http://ansinet.com/itj

ITJ

ISSN 1812-5638

INFORMATION TECHNOLOGY JOURNAL

ANSInet
Asian Network for Scientific Information
308 Lasani Town, Sargodha Road, Faisalabad - Pakistan

Information Technology Journal 4 (4): 360-366, 2005
ISSN 1812-5638

Web Operating Systems and Computing on the Web

Najib A. Kofahi and Ahmad T. Al-Taani
Department of Computer Sciences, Yarmouk University, Irbid, Jordan

Abstract: Web computing was and still considered to be one of the areas that capture the attention of most researchers. This is probably due to the popularity of the Internet. The objective of this study is to address a few different issues in connection to Web Operating System (WOS) and outlined some of the grievous difficulties with computing on the web. The issues discussed in this study were searching, fault tolerance, load balancing and security. Also a perspicuous description of how the WOS works was presented. The study discussed the infrastructure, communication framework, protocols as well as service classes of WOSs. It is found from the study that though the WOS promises to unravel most of the problems facing computing in the web, it is not yet fully operational due to the fact that some issues are yet to be resolved.

Key words: Web Computing, Web Operating System, Eductive Engine, WOS protocols, RD-WOSP, HP-WOSP

INTRODUCTION

Perhaps the Internet is the most widely used technology since the advent of the computer. It has created a mass communication and information resource across the world that is easy and inexpensive to use and effective in commercial, social and personal environments. It is clear that the internet has become part of us; we simply cannot function without it. With the rapid development of the web, it is increasingly clear that an operating system is needed so that operation of the web can be improved[1]. In view of this, many researches have been conducted in order to come up with an operating system for the web, call it WOS.

The problem of computing on the web has partially been addressed in the work of metacomputing. Metacomputing is a method by which an illusion is given to the user that he is using a single powerful computer whereas in the real sense he is using a collection of computers. This is achieved by using a software layer (middleware) which transforms a collection of independent resources into a single, virtual and coherent machine. One important feature of metacomputing is its dynamic nature; each node has its own complete operating system and may join or leave the environment whenever it desires[2]. However, the web is more than just a metacomputer, in that it does not have a complete catalog of all the available resources on the web. As a matter of fact, having such a catalog is near to impossibility because of the extreme dynamic and heterogeneous nature of the web and coupled with the fact that we are dealing with a network that goes across the whole world[1].

Computing on the WEB is not as easy as it may sound, partly because the WEB is incredibly growing and coupled with the fact that it encompasses lots of heterogeneous systems. Moreover, the resources available on the WEB may also be versioned. In other words, different versions of the same resource or application exist on the WEB so we are faced with the problem of searching for the appropriate version required for the respective requests from clients[3].

A lot of research has been undertaken in this area and several approaches were developed in order to integrate the computational resources available over the Internet[4,5]. Some of these approaches include the Jini architecture proposed by SUN Microsystems[5-8]. Other approaches include Linda, PVM, MPI and Netsolve etc.[5].

There are lots of problems revolving around computing on the web. In this paper, the authors address some of these problems. This study was motivated because computing on the Web is very complex and some of the problems revolving around such kind of computing need to be addressed properly.

THE WOS

When dealing with resources on a private network, it is easy to control how user requests are served. This is not easy to achieve in the case of the web due to its heterogeneous and dynamic nature and also owing to the fact that the available resources may be versioned. In order to unravel the problems surrounding computing on the web, the WOS concept was developed several years ago. In other words, the

Corresponding Author: Dr. Ahmad T. Al-Taani, Department of Computer Sciences, Yarmouk University, Irbid, Jordan

WOS was developed to provide a user with the possibility to request a service without having to have prior knowledge of the service (i.e. where it is available, at what cost, under which constraints etc.)[2,4].

The main aim of WOS is to supply users with adequate tools that allow for the implementation of specific services and to provide them with great flexibility in the semantic of their services[9,10]. One interesting thing about the WOS is that services appear and disappear depending on the context; this implies that services and environments have to adapt to their context.

