Web Operating System

A. Mufti, K. Salah*
Department of Information and Computer Science
King Fahd University of Petroleum & Minerals
KFUPM # 1067, Dhahran 31261, Saudi Arabia.
Email: *muftiah@aramco.com.sa, salah@ccse.kfupm.edu.sa**

Abstract

One of the hottest topics that emerged these days between the area of Internet and distrubted computing and the area of operating system is Web Operating System (WOS). The objective of WOS is to deliver the full benefit of the World Wide Web. The WOS will support geographically distributed, highly available, incrementally scalable, and dynamically reconfiguring applications. WOS will include mechanisms for resource discovery, resource collaboration, persistent storage, remote process execution, resource management, authentication and security. This paper presents an overview of a typical WOS. It describes the WOS process, components, communication protocols, and resources. Additionally, the paper discusses all the resolved and unresolved issues and difficulties surrounding the implementation and design of WOS

1 Introduction

Development of a new single operating system enabling global computing is a hot issue these days. Such an operating system is called the Web Operating System, or WOS. Major Internet users use WOS to download files, execute of servers programs remotely, fetching client scripts, etc. The common model of these services consists of client-server or master-slave configuration with a network as a transportation media. WOS offers variety of services. These services could be software or hardware (computation, communication channels, storage capacity, specialized drivers, etc.).

The use of web resources is highly motivated by different reasons. These include reliability, availability, fault tolerance, load sharing, function sharing, and performance aggregation. The many various real applications exhibit very different requirements. For example, 3D animation rendering is massively matrices computation. To take advantage of distributed infrastructure, mechanisms for efficient

* Corresponding Author

resource management and access are needed. However, the heterogeneous and dynamic nature of the web infrastructure ensures that it is impossible to provide a complete catalog of all resources available on the web. Therefore, new approaches are needed which take into account the inherently decentralized and dynamic properties of the Internet and distributed system in general.

In order to meet the need for such requirements, WOS has a framework for supporting applications that are geographically distributed, highly available, incrementally scalable, and dynamically reconfiguring. It will also include fetures for resource discovery, resource collaboration, persistent storage, remote process execution, resource management, authentication and security.

This paper has been organized as follows. Section 2 gives an overview of a typical WOS. The overview includes a description of WOS nodes, process, components, and communication protocols, and resources. Section 3 addresses the issues and difficulties surrounding implementing and designing WOS. And finally, Section 4 has a summary and conclusion.

2 WOS Overview

WOS is designed as a distributed system. The WOS framework enables a new paradigm for Internet services. Instead of being fixed to a single location “client workstation”, services can dynamically push parts of their responsibilities out onto Internet computing resources and all the way to the client [1]. WOS goal is to provide a platform which allows the user to benefit from the computational potential offered

by the web. It's aimed is to make available to all sites of the network resources to execute computations for which local resources are missing [1].

To account for the dynamic nature of the Internet, generalized software configuration techniques, based on demand driven technique called ***eduction*** are developed for the WOS. The kernel of a WOS node is a general ***eduction engine***, a reactive system responding to requests from users or other eduction engine. A WOS-node integrates thus client, server, and broker/trader functions. It is capable of providing a set of services, which can pass on to each other requests when appropriate. Again, because of web is dynamically changing, there exist some warehouses that associated with the WOS node provide the necessary information and components for meeting requested services. Each WOS node is using its own warehouses to store and continuously update information about the node and available services and resources. Therefore, with the above approach, the communication protocols may be seen as the glue of the WOS. Communication between WOS nodes is realized through a simple discovery/location protocol (WOSRP) and a generic service protocol (WOSP) [2].

What distinguishes one version from another are *explicit feature*s, which can come about in many different ways. Some will be changes resulting from straightforward linear development. Others will designate choices or variants, for such concepts as interface languages or host implementations.

2.1 WOS Nodes

The collection of WOS nodes constitutes the WOSNet or WOSspace. However, since the WOS is versioned system, subnets can be defined as a collection of some WOS

nodes may be defined as a particular version of the WOSNet. For example, a number of Internet services could be defined as a WOSspace “a version of the WOSNet including only these services”. Figure 1 shows the layered structure of a WOSNode.

