc. of OOPSLA* (1991), pp. 271–285.

[9] ISO/IEC JTC1. Information technology - open distributed processing - naming framework, ISO/IEC DIS147771.

[10] KALAMBOUKIS, T., AND MANTZARIS, S. Towards optimal distributed election on chordal rings. *Information Processing Letters 38* (1991), 265–270.

[11] LOUI, M., MATSUSHITA, T., AND WEST, D. Election in complete networks with a sense of direction. *Information Processing Letters 22* (1986), 185–187. see also Information Processing Letters, vol.28, p.327, 1988.

[12] MANS, B., AND SANTORO, N. On the impact of sense of direction in arbitrary networks. In *Proc. of 14th International Conference on Distributed Computing Systems* (Poznan, 1994), pp. 258–265.

[13] MASUZAWA, T., NISHIKAWA, N., HAGIHARA, K., AND TOKURA, N. Optimal fault-tolerant distributed algorithms for election in complete networks with a global sense of direction. In *Proc. of 3rd International Workshop*

on Distributed Algorithms (1989), Lecture Notes in Computer Science 392, Springer-Verlag, pp. 171–182.

[14] OMG. Naming service specification clause 3, corba services, OMG.

[15] R. WIERINGA, W. D. J. Object identifiers, keys, and surrogates - objects identifiers revisited. *Theory and Practice of Object Systems*. To appear.

[16] SINGH, G. Efficient leader election using sense of direction. *Distributed Computing 10* (1997), 159–165.