The Web Operating System can be considered as a virtual operating system that supports efficient computing on the web and effectively manage such kind of processing[11,12]. The WOS is designed to enable the user to make a request for a service without having any prior knowledge about the service and to have it accomplished within his desired parameters. The universality of WOS and the fact that it provides an open access makes it an attractive environment for metacomputing[2]. Examples of WOS include: Alteon WebOS version 9.0 (Nortel networks) and JMX Managed Beans (MBeans for short).

Communication framework/protocols: The web provides the user not only with informational resources but also with computational ones which are under utilization; because the web is dynamic, nodes are frequently added and removed[11]. For the resources to be utilized accordingly there is a need for effective communicational structure. In order to achieve this, two protocols were developed: WOS Request Protocol (WOSRP) and WOS Protocol (WOSP). The WOSRP is the version independent layer, which looks up WOS servers and compares its version to the others to find compatible ones. The WOSRP iteratively broadcasts messages beginning with the nearest networks to networks further away to build a view of its environment. Once discovery is completed, the more powerful WOSP is then used between mutual compatible nodes[13]. Figure 1 shows the four major components of WOS[5].

Graphical User Interface (GUI): provides an interface or a means by which the user interacts with the system and it is divided into three parts, the profile editor, the resource editor and the request menu. Resources are described through a profile. The profile editor helps the user to generate profiles of resources he wants to make available for other users. The resource editor provides means by which resources can be selected and used. The third part of the GUI, the request menu, provides an easy-to-use interface to resources of the WOSNet.

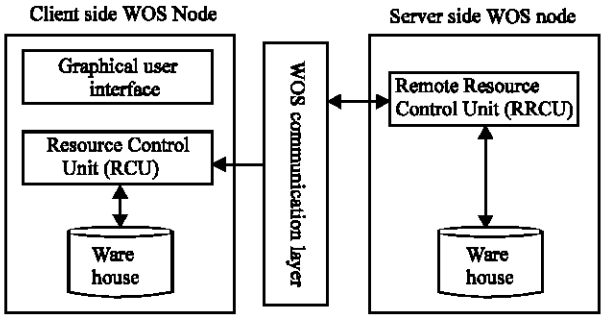

Fig. 1: Communicational framework

Resource Control Unit (RCU): accepts service requests from the user interface and contacts several known warehouses to find a WOS Node, where the requested service can be executed. The first thing that happens is that the local warehouse is contacted, then other known warehouses in the WOSNet. 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.

Remote Resource Control Unit (RRCU): accepts service requests from other WOS Nodes 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 as shown Fig. 1. 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.

WOS Communication Layer (WOSCL): uses two level approach. The first layer is the WOS Protocol and the second layer is the WOS Request Protocol.

WOSP service classes: The WOS also allows the administrator of the WOS nodes to add and broadcasts the presence of any new service class that correspond to the specific need of a particular user without having to reinstall WOS in its entirety[2]. Two service classes are discussed in this section.

RD-WOSP (The Resource Discovery WOSP): One of the most challenging services that WOS must provide is the mapping of the parallel and distributed applications onto the target machine. In order to achieve this, the metacomputing environment should be able to localize the resources requested by the different modules of the parallel and distributed application.

The following are the three major services provided by the RD-WOSP[2]:

Discovery service: this service collects and makes available the set of WOS nodes providing the resources suitable for the execution of the user needs.

Reservation service: provides the means by which resources are reserved and this normally done in connection to what the outcome of the discovery is. The reservation may and may not be accepted.

Setup service: triggers the application execution. Normally, the requesting WOS node sends a command to every node to start the execution at the desired time.

HP-WOSP (The high performance WOSP): Some of the computations carried out on the internet pose high performance constraints. High performance constraints are expressed in terms of CPU performance, bandwidth and time latency of the network, etc.[2]. This subsection gives brief description of how the HP-WOSP works. Similar to RD-WOSP, the HP-WOSP service class consists of three services: HP-WOSP (Discovery, application), HP-WOSP (Reservation, application), HP-WOSP (SetUp, module).

