WOS: an Internet Computing Environment

Peter Kropf* , Herwig Unger† and Gilbert Babin‡

* Dép. d'informatique et de recherche opéprationelle
Université de Montréal
Montréal, Canada H3C 3J7
kropf@iro.umontreal.ca

† Fachbereich Informatik
Universität Rostock
Rostock, Germany
hunger@informatik.uni-rostock.de

Abstract

Given the current development of the Internet, the Web, mobile communications and services, we are clearly heading towards an era of widely integrated ubiquitous services sharing some kind of global operating system. This article describes the idea, the objectives and the current state of the development of the WOS-project. The Web Operating System (WOS™) consists of a series of versioned servers where each one can offer different services, themselves versioned. Each node can act as a server or a client. A common protocol, itself versioned, is used for communication among WOS nodes. Requests for services can be passed on to other servers as appropriate. The WOS is defined by the combined actions of different nodes.

1 Introduction

1.1 The Context

With the emergence of widespread computing and telecommunication networks, an explosion of networked and mobile computing is taking place; in turn there is a permanent growth in areas such as electronic-commerce, multi-media applications, or large-scale high-performance scalable distributed computing. These developments lead to the conclusion that the global computing infrastructure is in a permanent process of *evolution*.

Because of the rapid changes in the underlying infrastructure, it is clear that component-based systems are best suited for large-scale distributed systems, since, as needs change, components can be replaced or adapted more easily than can entire systems. However, components can themselves be programmed to act differently according to the context in which they are immersed; we call this *versioned programming*, and we assume that as a context evolves, the collaboration between components may change and evolve. We call these evolving contexts and collaborating components – along with their interactions – *communities*. Programming models and techniques based on the above principles require an infrastructure that supports versioned dynamic and adaptive resource management and communication between versioned components can provide answers to these challenges. The goal of the proposed Web Operating System (WOS) initiative is precisely to create such an enabling infrastructure for distributed applications. In a technical sense this middleware can be viewed as a Network Operating System for ubiquitous computing that spans the higher layers on top of the enabling communication network infrastructure to provide to the applications and users eased access to the advanced network services.

Electronic commerce, communication and multimedia services, high performance large-scale cooperative distributed computing, and ubiquitous computing are among today's most relevant wide-area distributed systems. Therefore, the programming models, and distributed computing infrastructure investigated should specifically target these applications.

1.2 The Web Operating System

The design Web Operating System (WOS) approach for global computing relies on the novel concept of dynamically defined or versioned *communities* of components (software and hardware). For example, a community of nodes acting as a parallel computer may now be defined by searching the node's information warehouses (or catalogs) for the resources necessary to define the virtual parallel computer. This will thus define a new context of computation. To deal with change, generalized software configuration techniques, based on a demand-driven technique, called *eduction*, are used for the WOS. The kernel of a WOS node is a general eductive engine, a reactive system responding to requests from users or other eductive engines using the warehouses' information to provide the necessary components for fulfilling service requests. This approach allows interaction with many different warehouses, each offering different versions of services, resource-management techniques, applications, platforms, hardware, and so on. Undoubtedly this approach will help to overcome restrictions of other middleware structures such as CORBA, Java/RMI (and Jini), Globus or Legion, which require user configuration and complete resource catalogs and restrict change to and dynamism to a controlled deployment of changes in components' functionalities.

The concept of the WOS calls for a generic communication framework as its central component instead of a central server (or a fixed set of servers) on which clients rely. Therefore a communication layer supporting versioned protocols is needed to support communications within a community considering the negotiation of appropriate protocol selection, communication set-up, QoS and security issues.

1.3 Communities in computing

Networked or distributed computing means that *multiple* components, arising from several sources, will be put together in a single context; furthermore, objects will not necessarily remain in a given context, and may migrate to other contexts. Intuitively, these contexts may be understood as supporting the creation of communities. Different communities may agree to trade, meet, or discuss, depending on the communication protocols that they can agree upon.

