OVERVIEW OF THE WOS PROJECT

Peter G. Kropf
Département d'informatique
Université Laval
Sainte-Foy (Québec) Canada G1K 7P4
Email: kropf@acm.org

KEY WORDS

Web operating system, software configuration, eduction, resource management, operating system services, global computing, communication protocols.

ABSTRACT

We discuss the Web Operating System (WOS^{TM}) approach to global computing. We show that the heterogeneous and dynamic nature of the Web or Internet makes it impossible to define a fixed set of operating system services, usable for all services. Rather, we propose that generalized software configuration techniques, based on a demand-driven technique called eduction, can be used to define versions of a Web Operating System that can be built in an incremental manner. This net-centric approach considers communication as the central issue as opposed to the common notion of central servers. We present the communication mechanisms as well as the WOS node architecture and components to achieve this.

INTRODUCTION

The rapid development of networked and mobile computing, as demonstrated with the ever growing Internet (or Web) has lead to a global infrastructure, as well as to the introduction of new IT functionality. The presently available tools essentially allow users to download files, execute remote pre-defined scripts, fetch mobile code to be run locally or develop and run distributed applications within specialized metacomputing environments. The underlying model for those tools consists mostly in a client-server or master-slave setup with the network as the transport means. However, the role of the network is evolving towards the service delivering platform where services are offered through delivery contracts. A service may be understood in this context as a piece of software or hardware (computation or storage capacity, communication channels, specialized drivers, etc.). From a user's point of view, a service may be of any kind: the World Wide Web, conferencing, auctions, database search, mail, media, scientific applications or simply the translation of a document from one format to another.

The use of widely distributed computing resources is motivated by various reasons such as load sharing, performance aggregation (including the exploitation of workstation idle time), reliability, availability and fault tolerance, function sharing and data sharing. The many different imaginable applications (open markets/electronic commerce, workflow management, collaborative work, scientific computing, tele-immersion, smart instrumentation, etc.) exhibit very different requirements. For example, scientific computing encompasses massively parallel computing, large data sets and real time visualization. Workflow management, as e.g. in the automotive industry, includes CAD/CAE, multiple data formats and multiple programs. To take advantage of the global distributed infrastructure, mechanisms for efficient resource management and access are needed. This central problem of resource management is a classical function of the operating system. Consequently, one could imagine the development of a new single operating system enabling global computing. However, the heterogeneous and dynamic nature of this infrastructure ensure that it is impossible to provide a complete catalog of all the resources available. Moreover, the overhead costs associated could dominate the savings and it is questionable whether such an operating system would even be useful, considering the different levels of granularity of service that are necessary to be provided. Therefore, new approaches are needed which take into account the inherently decentralized and dynamic properties of the Internet and distributed systems in general.

This paper presents an overview of the WOS architecture and its components. We start by describing the general approach taken in the WOS project to define system services for global computing. This is briefly compared to some related work. It is however beyond the scope of this paper to present an ex-

haustive in depth analysis of the many approaches to global distributed computing. The next section gives an overview of the architecture of a WOS node, the definition of the WOSNet or WOSspace followed by the presentation of the communication protocols. Resource management functions are then illustrated by load sharing and load balancing mechanisms. The use of multiple versions is discussed in the next section. Finally, the presentation of the state of the work and a brief discussion on the future work conclude the paper.

CONCEPTS OF THE WEB OPERATING SYSTEM (WOS)

The WOS approach to global computing first presented in (Ben Lamine *et al.* 1997) aims to provide solutions for global ubiquitous computing and to develop service mechanisms which meet the requirements of the net-centric view of services and processes. To account for the dynamic nature of the Internet, generalized software configuration techniques, based on demand driven technique called *eduction* (Plaice and Ben Lamine 1997, Paquet 1999). 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 engines. A WOS-node integrates thus client, server, and broker/trader functions. These nodes are capable of providing a set of services, that can pass on to each other requests when appropriate. Because the Internet or the Web is dynamically changing in many directions, any attempt to design one single operating system offering a fixed set of resource management functions is fore-doomed. Therefore, the WOS is designed not only as a *distributed*, but also as a *versioned* system. Different versions of services and of the WOS itself are running simultaneously on the network. Warehouses associated with the WOS nodes provide the necessary information and components for fulfilling service requests. Each node thus uses its warehouses to store and continuously update information about the node and available services and resources. This approach allows for interaction with many different warehouses, each offering different versions of available services, resource management techniques, applications, platforms, hardware, and so on.