HP application is represented by a tree, called the Granularity Tree (GT) (Fig. 2), where each vertex can be recursively decomposed until all the vertices represent atomic sequential processes. The root represents the entire application and the leaves represent the elementary sequential communicating processes. The intermediate vertices are some aggregation of the elementary sequential processes. Edges between a given vertex and its children represent the decomposition process[2]. Each vertex in the GT represents a module and each vertex contains all the resources it needs for its computation to be carried out. With the GT construct, the HP-WOSP discovery service class is used for locating adequate WOS nodes able to execute a subset of the Granularity Tree vertices.

Once a suitable WOS node is found, mapping of the vertex to the found node is done. This mapping must take into consideration the current workload of the WOS node as well as the traffic over the Network, in order to meet the performances (HP resources) required by the corresponding vertex being assigned[2]. The mapping of the GT vertices to WOS nodes is made from top to bottom. The root of the GT represents the whole HP application. The HP-WOSP discovery service identifies a WOS node that can provide these resources and assign the whole application to it. The mapping algorithm proceeds recursively to the assignment of the children vertices until each covered vertex is allocated to a WOS node. Obviously, the deeper a vertex is in the GT, the higher is its chances of getting a WOS node with resources required to service its request[2].

The kernel of WOS: The kernel of the WOS is designed as a general eductive engine, which is a reactive system responding to requests or queries from users or other