For instance, human or software agents trading and brokering in electronic marketplaces may form communities interaction or common interests. Other examples of communities are Internet chat-rooms or dynamic intra- and extranets of large companies. We believe that this community concept exhibits a potentially very rich model for ubiquitous computing. This concept of evolutive communities might be formally defined and analyzed as a generalization of the agent oriented programming (AOP) while applying the intensional programming paradigm. Intensional programs are programs that are immersed in an implicit environment and allow the manipulation of versions of identifiers and functions. The intensional language ISE (Intensional Sequential Evaluator) is a candidate to experimentally include support for communities [39]. The required infrastructure to support versioned resource management and communication between versioned objects will be provided by the WOS.

1.4 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 [9]. 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. These lookup services, however, require the knowledge of all lookup attributes. Moreover, it must be exactly specified what is looked for, which means that only attributes to be exactly matched may be specified. For example, a search for *the nearest printer* cannot be realized. 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. Every service is versioned in the WOS, and a suitable version is selected according to a 'best fit' strategy. This allows the implementation of smart lookup services where attributes need not to be exactly matched.

Other efforts to exploit distributed resources for wide-area computing include Linda, PVM, MPI, Netsolve [5], Globe [11], WebOS [10], Legion [8] and Globus [7]. 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 [1], ParaWeb [3], Charlotte [2], Javelin [6] and Popcorn [4]. 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.

Peter Kropf, Herwig Unger, and Gilbert Babin. "WOS: an Internet Computing Environment." in Workshop on Ubiquitous Computing, International Conference on Parallel Architectures and Compilation Techniques. Philadelphia, PA, USA. October 2000. pp. 14-22.

2 Node Structure

2.1 General Characteristics

The entire WOS is written in Java. This programming language was chosen to achieve a highly portable system. Because the WOS makes heavily use of the communication capabilities of the operating systems, Java was the best choice in view of its rich features for communication and security.

A WOS communication layer [14] was created to optimize the communication speed while saving resources, e. g. bandwidth, at the same time. Each WOSNode operates as a server as well as a client. The WOSNet consists of a series of versioned servers or nodes [15] which can provide a set of services and resources. There are no central catalogs of resources in the WOSNet. Each WOSNode stores information about other nodes locally in its own resource warehouse [30]. That means, no machine has a global information about all other nodes in the WOSNet. The information stored in the warehouses will be updated each time the node finds other, previously unknown nodes. Using such a kind of decentral resource warehouses, the system achieves a high flexibility and avoids some of the bottle necks of systems with a central information management.

The structure of a WOSNode is shown in figure 1. The left side of this figure shows the server, and right side represents the client part of each WOSNode.

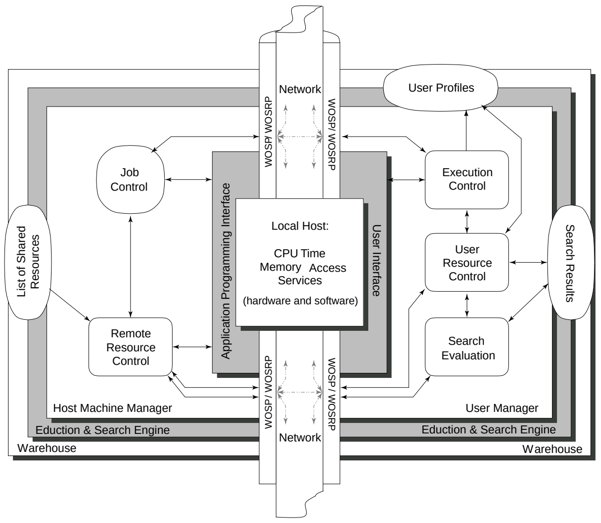


Figure 1: Structure of a WOSNode

For the description of services in the WOS *Profiles* are used. Profiles describing resources with a list of key-value pairs. Each pair defines a special feature of a resource. A printer for instance, has a special type (ink, laser etc.), is able to print black and white or color and can handle Postscript files. Each resource has a corresponding access-object describing its methods, for a printer e. g. self test, economy mode etc. That means that the user does not need to use the commandline anymore. Restrictions which belong to the profiles are described using the same data structure. It is possible to define new forms of restrictions.