While the WOS architecture relies on the decentralized eduction engines with their warehouses, it considers the communication protocols to be the centralized part. The communication protocols may thus be seen as the “glue” of the WOS architecture. Communication between nodes is realized through a simple discovery/location protocol (WOSRP) and a generic service protocol (WOSP). The WOSP protocol is in fact a protocol language with a corresponding parser and serves to easily configure service-specific protocol instances. For example, one WOSP instance could implement an interface to XML, CBL (Common Business Library) or GIOP/IIOP of CORBA. At the lower levels of the protocol stack, we assume the usage of the TCP/IP protocol family.

RELATED WORK

There are several approaches to integrate the computational resources available over the Internet into a global computing resource. The closest approach to the WOS is the Jini architecture proposed by SUN Microsystems (SUN 1999). Jini allows one to build federations of nodes or distributed objects offering different services each relying on its own service protocol. Lookup services provide location and discovery functions. The WOS approach is qualitatively different and more general in that federations, i.e. subsets of WOS nodes, defining a specific environment and context are dynamically and autonomously created. This is achieved with versioning and powerful lookup/discovery protocols and generalized service communication protocols.

Other efforts to exploit distributed resources for wide-area computing include Linda, PVM, MPI, Netsolve (Casanova and Dongarra 1997), Globe (van Steen *et al.* 1997), WebOS (Vahdat *et al.* 1998), Legion (Grimshaw *et al.* 1997) and Globus (Foster and Kesselman 1997). In contrast to the WOS approach, most of these systems require login privileges on the participating machines, or require operating system or compiler modifications. They usually also require architecture specific binaries. The use of Java addresses the latter issue in a number of projects including Atlas (Baldeschwieler *et al.* 1996), ParaWeb (Brecht *et al.* 1996), Charlotte (Baratloo *et al.* 1996), Javelin (Christiansen *et al.* 1997) and Popcorn (Camiel *et al.* 1997). Those projects aim mostly to provide Java oriented programming models for Internet-based parallel computing. Our approach is orthogonal to these proposals in that Java oriented programming models could be integrated into the WOS through gateway interfaces. But the WOS is different in that it does not require any global centralized catalog of resources as it is for example necessary in Javelin, ParaWeb, Atlas or Globus.

THE WOS APPROACH

The WOS framework consists in the sum total of the interactions between nodes relying on the basic communication protocol and multiple warehouses on each node. It is designed as a completely open system to every user. Conceptually, the totality of WOS nodes constitute the *WOSNet* or *WOSspace*. However, since the WOS is a versioned system, subnets can be easily defined: a collection of some WOS nodes may be defined as a particular version of the WOSNet. For example, a number of servers of an Intranet or an Extranet could be defined as a WOSspace, i.e. a version of the WOSNet including only those servers. Every WOS compliant system can potentially contact those servers, but only if they are version-compatible. In this way, a business traveler could read his e-mail while traveling even if the service is hidden behind the company's firewall. He would connect with his WOS compliant wireless PDA to a local ISP, locate the company's WOS version, and connect through this WOSNet to the company's e-mail service.

Architecture of a WOS node

Fig. 1 shows the general architecture of a WOS node and Fig. 2 the corresponding layered structure.

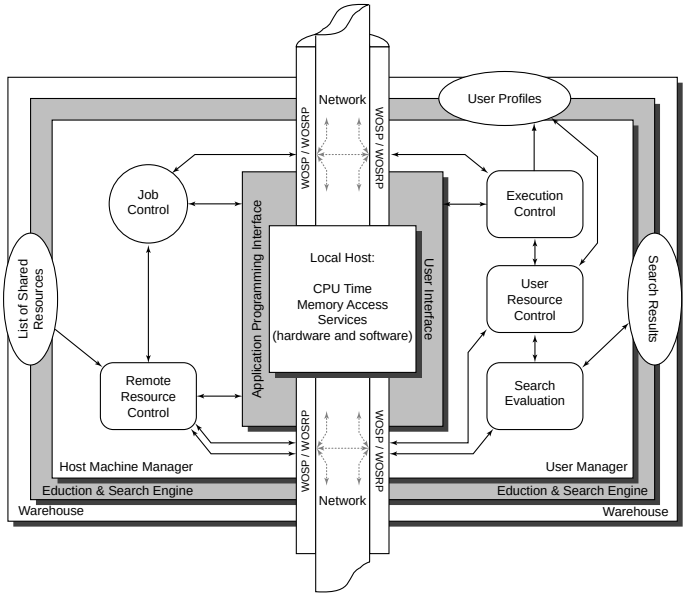

Figure 1: The Architecture of a WOS Node.