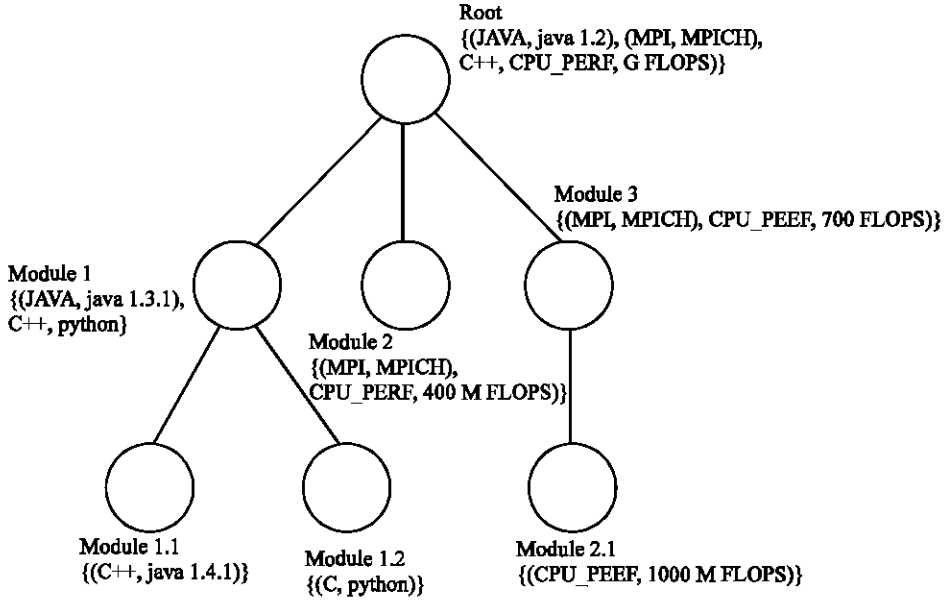

Fig. 2: Granularity tree

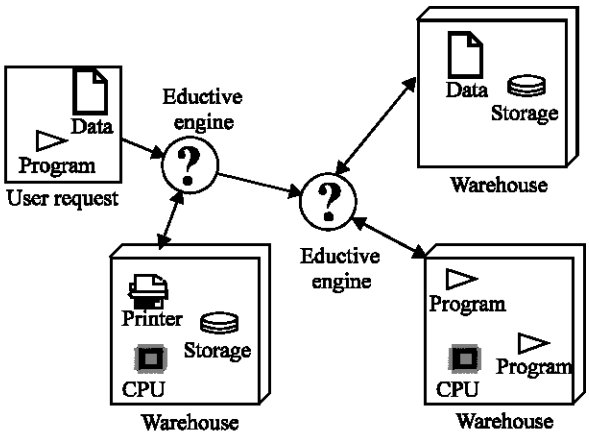

Fig. 3: The kernel of WOS

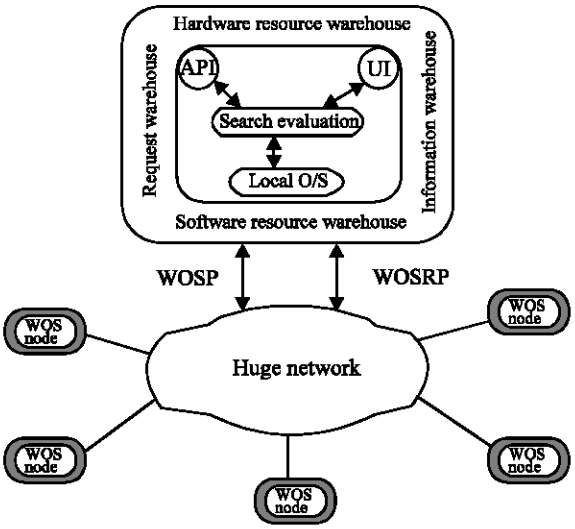

Fig. 4: The WOS infrastructure

eductive engines. The eductive engine fulfills these requests using its warehouses if the required resources are available in the warehouse (Fig. 3)[1].

Whenever a user initiates a request to run a particular application, the eductive engine of the local system decide whether it is capable of dealing with the request. Two factors determine the capability of this system: the presence of the required resources in the warehouse of the eductive system and the load of the current engine. It might be overloaded and the service might have too high priority to wait. If the eductive engine is incapable of handling the request, it passes it to the nearest eductive engine which in turn decides whether it can handle the request or not. This process continues until a suitable engine is found which accepts the responsibility of executing the request or else a negative acknowledgment is send to the requesting user[1].

Infrastructure of WOS: Across the web, there are several nodes, each node can be considered as a client when requesting for a service and a server when providing service. The collection of WOS nodes form what is known as the WOSNet or the WOSspace. Services available on the WOSNet are described using profiles.

From Fig. 4, whenever a node initiates a request, it goes to the search evaluation unit it is decided that the node can handle the request; the API provides a level of abstraction between the application and the kernel of WOS.

The API's main task is the translation of parameter lists from one format to another. The search continues to other nodes in the network and the things that give the platform on which servers are searched, located and used effectively are WOSP and WOSRP.

INTRICACIES WITH COMPUTING ON THE WEB

What distinguishes computing on the Internet from classical distributed systems is the fact that it is highly heterogeneous and also there is no complete catalog of all available resources. Different issues with respect to WOS[5] are discussed here.

Searching for appropriate resource/service: In the context of WOS, there are two major searching strategies: the broadcast strategy and the serial strategy[14].

The broadcast approach: This provides a sort of parallel approach to the searching process whereby the requesting machine broadcast the request to all machines on the list. Each of these machines then sends back a message to the requesting machine, the message could be a positive respond if the required service is found, or a negative one if the required service does not exist. The problem with this approach is that there is high utilization of the network, this result in high network load. In addition, the broadcast implementation is realized through a multiple point to point communication scheme, thus resulting in data transmission delay[14].

The serial request strategy: In this approach the searching process is done sequentially. This means that the requesting machine sends a message to only one of the machines from the list. The service is provided by this machine if it has the service, otherwise the request will automatically be directed to one of the machines in the list included in the message. This procedure continues until

the service is found or no machine in the list has the service[11,14]. One good thing about this approach is that the respond time is much higher than in the broadcast strategy. However, any communication problems of long transfer times directly affect the respond time.

Boehm and Unger[14] presented a good model of searching which looks at the whole concept from the perspective tree structure. They represented the searching strategy using a directed rooted tree with vertex set L U {m0}, where m0 represent the root and L is the list of the available machines. The search is carried out in the following way. First, m0 broadcasts the request to its direct successors. When a machine m' (m' is an end vertex of the tree) receives the request, it searches whether the service is available or not. If the service is available, it immediately sends a positive answer back to m0. Otherwise, the request is directed to its direct successors. The final result of the search process is obtained when either a positive answer is received by m0 or a negative is received m0 from all end vertices.