With the prototype version 0.4 of the WOS comes a profile editor to create those profiles and to store them in the so called profile warehouse. Some predefined types like integer, domain etc. are available to store the values of the features. Further types can easily be created and dynamically loaded into the system.

Restrictions are store in a separate warehouse to allow more than one access schema for one profile, e. g. for different user groups. Before a user can get access to a profile, he has to fill a form with the restrictions set for this profile. This form is sent to the server together with the request. Then the server can check the restrictions and if the content of the form is correct, the server can perform the requested action.

3 WOS prototypes

3.1 Overview of WOS Releases

In this section an overview of all WOS releases and its main features is given. Table 1 shows also the release dates and gives an outlook of the coming release 1.0 of the WOS.

Table 1: The history of the WOS releases and its main features.

WOS version	**Release Date**	**Features**
0.1	Sep 1998	RCU, RRCU, result propagation via Netscape, Altavista search client, HTML based request forms, implemented in C
0.1.7	Nov 1998	bug fixes, speed optimizations
0.2	Jun 1999	Communication layer vI, implemented in Java, GUI with Java, first services
0.4	Dec 1999	Communication layer vII, search chains, warehouses, profiles, profile editor, services
1.0	Sep 2000	API, bug fixes for WOS kernel, kernel interfaces, Message Chains, security, Job Control, services

In the following, the current WOS release is described.

3.2 Description of the current Version 0.4

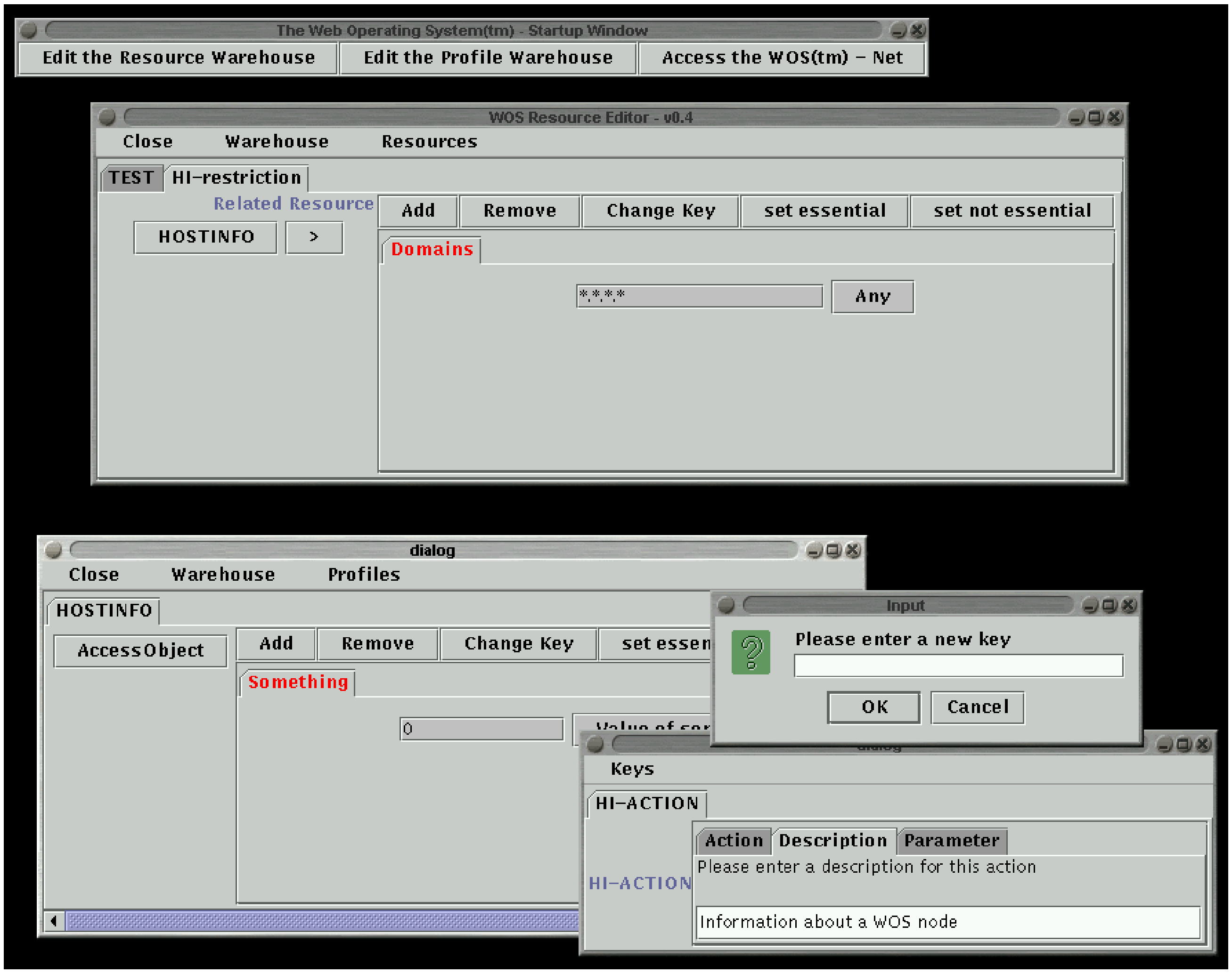
The version 0.4 is the first version with resource search mechanism and a profile based resource administration. It consists of four major components:

