Architecture of a Web Operating System Based on Multiagent Systems

José Aguilar, Niriaska Perozo, Edgar Ferrer, and Juan Vizcarrondo

CEMISID, Dpto. de Computación, Facultad de Ingeniería, Universidad de los Andes
Av. Tulio Febres, Mérida, 5010, Venezuela
aguilar@ing.ula.ve

Abstract. The amount of systems, services and applications developed for the Web has grown considerably. The support of the existing operating systems to them, is not the awaited one. Like a solution to this necessity, we propose a model of operating system denominated SOW. It supports and handles a set of services in a heterogenous and dynamic environment like Internet. The SOW is conformed by four subsystems (resource, repositories, web object and communities manager subsystems), where each one carries out a series of coordinated functions, that allow an efficient use of the resources on Internet. In this work we present the design of the SOW based on Agents.

1 Introduction

Given the wide variety of unimaginable services everyday in the web, it's difficult to design an operating system to support each of those services individually. A new domain to solve these needs are the Web Operating Systems (WOSs), which have like main objective provide a platform that allows users to benefit from the computational potential offered in the web, through resource sharing and solving problems of heterogeneity and dynamic adaptability present in it. Thus, in order to reach the best performance in a dynamic environment of distributed resources such as the Internet, the WOS should be configurable and able to adapt to changes related to the availability of software and hardware resources. Taking into account that, the WOS model presented in this paper suggests a series of aspects in order to provide services that can adapt to the web's special features. Hence, our WOS (called SOW) offers a set of services to enable the utilization of the available resources through the Internet.

There are different proposals for management and integration of the computational resources available in a global computer system. Perhaps the most general project is the *WOS* [2] since it allows for management and integration of resources addressing the issue of heterogeneity and volatility in the Web. This project, as well as our proposal, is based on the idea of using versions as a solution to these problems. Other efforts to exploit the resources distributed in global computing include *Jini, Netsolve*, *Globe, Legion*, and *WebOS* [1, 3, 4, 5]. In contrast with our proposal, most of these systems require entry privileges (*login*) in participating devices and software. Particularly, our model differs from all the mentioned projects, in the sense that no centralized global resource catalog is needed. Our SOW model supports and manages a series of services in a heterogeneous, dynamic and adaptive context, under the application of a reconfiguration approach.

R. Khosla et al. (Eds.): KES 2005, LNAI 3681, pp. 700–706, 2005.

2 General Considerations of the Proposed Architecture

2.1 SOW Features

The main features of our SOW are: **i) Distributed and with different versions:** Different versions of services offer by the SOW are running simultaneously along the network; **ii) Dynamic:** The web is a constantly evolving entity, therefore, the SOW should adapt to unforeseen changes, through the dynamic configuration of its services; the dynamic modification of the information on the resources available in each node; the dynamic grouping of existing nodes according to certain features, and the migration and replication of web objects, **iii) Open:** Our SOW is an open system from two points of view: it accepts different technologies through the network (heterogeneity), and it also allows every node in the Internet to be incorporated into the system; **iv) Intelligent:** Each one of the SOW subsystems will have a certain level of intelligence for the development of some of its functions.

2.2 Integration of the SOW in a Distributed Environment

Our SOW should be at the top of a standard distributed platform (Middleware) in order to use the services provided by this platform. In this way, it coexists with traditional distributed applications, and can be used by a wide community. Particularly, among the distributed services that require our SOW are: a *Name Service* (white pages) for identification and location of resources in the distributed environment by designation; a *Directory Service*; (yellow pages) for identification and location of resources in the distributed environment by attribute; a *Security Service* that provides confidentiality, integrity, and availability; *Synchronization and Coordination Services*; a *Process Management Service* which will be in charge of the processors allocation, internal and global processes scheduling, and processes migration; a *Distributed File System* that will allow the management of different file systems in different nodes and the sharing of information; and a *Communication Services* that allow interaction between existing models (client/server, intermediaries (Proxy, Caches, among others), group communication (multicast), etc.). The interactions can be carried out by messages passing and shared memory.