Computer networks are not very reliable because the communication link may be down or the system that has the service may have crashed. In such a scenario, the requesting system might be waiting endlessly for the service to be provided. In order to unravel such kind of problems, two types of acknowledgment messages were suggested by Gilbert *et al.*[11]. The reception acknowledgment (RACK) confirms proper reception of the message by the next machine in the list while the termination acknowledgment (TACK) indicates that either all the WOS servers in the list were visited or a suitable candidate was found as shown below.

Security issues in WOS: With the enormous amount of electronic transactions carried out on the internet, it is becoming increasing important to provide secured means for such transactions because the safety of the transactions using electronic means is of very important[7]. 'Secured means' implies to provide confidentiality, integrity, availability and authenticity. Confidentiality requires that information in a system be accessible for reading only by authorized parties. Integrity requires that system assets be modified by authorized parties only. The availability security requirement ensures that the assets of a system are available to authorized parties and finally the authenticity requirement ensures that only verified users access the system[7]. There are different kinds of security problems effect transactions carried out on the internet. Some of these problems are:

- Eavesdropping: Intercepting and reading messages for other users.
- Masquerading: Sending and receiving messages using another user's account.
- Message Tampering: Intercepting and altering messages for other parties.
- Replaying: Passively capturing previously sent messages and subsequently retransmitting it in order to produce an unauthorized effect.
- Infiltration: Abusing a user's account in order to run hostile or malicious programs
- Denial-of-service: Preventing authorized users from accessing the various resources of the system. In order to overcome these problems, the architecture of the WOS is designed in such a way that it can support the following methods of security.
- Authentication: Basic protection is established on the authentic usernames and passwords. System does not allow short or simple passwords. None of the features of the system or specific module can be accessed if the user is not properly logged in to the system.
- Limited number of login attempts: In the configuration options of WOS we can specify the maximal number of login attempts and consequently separate a usual login attempt that failed and external attempt to breach the system. External attempts are defined as attacks from a specific location on the network including the intranet. When a limited number of attempts are made, data about the location of the attack are recorded and login attempts from that location are blocked in the future. The administrator of course gets the SMS or e-mail notification immediately.
- Limited inactivity time: If the user is inactive for a specified time the system automatically signs the user off. This protection measure is very useful if the user leaves the workplace and remains logged in to the system. This logout time can be specified through administration options.
- Application log: All important activities on the system (specified by the client) including the activities of the system administrator can be recorded into the Log files.
- Encryption: A lot of information on our system is encrypted. Usernames or any other sensitive data can be encrypted. This means that any external control or attack cannot access the database directly.

Fault tolerance: Searches for resources on WOS servers maintain sequential search chains. These chains are created on demand by the WOS client, using information stored in its local warehouse[11].

From Fig. 5 it becomes clear that for the search to be effective, each node in every chain must be reached until

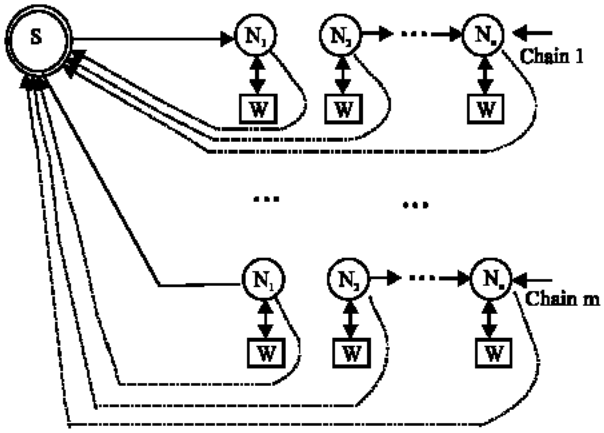

Fig. 5: Sequential search chains

the required resources is found. This calls for a fault-tolerant approach to overcome network or server breakdowns. There are two types of failures: the network breaks down or times out while the message is being transmitted, or a WOS is incapable of serving a request after receiving it. The TACK and RACK protocols are used to overcome these problems.