Users interact with the WOS through a User Interface. That interface provides a unique gateway to the services available on the WOS. All service requests are made through that interface. The User Interface will display execution status and results, as they come. The Host Machine Manager handles all service requests received by this node. It is responsible for responding to resource search requests and for executing services, once approved. The User Manager is responsible for the coordination of any WOS-service required by/for any given user. It requests and allocates the resources required by a service, based on information stored in the local warehouses.

Communication in the WOSNet

The two layers below the managers shown in Fig. 2 implement WOSP. This protocol is versioned. Here the question arises how we can implement a single infrastructure to handle a multi-versioned protocol ? The answer is quite simple. First, we use WOSRP to lookup the different available WOSP versions, as described later. Second, WOSP versions actually differ only on the semantics they convey. A single, open and complete syntax can therefore be defined to handle the transmission of different universe of discourse. The WOSP Parser handles the conversion to and from that syntax. The different versions of WOSP are implemented by specializing the WOSP Analyzer module. At run-time, a warehouse lookup is made to bind the node to the appropriate instance of the WOSP Analyzer, based on the WOSP version ID.

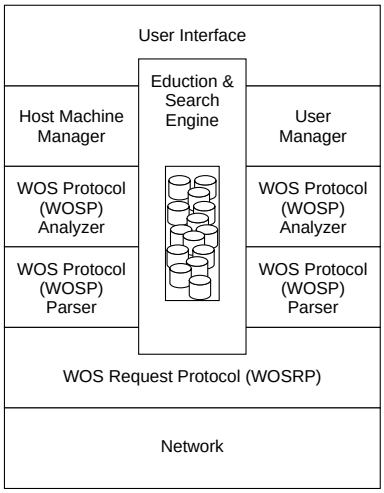

Figure 2: Functional Layers of a WOS Node.

Finally, the WOSRP layer isolates the WOS node from the network. It provides two basic services : locating WOS nodes understanding specific WOSP versions and connecting to other WOS nodes using a specific WOSP version. A detailed description of WOSRP and WOSP, their specification and implementation is presented in (Wulff *et al.* 1999).

The bootstrap problem

When a new WOS node is added to the WOSNet, all it knows is an initial list of WOSP versions it understands[1]. It knows nothing about other nodes in its neighborhood. The first order of business for the node is therefore to locate other WOS nodes in its neighborhood. To do this, the WOSP Version Manager will broadcast to its local network a WOSRP message requesting information about any WOSP version. If no answer is received (i.e., no WOS nodes are located in the local network), the WOSP Versions Manager in the worst case broadcasts to the next network level. This process continues until at least one WOS node is found or until every machine on the Internet is visited (Babin *et al.* 1998). This approach will not flood the Internet, since we proceed by recursive waves, instead of broadcasting everywhere at one time. If at least one WOS node is found, requests for specific WOSP versions may be sent as seen above.

Load sharing and load balancing mechanisms

First results on load sharing mechanisms and resource management approaches: an efficient approach to resource discovery on large networks have been presented in (Unger *et al.* 1998) and (Böhme and Unger 1998). Multiple independent search chains or threads are used to search for resources. This method maximizes the number of nodes visited while minimizing network traffic. The efficiency and correctness of the method was demonstrated with stochastic simulation as well as experimentally on the Internet. In (Unger and Kropf 1998), we discuss requirements and methods for load management in the WOS. We propose the definition of a user space with a set of resources on different machines which should satisfy most of the users' service requests. To define this user space, we suggest the application of statistical methods on a user's resource usage. As mentioned before, the steadily adapted and updated information on user spaces is locally kept in a warehouse and consulted for deciding on service request execution. The placement of an application or service execution within the reserved user space can then be done with standard load sharing methods, e.g. initial or random placement, sender/receiver initiated or others.

Versions

It should be noted that WOSP is only one element that may be versioned in a WOS node. Any resource may in fact be versioned. The use of the eduction engine becomes indispensable to identify and locate the actually requested service based on an identification scheme, which still needs to be formally defined. Version management in the WOS has been informally presented in (Ben Lamine and Plaice 1998). It follows from previous work on software versioning (Plaice and Wadge 1993) and intensional programming (Paquet 1999), in particular the Lucid programming language (Wadge and Ashcroft 1985). An intensional programming system is characterized by two operations, *lookup* and *eval* which define interactions with possibly multiple warehouses and definition catalogs. These catalogs are basically tagged objects denoting (context,value) pairs which represent identifiers. The two operations will, in general, have side effects, i.e. cause explicit changes of state by updating warehouses and catalogs along the way. The generalization and application of these concepts to the WOS allow for dynamically changing contexts and even creating new contexts of computation.