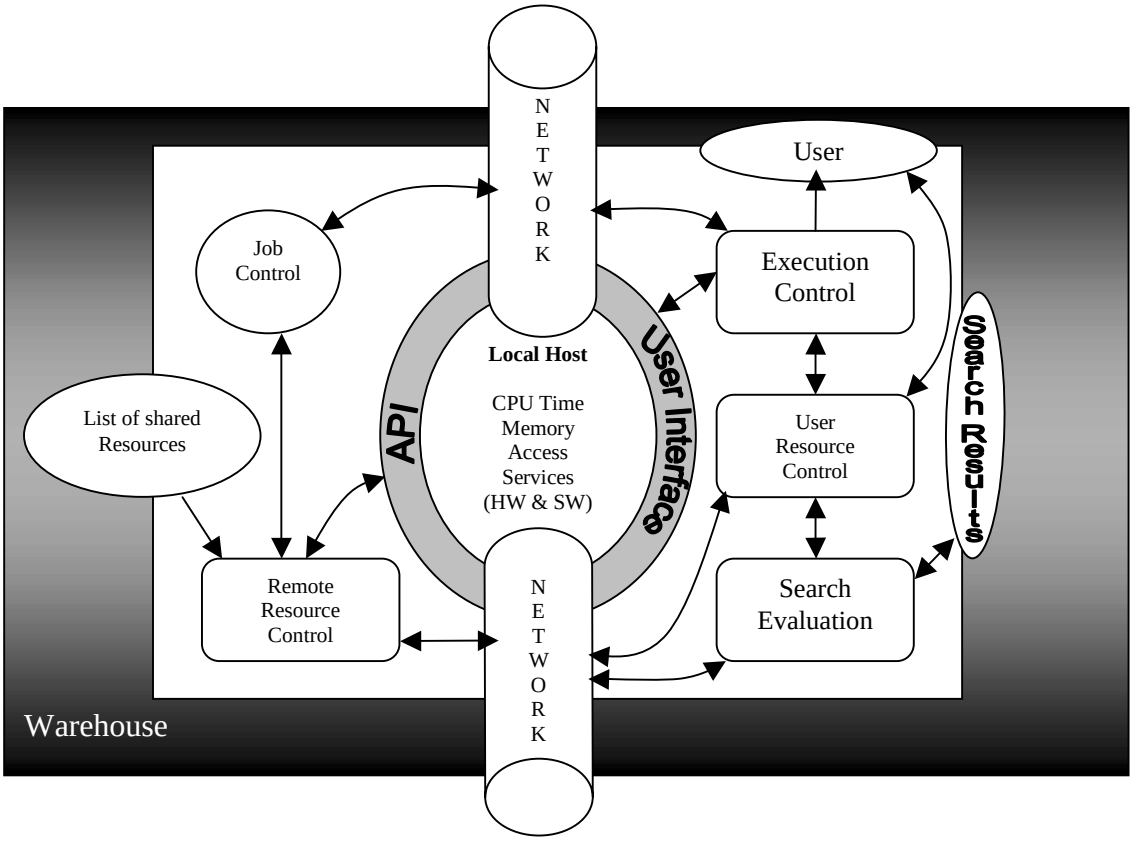

Figure1: WOSNode Architecture

The left side of this figure shows the server, while the right side represents the client features of each WOSNode. Services available on the WOSNode are described using *profile*s. Profiles describe resources with a list of key-value pairs, each pair defining a special feature of a resource. For instance, a printer has a special type (inkjet, laser etc.), is able to print black and white or color, and may handle Postscript files. Each resource also has a corresponding access-object describing its methods; e.g., for a printer, we might have self-test, economy mode, etc. That means that the user does not need to use the command line anymore. Restrictions on resource usage are described using the same data structure.

2.2 WOS Process

The WOS works as follows: A request is made by a user to run a particular program, along with specified data and quality of service parameter. The request is sent to the closest eductive engine, which might reside anywhere on the web. Upon reaction of such a request, the engine performs a lookup operation in its resources warehouses to determine if it actually has the requested program and checks whether the local machine can meet the requested quality of service parameters. The engine might refuse the service or transfer the request to one or more other eductive engine, until finally and engine accepts responsibility for the request. However, every WOS user should be able to share his or her local resources with other users. In addition, users should be able to combine and use different resources for interactive problem solving. Figure 2 shows how eductive engine collaborate with other eductive engines or other WOS nodes warehouses [1].