If node i, in the chain, receives a request from node i-1, it first sends a TACK before a time-out of length t expires on node i-1. Node i then checks if it is capable of providing the resources. If this node can provide the resource, it replies to the WOS client making the request and send a TACK to node i-1. Otherwise, the search continues to the next node in the chain in the same manner[11].

Load balancing: Load balancing is a method of distributing processing and communication activities evenly across a computer network so that no single device is overwhelmed[1-5]. Load balancing is especially important for networks where it is difficult to predict the number of requests that will be issued to a server; this is the more reason why load balancing should be properly addressed with regards to computing on the web.

In this section we will discuss how load balancing can be implemented across the Internet in order to achieve efficient performance. Figure 6 shows a load balancer that distributes requests among servers. Requests are directed from the load balancer to one of several web servers. The Web servers may access one or more databases for creating content[7]. Some of the approaches for implementing load balancing on the web are discussed.

The DNS approach: A name associated with a web site can map to multiple IP addresses, each associated with a

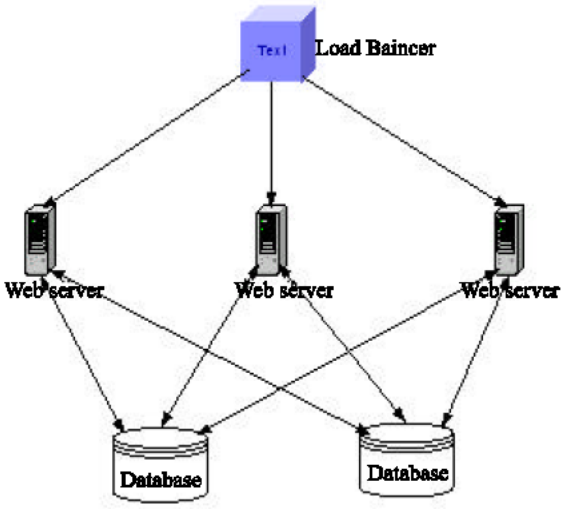

Fig. 6: Architecture of a scalable website

different web server. DNS servers can select one of these servers using certain methods such as round robin. Other approaches, which can be used for DNS load balancing, also exist. For instance the DNS server can use information about the number of requests per unit time sent to a web site as well as geographic information[7]. Load balancing can also be implemented at the client side (i.e. client-based technique). A number of client-based techniques for load balancing request on the web have been proposed, one of such approaches is the client's DNS[7]. The Connection Router is another method for load balancing which uses a connection router in front of several back-end servers. Connection router hides the IP addresses of the back-end servers.

The connection router: One other method through which load balancing can be achieved is by using a connection router in front of several back-end servers. Connection router hides the IP addresses of the back-end servers. That way, IP addresses of individual servers won't be cached, eliminating the problem experienced with DNS load balancing. Connection routing can be used in combination with DNS routing for handling large numbers of requests. A DNS server can route requests to multiple connection routers. The DNS server provides coarse-grained load balancing, while the connection routers provide finer grained load balancing. Connection routers also simplify the management of a Web site because back-end servers can be added and removed transparently[7].

CONCLUSIONS AND FUTURE WORK

With complete implementation of WOS over the Internet, computing on the web will be as simple as

computing on a private Local Area Network. WOS has the potential of supporting high performance application that are distributed over the global network with high degree of reliability and security.

Searching for nodes capable of servicing requests on the web requires for effective searching techniques especially due to the heterogeneous nature of the web. The Broadcast and Serial request approaches are two common techniques used for searching WOS nodes capable of providing WOS services. Locating the WOS node in order to service a request is just the beginning of the execution process; effective means of ensuring that the whole system has some degree of fault tolerance is required. Two acknowledgments: the Reception acknowledgment (RACK) and the Termination Acknowledgment (TACK) are used in order to make computing on the web fault tolerable.