- graphical user interface
- profiles, profile editor
- RRCU/RCU for the resource management

- communication layer (WOSRP/WOSP)

The communication layer and its API are described in section 3.3.

The *User Interface* is subdivided in three parts, the profile editor, the resource editor and the request menu. As mentioned before, each resource is described through a profile. The Profile Editor helps the user to generate profiles of resources he wants to make available for other users. He has to define descriptive features of these resources, the object which the remote user needs to access and the parameters for this object. All profiles are stored in the local profile warehouse. In the resource warehouse (see figure 2) the restrictions going for each profile are stored. The user can combine more than one restriction set with one resource. These restrictions will be checked before an other user can access the resources. The third part of the GUI, the request



..*.*

0

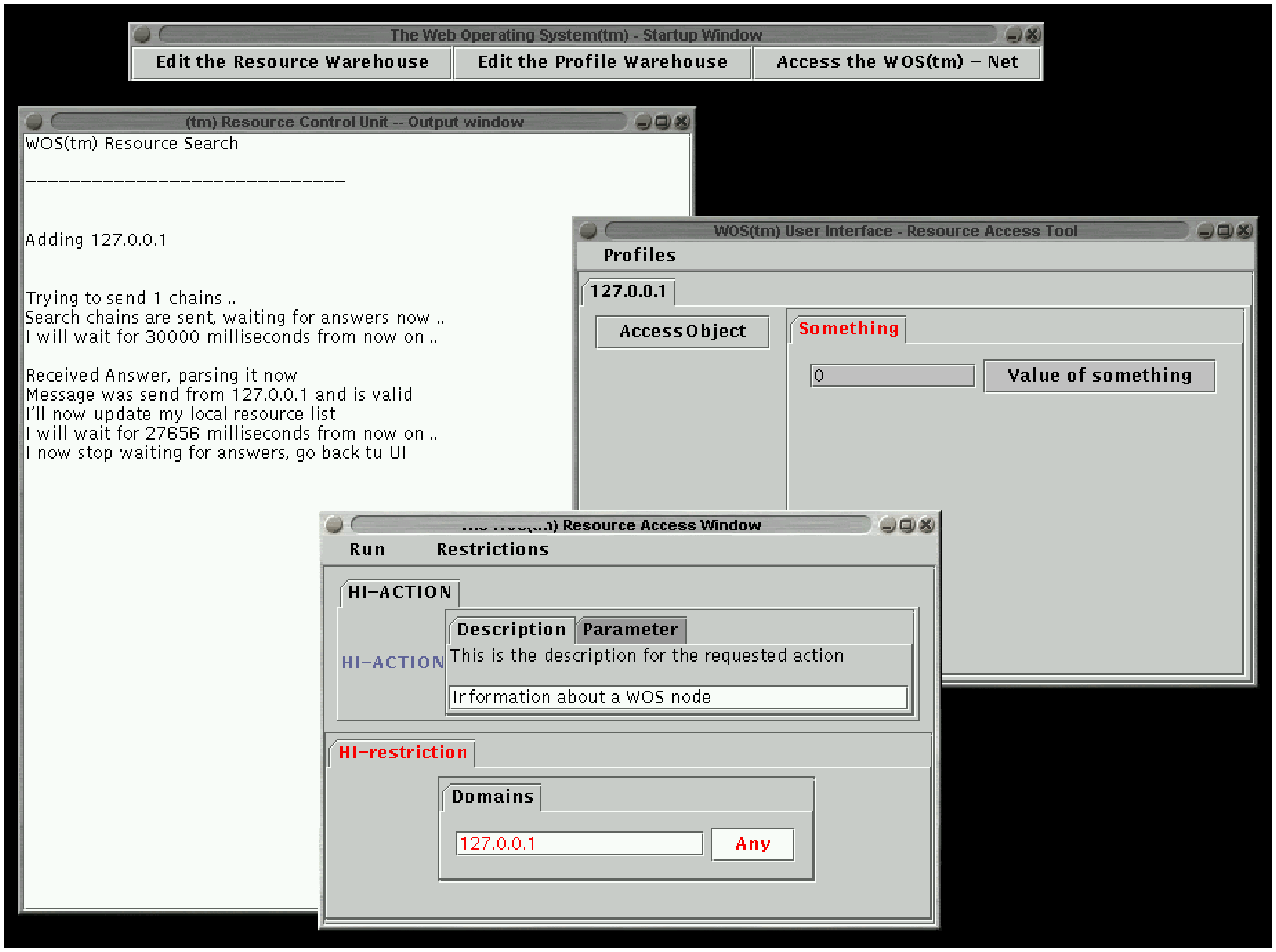
Please enter a new key

Please enter a description for this action

Information about a WOS node

Figure 2: GUI of WOS version 0.4: The warehouse editor, profiles and actions.

menu, provides an easy to use interface to the resources of the WOSNet (figure 3). The user can access all resources stored in the local warehouse and also initiate a search for new resources an update the warehouse.



```
WOS(tm) Resource Search

______________________________


Adding 127.0.0.1


Trying to send 1 chains ..
Search chains are sent, waiting for answers now ..
I will wait for 30000 milliseconds from now on ..

Received Answer, parsing it now
Message was send from 127.0.0.1 and is valid
I'll now update my local resource list
I will wait for 27656 milliseconds from now on ..
I now stop waiting for answers, go back tu UI
```

This is the description for the requested action

Information about a WOS node

Figure 3: GUI of WOS version 0.4: Accessing the WOSNet.

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

The *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 ask. 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 the access rights. After that, the service is executed and the results are passed to the client-side RCU.

3.3 The Communication Layer

For all communication which comes up to realize a service request, Triplets [18] are used in the WOS communication interface. The communication in the WOS provides two possible modes — the connectionless mode and the connection-oriented mode. In order to ensure an efficient search, the connectionless mode is used for the search. That means that there will be no WOSP connection established to sent a request. However, a TCP communication mode is used for a save delivery on the lower communication layers.