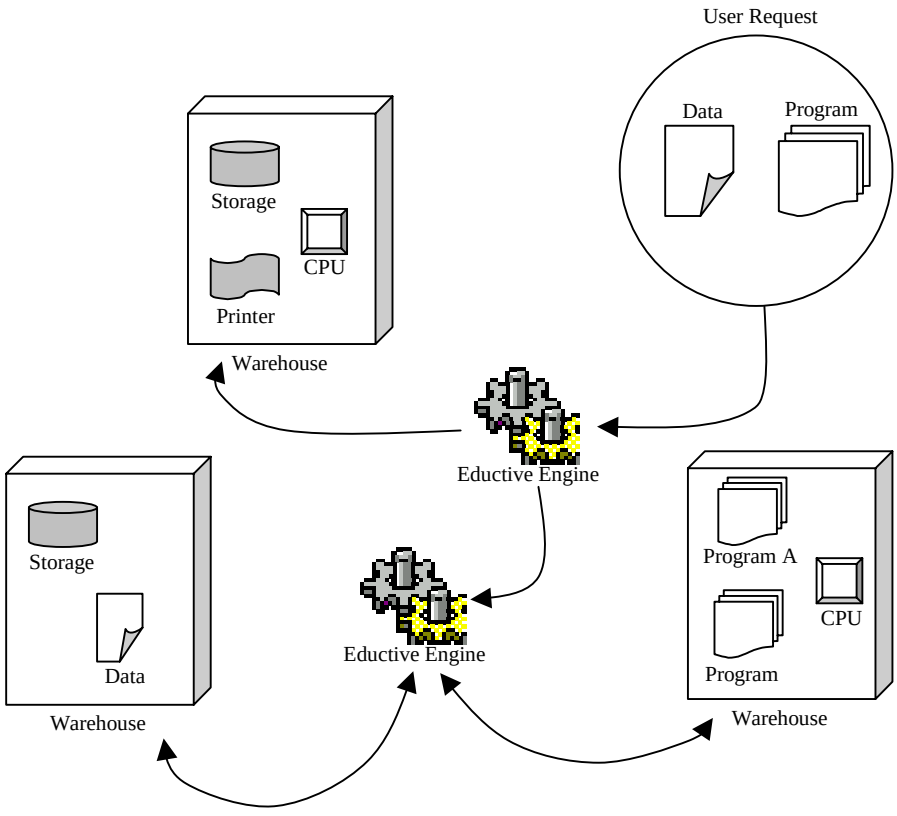

Figure 2: Eductive engines collaboration

Due to the highly distributed WOSNode on the web, a lot of communication is needed to find these resources. Thus efficient strategies for communication and searching are needed. Two typical searching strategies are available to use:

- **The broadcast strategy**

  The requesting machine submits the request to each machine in the list. Each of these machines then sends messages back to the requesting machine. Since these machines can almost work in parallel the answer will be quickly available on the requesting machine. If the list contains *n* machines, *2n* messages will be generated. *n* messages from requesting machine and *n* answers both positive and negative. Thus the network load is high. Furthermore, the broadcast implementation must be realized, hence delaying data transmission.

- **The serial request strategy**

  In this case the requesting machine sends one message containing the list of the remaining machines to one of the machines from the list. If the service is available on this machine, a positive answer is directly sent back to the requesting machine. So the generated network load is much less than in the first case. On the other hand, the respond time is much higher than in the first case and any communication problems or long transfer times directly influence the respond time [7].

2.3 WOS Components

A typical WOS consists of three major components:

1- *User Interface*: It is subdivided in three parts:

i. Profile editor: It helps the user generate profiles of resources he wants to make available for other users. It defines descriptive features of these resources. It also defines the object the remote user needs to access and the parameters for this object. All profiles are stored in the local profile warehouse.

ii. Resource editor: The restrictions for each profile are stored in the resource warehouse. These restrictions will be checked before a user can access the resources.

iii. Request menu: Provides an easy-to-use interface to resources of the WOSNet. The user can access all resources stored in the local warehouse and might also initiate a search for new resources to update the warehouse.

2- *Resource Control Unit* (RCU) accepts service requests from the user interface and contacts several known warehouses to find a WOSNode. First, the local warehouse is contacted, then other known warehouses in the WOSNet. If no service is found, a search for the requested service will be started. If an answer is found, the RCU asks for the service execution and returns the results to the user. After successful execution, the local warehouses are updated.

3- *Remote Resource Control Unit* (RRCU) accepts service requests from other WOSNodes and examines whether the execution is allowed or not. Therefore, the resource warehouse is accessed. The RRCU transmits the answer to the client-side RCU. The service execution itself is also managed by the RRCU, which contacts the resource warehouse a second time to verify access rights. After that, the service is executed and the results are passed to the client-side RCU.