3 Description of the SOW

The SOW is composed by the resource manager subsystem, the repositories manager subsystem, the web object manager subsystem and the community manager subsystem.

3.1 Resource Manager Subsystem (SMR)

In our proposal, the SMR is made up of mechanisms that allow managing and integrating computational resources present in the Web. The resources may have a physical representation, such as a file or a printer, or abstract representations, such as the CPU time. Due to the dynamic nature of the Web, it's impossible to develop a catalog

with every available resource and service. Therefore, instead of having an operating system that provides a set of fixed techniques of process scheduling or caches management, we have an operating system that satisfy the requirement with the particular scheduling technique needed in a given time. In such cases, the concept of versions will be used, and in this way, an action will have the possibility of being coordinated by combining the appropriate versions (which could be physically distributed along the network) in order to achieve an efficient execution of the required services. The SOW has a service configuration mechanism to adequate the utilization of these versions and to have the catalogue of the versions of the different resources, services or strategies. An inference engine handles every requirement to the system. This works as a demand managing reactive system through which heterogeneous computational environment resources provided by the Web are managed. It will particularly carry out the complete search process, determine the service to be provided and assign resources. The SOW will have various inference engines distributed along the network. Whenever a user tries to access a web resource, the SOW gets the request and this request is handled by the local inference engine, which will consult with its repository to determine whether it's possible to satisfy the request locally. Otherwise the request is sent to another inference engine and the previous process is repeated until the request is finally fulfilled. In general, each inference engine has access to its local and remote resource repositories, which store information about the resource versions. The local resource repository is always consulted; the remote resource repository is consulted only if the resource request can't be satisfied locally.

3.2 Local Repositories Manager Subsystem (SMRL)

One of the great challenges of our SOW is implementing an efficient algorithm for the search and discovery of services or resources available in the network. The repositories associated with the SOW nodes provide the necessary information in order to satisfy the service requests. Thus, every node uses its repositories in order to store and continuously update the information about the node, the services and the resources available in it and in its near nodes (a near node is defined by the community manager system). This allows the interaction with different repositories, each one offering different versions of the available services, resource, techniques, applications, etc. The local repositories will have the ability to store local information. When that information is perhaps replicated in other sites, data coherence mechanisms must be at hand in order to keep the information consistent. The SMRL will establish policies to make efficient use of the available capacity in the devices and will use efficient local search procedures. For example, some of the techniques are for the storage optimization, disk access scheduling, among others. On the other hand, since most Internet applications require real time interactions and it's necessary to share available information, the SMRL might establish efficient transaction mechanisms such as the use of parallel I/O scheduling, etc.

3.3 Remote Repositories Manager Subsystem (SMRR)

A problem that our SOW must solve is that it has not an adequate bandwidth to transport web objects. One of the ways to tackle this deficiency is trying to bring users closer to the information they'll require. In our proposal we'll consider the remote

repositories as a cache in the web, with the capacity to store objects close to the users, so it's not necessary to move an object from its original location every time it's needed. The most critical aspect in designing these caches in the web is that the storage size is finite. That's why a policy must be used to guarantee optimization of the reduced space, every time old objects have to be replaced with new ones. Other basic elements are the mechanisms to keep the information coherent in these repositories. Among the advantages of having a cache in the web are the reduction of response times (when downloading web objects), the elimination of traffic in the network (by having the information the user would need close to him), among others. The cache in the web works as follows: if a user requests an object, the cache makes a local copy; when the same object is required again, the cache shows its available copy, this way it's not necessary to request it again from the original server.