When it comes to effective computing, two other issue that are of high significance are load balancing and security. A couple of security issues with respect to computing on the web are discussed and two load balancing techniques (the DNS approach and the connection router approach) are presented in this paper. Though WOS promises to unravel most of the problems revolving around computing on the web, it is not yet fully operational on the Internet due to the fact that some issues are yet to be resolved. For instance effective implementation of RD-WOSP and HP-WOSP, the issues of resource versions, heterogeneous nature of the web etc need to be effectively handled. Some of the intricacies surrounding computing on the web need to be further studied and analyzed with a view to providing better solutions than the existing ones.

ACKNOWLEDGMENT

The publication of this research was supported by Yarmouk University Research Council

REFERENCES

1. Lamine, S.B., J. Plaice and P. Kropf, 1997. Problems of computing on the web. SCS A. Tentner, Eds., High Performance Computing Symposium, Atlanta, Ga, pp: 296-301.
2. Abdennadher, N., G. Babin and P. Kropf, 2001. A WOS-based solution for high performance computing. IEEE-CCGRID 2001, Brisbane, Australia, pp: 568-573.
3. Vahdat, A., T. Anderson, M. Dahlin, D. Culler, E. Belani, P. Eastham and C. Yoshikawa, 1998. WebOS: Operating system services for wide area applications. 7th IEEE Symposium on High Performance Distributed Systems, Chicago, IL.
4. Unger, H., 2000. Distributed Resource Location Management in the Web Operating System. In SCS A. Tentner, Ed., High Performance Computing 2000 (ASTC), Washington, DC, SCS Intl., pp: 213-218.
5. Kropf, P., H. Unger and G. Babin, 2000. WOS: An internet computing environment. Workshop on Ubiquitous Computing, PACT 2000. IEEE Intl. Conf. Parallel Archit. Compilation Technique Philadelphia, PA, October 19th, 2000, pp: 14-22.
6. Sun Microsystems Inc., Jini Specification. www.javasoft.com/products/jini/specs, 1999.
7. Saliba, R., G. Babin and P. Kropf, 2002. SecAdvise: A Security Mechanism Advisor. In Distributed Communities on the Web. (DCW 2002), LNCS 2468, Sydney, Australia, April 2002, Springer, Verlag, pp: 35-40.
8. Rigole, P., T. Holvoet and Y. Berbers, 2002. Using Jini to integrate home automation in a distributed software. 4th Intl. Conf. Distributed Commun. Web, Sydney, Australia, April 3-5, 2002, pp: 291-304.
9. Kropf, P., J. Plaice and H. Unger, 1997. Towards a Web Operating System. WebNet '97, Toronto, 1997. Association for the Advancement of Computing in Ed., Charlottesville USA, 1997, pp: CD-ROM.
10. Mufti, A. and K. Salah, 2002. Web operating system. Workshop on Information and Computer Science, KFUPM, Dhahran, Saudi Arabia, March 17-18, 2002, pp: 279-287.
11. Gilbert, B., P. Kropf and H. Unger, 1998. Two-level communication protocol for a Web Operating System (WOS). IEEE 24th EUROMICRO workshop on Network computing,(EUROMICRO'98) Sweden, August 25-27, 2: 939-944.
12. Arun, A.L., E. Nahum, A. Shaikh and R. Tewari, 2002. Enhancing Web Performance. Proc. IFIP 17th World Computer Congress-TC6 Stream on Commun. Sys., The State of the Art, August 2002, pp: 95-126.
13. Abdennadher, N., G. Babin, P. Kropf and P. Kuonen, 2000. A dynamically configurable environment for high performance computing. Advanced Simulation Technologies Conference (ASTC), Washington, USA, April 2000, pp: 236-241.
14. Boehm, T. and H. Unger, 1998. Search in Tthe WOSNet. In Distributed Computing on the Web (DCW'98), Rostock, Germany,1998, pp: 141-142.
15. Kofahi, N.A. and Q. Rahman, 2004. Empirical Study of Variable Granularity and Global Centralized Load Balancing Algorithms. Proc. Intl. Conf. Parallel and Distributed Proc. Techniques and Applications (PDPTA'04), Las Vegas, Nevada, USA, June 21-24, 2004, CSREA Press, 1: 283-288.