2.4 WOS Protocols

The WOS has its own Communication Layer (WOSCL) and it uses two protocols. First protocol allows the selection of an appropriate version of WOS resources, the WOS Request Protocol (WOSRP). Second protocol, the WOS Protocol (WOSP), allows locating and using distributed resources, such as services, machines, etc., in a fault-tolerant manner over the Web. These protocols have been designed using TCP/UDP as transportation means [3].

2.4.1 WOSRP

At first, a WOS client will broadcast a request to all machines in its immediate neighborhood. Any WOS server being able to provide a positive answer will respond. In this case, more detailed requests may be submitted to those WOS servers. Otherwise, the WOS client broadcasts a request to all the machines at the next network level, and so on.

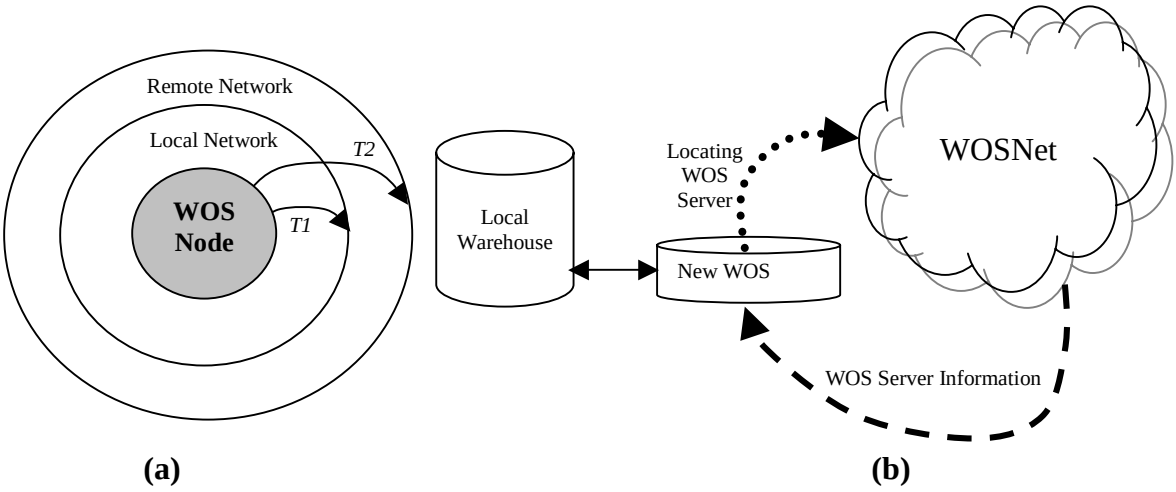

Figure 3: (a) **T1:** Initially, a message is broadcast to machines in the current local Network. **T2:** In the case where no local host responds to the initial request, a message is broadcast to hosts located in the next network level. (b) WOSRP Information Retrieval Strategy.

All the responses received are used to populate the WOS client warehouse, which is at first empty when the WOS node enters WOSNet. Figure 3 (a,b) depicts how a WOS

client could use WOSRP to obtain information about other WOS servers. WOSRP uses a “pull” philosophy, where a WOS node requests information from other nodes in its neighborhood. These messages may be lost without any disruption of service. Furthermore, WOS nodes may decide to propagate these messages to other items. Eventually, replies may be returned to the node which made the original request.

2.4.2 WOSP

It allows WOSNode administrators to implement a set of services, called a *service class*, dedicated to specific users’ needs. WOSP is in fact a generic protocol defined through a generic grammar. A specific instance of this generic grammar provides the communication support for a service class of the WOS. This specific instance is also referred to as a *version of WOSP*; its semantics depends directly on the service class supported by that version. Several versions of WOSP can cohabit on the same WOSNode. The WOSP is used to execute a service, to transmit the results of the execution, and to search the WOSNet. It is handling following interactions between WOS nodes:

1- Execution commands allowing a WOS client to use resources from another node.
2- Query commands used by a WOS client to interrogate another WOS node’s warehouse.
3- Setup commands to change the execution parameters of a WOS node [8].

2.5 WOS Resources

A resource is every thing that can be manipulated by a given host. It could have physical representation such as printers or an abstract concept such as CPU-power.