Prototype and application development

Various prototypes focusing on different aspects of the WOS are currently being developed. One prototype focuses on search mechanisms and load distribution/sharing. It uses UNIX sockets for communication, a simple resource description language and HTML forms for service requests. The Altavista search engine serves to locate resources and CGI scripts are used to emulate some basic functions of the eduction engine. A second prototype allows for transparent remote access to resources based on the Java RMI technology and the well known desktop metaphor as a user interface (Krone and Schubiger 1999).

In parallel, development of various components of the WOS including the implementation of the WOSRP/WOSP protocols (Wulff *et al.* 1999), resource and warehouse management, a security system (Unger 1999) and an agent-based distributed file service mechanism (Mikler 1999) is under way. As an explicit design choice, we try to leverage as much as possible from existing services such as TCP/IP (including IPv6). Moreover, the Java programming environment has been chosen as the implementation platform in order to allow for maximal portability, easy maintainability (only one code must be maintained) and easy deployment. Furthermore, this choice may be advantageous in the future to exploit some interesting features of the recently introduced JavaOS (Saulpaugh and Mirho 1999), such as for example the JavaOS System Database (JSD) and its

[1] This initial list still needs to be defined.

name-spaces. The various pieces developed are continuously integrated into prototypes providing more and more functions.

First simple applications for the WOS prototypes include tool services (e.g. data format translators, formatting/typesetting services, distributed document generation). There exists a wide range of possible application areas which could be worthwhile to explore in the long term:

- Creation of virtual data centers which contain data, data manipulation, analysis, modeling and visualization tools for many different domains of interest such as environmental, geographic or marine information technology, retailer groups, banking etc.
- The integration of distributed programming models for specific application domains (e.g.. scientific computing, multi-media services, etc.).
- Application of the WOS to integrate heterogeneous and incompatible applications.
- Development of a WOS based framework and services for open virtual marketplaces or e-commerce in general.

As mentioned earlier, particular WOSP versions will implement interfaces to existing and future application and service protocols. Similarly, WOS nodes may offer gateway functions to other computing models and programming systems. Such a gateway service is currently being developed for MPI (Banicescu and Unger 1999).

CURRENT AND FUTURE WORK

The development of the WOS concepts and software is an on-going collaborative effort from Laval University (Canada), University of Rostock (Germany), University of New South Wales (Australia), University of North Texas (USA). The current projects under way include the development of security models for the WOS, the specification of version identification (eduction), name spaces and warehouse management, the development of agent-based distributed file system mechanisms, the implementation of resource management mechanisms, and the consolidation of already developed concepts and prototypes. Furthermore, a gateway to the MPI programming model is being implemented as well as the application of the WOS concepts to electronic commerce in the case of auctions. In the future, we aim to develop methods and tools to support the development of new WOSP versions, taking into account the service requirements for the new versions, the development of other system gateways and WOS compliant application components.

The long term objective of the WOS project is to define and develop methods and frameworks for current and future interconnected systems and thus to contribute to future global information infrastructures and technologies. In particular, the presented approach relying on multiple simultaneous versions and the eduction engine aims to provide better abstractions of the underlying complexity of networked systems and services while allowing for handling dynamically changing contexts and environments.

Acknowledgements

We would like to thank the many researchers and students for their enthusiastic participation in realizing this ambitious project: J. Plaice, H. Unger, G. Babin, A. Mikler, I. Banicescu, O. Krone, S. Ben Lamine, M. Wulff, H. Coltzau, S. Schubiger, Q. Zhong, S. Yuen, F. Daignault.

References

Babin, G.; P. Kropf; and H. Unger. 1998. "A Two-Level Communication Protocol for a Web Operating System (WOS^{TM})." In *IEEE Euromicro Workshop on Network Computing* (Västerås, Sweden), 939–944.

Baldeschwieler, J.; R. Blumofe; and E. Brewer. 1996. "ATLAS: An Infrastructure for Global Computing." In *Seventh ACM SIGOPS European Workshop on System Support for Worldwide Applications*.

Banicescu, I. and H. Unger. 1999. "Running Scientific Computations in a Web Operating System Environment." In *High Performance Computing Symposium '99* (San Diego, CA, USA), Tentner, A., ed., SCS International.

Baratloo, A.; M. Karaul; Z. Kedem; and P. Wykoff. 1996. "Charlotte: Metacomputing on the Web." In *9th Conference on Parallel and Distributed Systems*.

Ben Lamine, S. and J. Plaice. 1998. "Simultaneous Multiple Versions — the Key to the WOS." In *Distributed Computing on the Web (DCW'98)* (Rostock, Germany), 122–128.