The WOS communication layer uses a two-level approach [14]. The first layer (WOSRP) offers discovery/location services and the second one (WOSP) is a generic service protocol defined through a generic grammar. Any instantiation of WOSP is then considered as one version of the protocol. Services which are available in the WOSNet can exist in many different versions. In addition, the WOS protocols itself can be versioned. A two-level protocol is required for the WOS nodes to be able to communicate. On one hand, a protocol must allow the selection of an appropriate version of WOS resources, the WOS Resource Protocol (WOSRP), and on the other hand another protocol, the WOS Protocol (WOSP), is needed to locate and use distributed resources over the Web. A single protocol is not sufficient. A suitable version of the WOS server must be identified before any resources may be accessed. Therefore, each client manages a warehouse. This warehouse also contains information on available other WOS servers and their versions. Once a suitable version and server are identified, a richer language is needed to request services. Therefore, in the WOS are two protocols defined: WOSRP to identify suitable WOS servers, and WOSP to submit service requests.

WOSRP is an application-level protocol which is assumed to be used over IP networks. The rationale behind WOSRP is to provide mechanisms for WOS nodes to exchange information about WOSP versions they support. It is also used to obtain information about other WOS nodes that understand specific WOSP versions. WOSRP has to be simple and flexible. Any version of WOSRP should be fully downwards compatible. Each machine wishing to join the WOSNet may do so without worrying about:

- which version of WOSRP it understands,
- having to gain prior knowledge of other WOS servers;
- any administrative overhead.

A WOS node may "speak" a certain version of the WOS protocol, which means that it can interact with other nodes using that version. A WOS node may also "know" a version of the WOS protocol. That means, that even if cannot interact using that version, it can refer a WOS node to other WOS nodes which might have this capability.

WOSRP also serves to establish connections between WOS nodes. A WOSP message may be encapsulated in a WOSRP message. This way, a generic server may receive all the requests and select the appropriate version to process them.

The WOSP is used if a service should be executed, to pass the results and for the search in the WOSNet. It allows three types of commands:

1. Setup commands are used to change the execution parameters of a WOS node.
2. Execution commands allow a WOS client to use resources from another node.
3. Query commands are used by a WOS client to interrogate another WOS node's warehouse.

The query command is used for the search request messages. A query command can have data and meta-data attached to it. This is sufficient for the search algorithms and the use of WOSRP in the lower protocol level avoids version conflicts during a WOS search. An example of the syntax of a WOSP version is given in figure 9 and the API of the communication layer is described in detail in [31].

4 WOS services and versions

As described in section 3.2 the system services of the WOS allow to setup services and to make resources available to a WOSNet through the profile editor. A number of additional services and functions have been developed which are briefly described in the following sections.

4.1 WOSForward service

The location of services in a WOSNet is based on the mechanism of multiple sequential chains [41]. Based on the knowledge of the WOSNet, a node searching for a service is building lists of nodes to be visited. These lists define sequences of nodes to visit along disjunct paths. The search may thus be performed in parallel along these paths. Theoretical as well as experimental investigations have shown the efficiency of this mechanism [19]. The WOSForward service exploits the principle of multiple search chains to transfer data from a source node to a target node in the WOSNet in parallel along disjunct paths as shown in figure 4.