2.5.1 Resource Structure

Resource objects can be represented with following structure:

- **Presentation part** provides an interactive interface to a resource at user level.
- **Properties part** represents the state of the resource. Properties are the public fields of a class to implement a resource. It has READ and WRITE methods in order to interact with other entities.
- **Inputs** provide all public methods of a resource. Inputs and its corresponding calling interfaces are collected and published by the resource framework automatically.
- **Outputs** are stubs provided by the implementer of the resource. An output can be linked at runtime to one or more input if the interfaces match.

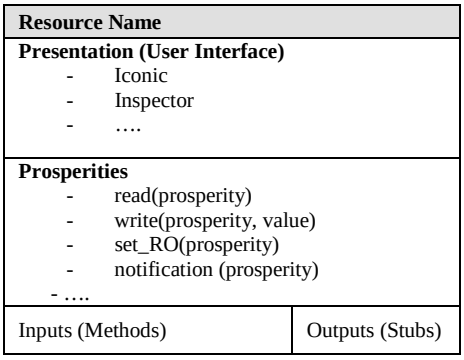

Figure 4: Resource Structure

2.5.2 Resource Components

Web Resources on WOS has been implemented on three different components:

1- Resource set as the implementation of a coordination space to mange-shared resources.

2- User interface for interaction purpose

3- Resource servers used to export local resources to the web [1].

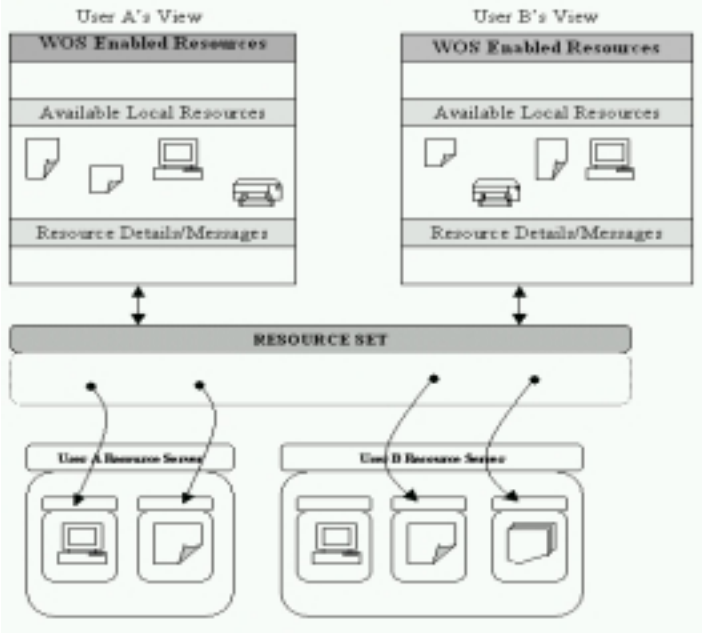

Figure 5: An interface between resource set, two-user resource server, and two-user interface.

3. WOS Difficulties and Issues

The fundamental problem that makes WOS more difficult is the heterogeneous and rapidly evolving nature of the web, and show how the concepts of software configuration and version control can be applied. A WOS should provide the following services:

- Network applications: WWW, email, video, etc.
- Computational applications: number counting, distributed simulation, etc.
- Transactional applications: banking, e-commerce, travel reservation, etc.
- Virtual entities: classrooms, companies, etc.
- Knowledge-base applications: data-mining, database, etc.
- Real time applications: process control, real time multimedia, etc.

An important issue is the way of communication among WOS nodes. It is clear that WOS resources will communicate heavily in order to execute a bit complicated

transaction. In this case normal TCP/IP connection will not fulfill the end users satisfactions from performance point of view. Therefore, a special communication protocol needs to be included on the system design in order to provide adequate communication channel among WOS nodes.

Additional consideration is the fact that the web is not just heterogeneous at the conceptual level, but also at the physical level (hardware and software that is available at different sites). Different technologies are used for all level of networking, and different machines, from personal computers to workstations to supercomputers, can all be found on the web, each with its own particular setup of operating system, supported protocols, and applications. To make a situation worse, the very basis for the web and the Internet, namely IP, is itself not static. The current standard, Ipv4, is soon to be replaced by Ipv6. However, this change will take place over long period, and the two version of IP will be used simultaneously. So even the web network level itself is heterogeneous [6].