Ben Lamine, S.; P. Kropf; and J. Plaice. 1997. "Problems of Computing on the Web." In *High Performance Computing Symposium 97* (Atlanta, GA), Tentner, A., ed., SCS International, 296–301.

Böhme, T. and H. Unger. 1998. "Search in the WOSNet." In *Distributed Computing on the Web (DCW'98)* (Rostock, Germany), 141–142.

Brecht, T.; H. Sandhu; M. Shan; and J. Talbot. 1996. "Towards World-Wide Supercomputing." In *Seventh ACM SIGOPS European Workshop on System Support for Worldwide Applications*.

Camiel, N.; S. London; N. Nisan; and O. Regev. 1997. "The POPCORN Project: Distributed Computing over the Internet in Java." In *6th International World Wide Web Conference*.

Casanova, H. and J. Dongarra. 1997. "NetSolve: A Network Server for Solving Computational Science Problems." *International Journal of Supercomputer Applications and High Performance Computing* 3, no. 11:212–223.

Christiansen, B. O.; P. Cappello; M. F. Ionescu; M. O. Neary; K. E. Schauser; and D. Wu. 1997. "Javelin: Internet-Based Parallel Computing Using Java." In *ACM Workshop on Java for Science and Engineering Computation*.

Foster, I. and C. Kesselman. 1997. "Globus: A Metacomputing Infrastructure Toolkit." *International Journal of Supercomputer Applications*.

Grimshaw, A.; W. Wulf; J. French; A. Weaver; and P. Reynolds. 1997. "The Legion Vision of a Worldwide Virtual Computer." *CACM* 40, no. 1.

Krone, O. and S. Schubiger. 1999. "WebRes: Towards a Web Operating System." In *11. Fachtagung Kommunikation in Verteilten Systemen (KIVS '99)* (Darmstadt, Germany).

Microsystems, S.. *Jini*. 1999. http://java.sun.com/products/jini/whitepapers/.

Mikler, A.. 1999. "Specification of an Agent Based Distributed File System for the WOS." In *High Performance Computing Symposium '99* (San Diego, CA, USA), Tentner, A., ed., SCS International.

Paquet, J.. 1999. *Intensional Scientific Programming*. Ph.D Thesis, Faculté des études supérieures, Université Laval, Québec, Canada.

Plaice, J. and S. Ben Lamine. 1997. *Eduction: A general model for computing*. In *Intensional Programming II* (Singapore), World Scientific.

Plaice, J. and W. Wadge. 1993. "A new approach to version control." *IEEE Transactions of Software Engineering* 19, no. 3:268–276.

Saulpaugh, T. and C. Mirho. 1999. *Inside the JavaOS^{TM} Operating System*. Addison Wesley Longman, Reading, MA, USA.

Unger, H.. 1999. "A New Security Mechanism for the Use in Large Distributed Systems." In *High Performance Computing Symposium '99* (San Diego, CA, USA), Tentner, A., ed., SCS International.

Unger, H. and P. Kropf. 1998. "Overview About the Resource Management in the Web Operating System (WOS^{TM})." In *Distributed Computing on the Web (DCW'98)* (Rostock, Germany), 134–139.

Unger, H.; P. Kropf; G. Babin; and T. Böhme. 1998. "Simulation of Search and Distribution Methods for Jobs in a Web Operating System (WOS^{TM})." In *High Performance Computing Symposium '99* (Boston, MA, USA), Tentner, A., ed., SCS International.

Vahdat, A.; T. Anderson; M. Dahlin; E. Belani; D. Culler; P. Eastham; and C. Yoshikawa. 1998. "WebOS: Operating System Services for Wide Area Applications." In *Proceedings of the Seventh IEEE Symposium on High Performance Distributed Systems* (Chicago, IL., USA).

van Steen, M.; P. Homburg; and A. S. Tanenbaum. 1997. "The Architectural Design of Globe: A Wide-Area Distributed System." Technical Report IR-422. Vrije Universiteit, Amsterdam.

Wadge, W. W. and E. A. Ashcroft. 1985. *Lucid, the Dataflow Programming Language*. Academic Press, London.

Wulff, M.; G. Babin; P. Kropf; and Q. Zhong. 1999. "Communication in the WOS^{TM}." Research Report DIUL-RR-9902. Université Laval, Sainte-Foy, Québec, Canada.

Biography

Peter Kropf received the M.Sc. and Ph.D. degrees from University of Bern (Switzerland). Since 1994, he has been working at Université Laval (Québec) where he holds a position of an Associate Professor. He has been carrying out research in parallel computing for over ten years, including mapping and load balancing algorithms as well as many application oriented projects. His research interests include Internet computing, distributed computing, networking and simulation.