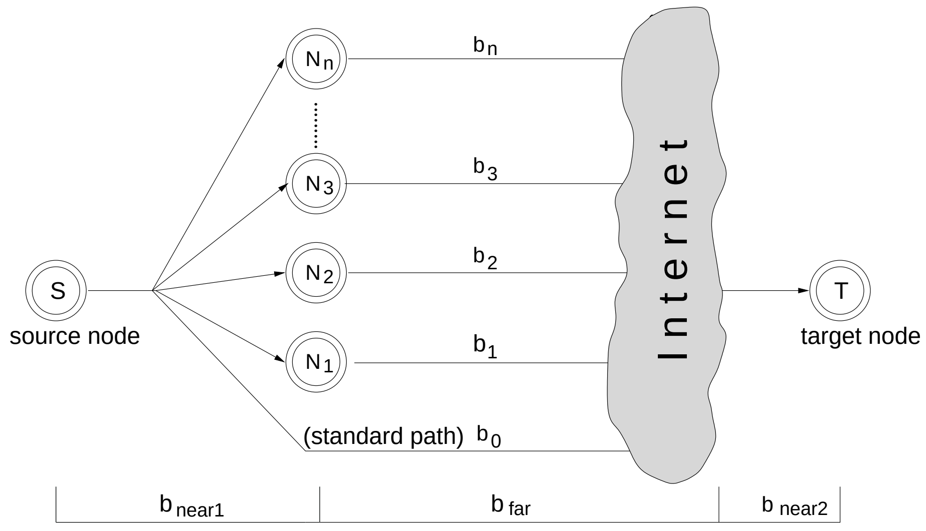


Figure 4: Communication along disjunct paths.

The speedup achieved with this service depends on the available bandwidth in the local area (b_{near}), the bandwidth b_i available on the Internet, and the amount of data to be transfered. In general, we obtain a better speedup, if the bandwidth in the local area is large and if the combined bandwidth of all the disjunct paths in the Internet is large as compared to one single transfer channel. The data D to be transfer is divided into chunks of data d_i which are transfer along the different paths. It is clear that those chunks may not become too small, otherwise the introduced overhead of this method will neutralize the speedup. Figures 5 and 6 show the results obtained with a data transfer between two machines in Europe and North America respectively using two distinct paths.

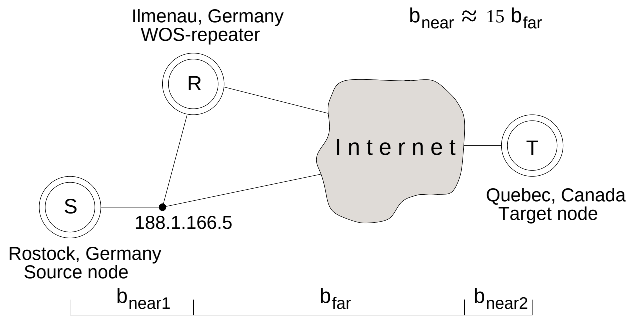


Figure 5: Experimental setup for the communication with two disjunct paths.

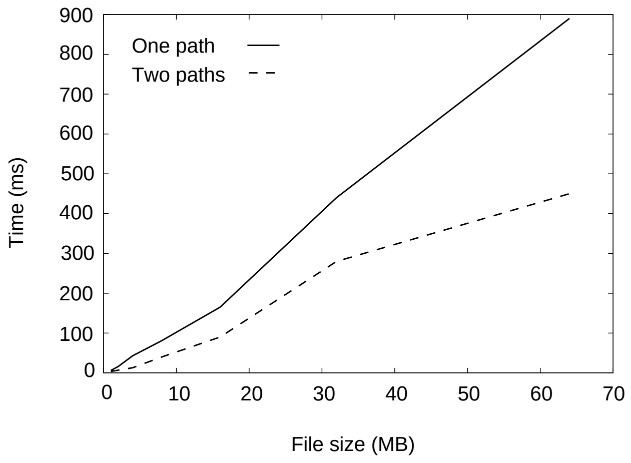


Figure 6: Transmission times for one path and two disjunct paths.