3.4 Web Object Manager Subsystem (SMOW)

This subsystem manages every kind of web object, particularly the mobile objects and those that have to be replicated. Mobile objects are those that move through Internet sites following certain criteria. Among the advantages offered by the migration of objects in the Internet, we find the reduction of communication costs, and a more fault tolerant system [5, 6]. We use the idea of traces in the definition of mobile object administration policies. When the object moves it leaves a trace of its movements, which is used to locate it when is requested by a user. In addition, the trace can evaporate each certain time. The intention behind the disappearing trail is not to leave confusing traces that might complicate the search of an object at any given time. On the other hand, web objects can be replicated, in this way the system must establish administration policies to guarantee accessibility and best possible location of the replicas in the web.

3.5 Communities Manager Subsystem (SMC)

The nodes will interact in the SOW through mechanisms of request/reply and negotiations. Those nodes exhibiting functional and behavioral affinity are able to associate themselves dynamically to form communities. A node in the SOW is defined according to its abilities and behavior. Those are the elements considered as node features, which as a whole (with its resources and functions) are affiliated to a given community. Our SOW will treat communities from an emerging point of view, that is, communities that emerge and adapt to their environment, that self-organize according to the requirements; without a central control and showing a global order (the group of nodes works very efficiently and with a good structure) [6]. The SMC will have protocols that manage the creation, change and elimination of communities and, in general, the incorporation, elimination or migration of nodes between communities.

4 Design of Our SOW Like a Multiagent System

Each subsystem of the SOW is a MAS. To describe each subsytem like a MAS we will use the MASINA methodology [7].

4.1 Design of the SMR

In the SMR the operations are coordinated by four agents:

- **Interface with the User:** It receives requests of the users, to send them to the reasoning system, and to send the results of the requests to the users.
- **Reasoning System:** It is the main agent of the system. It receives requirements of the interface for processing them, with the purpose of giving an suitable answer to them. It has an inference engine to coordinate the mechanisms of location, configuration and allocation of services required by the users.
- **Services Search:** This agent has like objective the search of services that are required by the inference engine.
- **Versions Manager:** This agent has a *self-configuration module*, which handles the versions of the services found, in order to produce the configuration of the response according to the requirement of the users.

4.2 Design of the SMRL

In the SMRL the operations are coordinated by five agents:

- **Search Coordinator:** It receives the requests of search of services of other subsystems. In addition, It controls the decomposition, classification and distribution of the requests of search of local service.
- **Local Resources Manager:** It manages the resources of hardware and software existing in the node. Also, it updates the catalogue of the local resources.
- **Local Information Manager:** It provides the local information required by the search coordinator. On the other hand, it updates the local information, and also maintains the consistent information after any process of update.
- **I/O Optimizer:** It establises policies to make an efficient use of the capacity available in the devices. In addition, it establishes techniques of optimization and efficient access to the devices.
- **Traces Manager:** it handles the traces at local level. This agent allows the SMRL to participate in the process of creation, reinforcing, evaporation and elimination of traces, altogether with the SMRR and the SMOW.

4.3 Design of the SMRR

In the SMRR the operations are coordinated by two agents:

- **Search:** this agent searches the replicas and the objects that are in the SMRR and that are required by the SMR.
- **Administrator:** this agent maintains the repositories of the SMRR. For this, it has three functions: a) when the SMRR needs to store a new replica or a referenciable object and there are not space in the RR, it eliminates less used objects. b) It enters, eliminates and updates replicas and referenciables objects in the RR. c) It verify the consistency of the information that is in the RR.

4.4 Design of the SMOW

In the SMOW the operations are coordinated by two agents:

- **Replicated Objects Administrator:** this agent transfers the replicas of the objects in the SOW.
- **Migrated Objects Administrator:** this agent migrates the objects Web in the SOW and creates the traces.

4.5 Design of the SMC

In the SMC the operations are coordinated by two agents:

- **Communities Manager:** it manages the existing communities in the SOW. A new community that emerges is characterized before being added to the SOW. In addition, it allows locating communities with certain characteristics or profile, and describing a node before it can be incorporated to a given community.
- **Search Manager:** It receives requests of search of services of a SMR or another SMC, and processes them (It makes the search of a given service).