How to get volunteers on the web where they can allow using their resources is an issue need to be addressed. In January of 1998 there were about 29,6 millions of computers connected to the Internet. This mass of processors connected together represents a very powerful parallel supercomputer with an incredible computational power. Most of these machines are only used for small interactive tasks, like the reading of electronic mail, the editing of files or just the browsing of Web pages and most of them remain idle in a significant part of the time. Thus, it seems insightful to apply this vast computing resource for solving some problems of cryptography, mathematics and computational science. In fact, there are some quite interesting

projects that use the computational power of machines that are connected to the Internet.

JET project [11] has a software infrastructure that supports parallel processing of CPU-intensive problems that can be programmed in a client/server, or master/slave, paradigm. There is a Master process that is responsible for the decomposition of the problem into small and independent tasks. The tasks are distributed among the worker processes that execute a quite simple cycle: receive a task, compute it and send the result back to the master. The Master is responsible for gathering the partial results and to merge them into the problem solution. Since every task is independent from each other, there is no need for communication between worker processes.

Other related issues, which also can affect the progress and future of WOS, may include:

- Because of large communication times, global on-line resource prediction is almost impossible. There is no global manager that can get and hold exact information about the current state of the whole system. Furthermore, the generated overhead would further increase the load of the already slow network.
- The structure of the system, especially of distant components can not be considered to be a fixed. Furthermore, it is truly heterogeneous in all aspects: structure of processing nodes, messages, used communication units, storage capacities and speeds, etc.
- In large networks with complicated structures failures or breakdowns of the system are not improbably. Specifically, failures at any intersection also result

in a system fault. Therefore, suitable reactions of the system in case of a failure must be implemented as a part of the load sharing system.

- Another general problem with a central data collection in the system is caused also by the transmission times, since the state of the system may already have changed during the transmission. The transmitted data will thus already be outdated upon arrival and therefore be invalid for any operations at the destination.
- Although most applications may be executed on any node without time limits, there might exist restrictions regarding the completion time of an application
- Off-line learning and adaptation methods cannot be used because too many parameters had to be collected. This could cause again a large overhead [5].

4. Conclusion

WOS has the potential of being an important distributed computing system for the Internet. It promises supporting applications that are geographically distributed with high reliability, security, scalability, and manageability. Some research and prototyping of WOS have been underway to overcome some of the challenges and difficulties pertaining to the design and implementation of such a system. However, many issues remain to be resolved. Some of these remaining difficult issues include volunteering user system or resources, global on-line resource prediction, and the heterogeneous nature of many Internet components and protocols.

5. References

[1] Oliver Krone, Simon Schubiger. *WebRes: Towards a Web Operating System*. Kommunikation in Verteilten Systems 1999: 418-429.

[2] Peter G Kropf. *Overview of the Web Operating System (WOS) project*, 1999

Advanced Simulation Technologies Conference (ASTC1999). San Diego, California, USA, pp.~350--356, April 1999.

[3] Peter G. Kropf, Herwig Unger, Gilbert Babin. *WOS: an Internet Computing Environment.* Quebec Canada, 19-21 Jun. 2000, Springer-Verlag, LNCS 1830, Berlin Heidelberg New York, 2000, pp1-1.

[4] Simon Schubigr, Oliver Krone. *Interactive Resource Sharing on the Web*.

[5] Herwig Unger, Peter Kropf. *Overview about the Resource Scheduling in the WOS* , Distributed Computing on the Web (DCW'98).Rostock, Germany, pp.~134--140, June 1998.

[6] Slim Ben Lamine, John Plaice, Peter Kropf. *Problems of Computing on the Web*, High Performance Computing Symposium 97, A. Tentner, ed., The Society for Computer simulation International, Atlanta, Georgia,USA, pp.~296—301, April 1997.

[7] Herwig Unger, Thomas Bohme. *Search in the WOSNet* , Distributed Computing on the Web (DCW), Rostock, Germany, 1998.

[8] Gilbert Babin. *Requirements for the implementation of WOS protocols*.

[9] Slim Ben Lamine, John Plaice. *Simultaneous multiple Versions : The Key to the WOS.* Distributed puting on the Web (DCW), Rostock, Germany, 1998.

[10] Helmar Gebert, Carsten Pitz. *Resource propagation within a Hyper-computer*.

[11] Francisco Soares, Luis Silva, Joao Silva. How to get volunteers for web base meta-computing

[12] Amin Vahdat, Eshwar Belani, Paul Eastham, Chad Yoshikawa, Thomas Anderson, David Culler, Michael Dahlin. *WebOS: Operating System Services for Wide Area Applications.* A project on Defence Advanced Research Projects Agency