5 Coordination, Communication and Intelligence Models

The design of each one of the subsystems of the SOW according to the MASINA methodology [7], allows to show the existing communications (speech acts) between the agents. These can be shown through a diagram of interaction of UML. For example, the conversation *to update community* [6], of the agent communities manager of the SMC, is shown in figure 1. In addition, MASINA allows the description of the existing intelligence in some of the agents through the intelligence model. For the same previous agent of the SMC we describe its intelligent model in table 1.

Table 1. Mechanism of Reasoning of agent CA

	REASONING MECHANISM
INFORMATION SOURCE	Obtained results of the group of nodes.
ACTIVATION SOURCE	Request to incorporate or to transfer node.
TYPE OF INFERENCE	Deductive, Inductive.
REASONING STRATEGY	To coordinate activities related to the group of nodes, update and creation of communities, through tools of recognition of patterns based on neuronal networks and fuzzy logic (Engine of fuzzy rules).

6 Conclusions

In spite of the great efforts put on so far, the existing operating system and traditional Middleware architectures aren't prepared to offer an efficient management of web resources. We propose a WOS model that tries to address these issues. The WOS proposed is made up of four subsystems that will carry out a series of coordinated functions that allows an efficient use of Internet resources, in spite of its dynamic features, high mobility and heterogeneity.

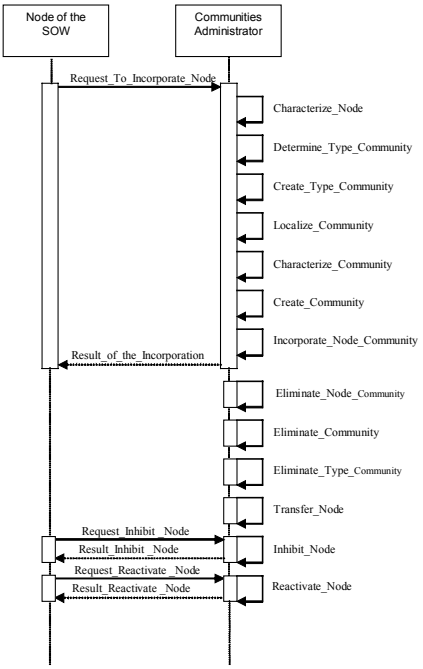

Fig. 1. Conversation: *To Update Community* in the SMC

References

1. Vahdat A., Belani E., Eastham P., Yoshikawa C.: WebOS: Operating System Services for Wide Area Applications". Proc. of Seventh IEEE Symposium on High Performance Distributed Systems (1998) 52-63.
2. Kropf P.: Overview of the WOS Project. Proc. of Advanced Simulation Technologies Conferences, High Performance Computing (1999) 350-356.
3. Casanova, H., Dongarra, J.: NetSolve: A Network-Enabled Server for Solving Computational Science Problems. International Journal of Supercomputer Applications and High Performance Computing, **3** (1997) 212-223.
4. Morgan, S.: Jini to the Rescue. *IEEE Spectrum*, **37** (2000) 44-49.
5. O'Reilly, T.: Building the Internet Operating System. Proc. of O'Reilly Emerging Technology Conference, (2002) 1-2.
6. Aguilar, J., Ferrer, E., Perozo, N., Vizcarrondo, J.: Arquitectura de un Sistema Operativo Web", *Gerencia, Tecnología*, *Informática (electronic journal: www.cidlisuis.org/aedo),* Instituto Tecnológico Iberoamericano de Colombia, **2** (2003).
7. Aguilar, J.: Especificación Detallada de los Agentes del SCDIA – MASINA CommonKADS. Reporte Técnico Proyecto Agenda Petróleo 9700381, Universidad de los Andes (2003).