4.2 CORBA – WOS integration

WOS and CORBA both achieve interoperability through a well defined protocol. CORBA makes use of the Internet Inter-ORB protocol (IIOP) to exchange General Inter-ORB protocol (GIOP) messages over a TCP/IP network. The GIOP uses then the Common Data Representation (CDR) to map IDL types onto a raw, networked message representation. The WOS in turn uses the WOSRP/WOSP protocol and the principle of warehouses to store information about services. The profile warehouse allows to store service profiles consisting of a name, and an access object representing the executable to invoke the service. The access object is identified by a key and uses a set of input and output parameters for the execution. To allow the two systems simultaneously, a (protocol)-bridge has been developed and implemented in Java [33]. The triplet [CORBA module, interface name, parameters] is mapped onto [WOS profile name, access object key, access object parameters] allowing to access the service from within both systems.

A generic WOSAdapter as shown in figure 7 provides a CORBA client with the usual view of an ordinary CORBA service, when accessing a WOS based service.

A specific version of the RRCU (Remote Resource Control Unit) of the WOS allows a WOSNode to directly access a CORBA service through the CORBA API. The invocation of a

CORBA service (instead of a WOS resource) is completely transparent for the client as it is the case for the generic WOSAdapter. Figure 8 describes how a CORBA service is invoked from within the WOS.

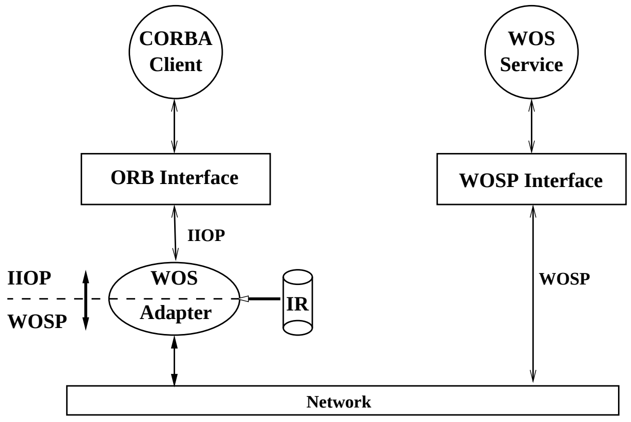


Figure 7: WOSAdapter in CORBA - generic server.

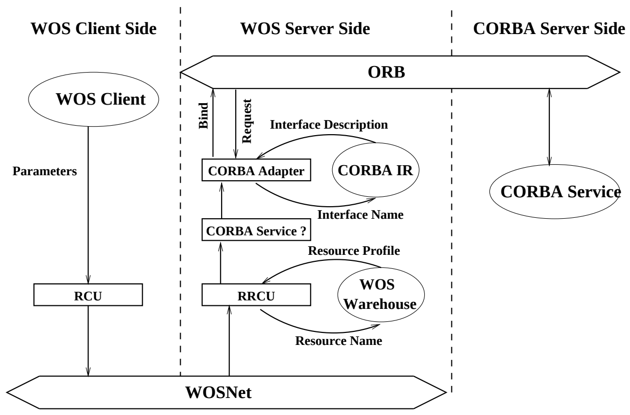


Figure 8: CORBA service invocation from within the WOS.

4.3 HPC – WOS

Tools for wide area high performance computing usually require all the computing resources to be known in advance. Often, this information is even directly compiled into the parallel applications. The computing resources must therefore be exactly configured to match the applications' requirements. This configuration task may involve tedious setup procedures or scripts requiring login privileges, exact knowledge of the resource locations etc.

A version of WOSP, called HPC-WOSP, has been defined to ease this task [38]. It allows to automatically configure and execute HPC applications in the WOS environment, i.e. on resources of a WOSNet. Specifically, it supports the communication requirements for HPC applications, which are:

Configuration stage: the location of suitable WOS nodes with the appropriate set of resources (hardware and software) for an application and the reservation of those resources.

Execution stage: the code distribution and launch of the application.

The figure 9 shows the syntax of the HPC-WOSP protocol version. The information about the properties of the resources and the requirements of the application are again specified with the help of profiles and are kept in the corresponding warehouses. The HPC–WOS implementation has been successfully tested for large MPI and PVM applications.

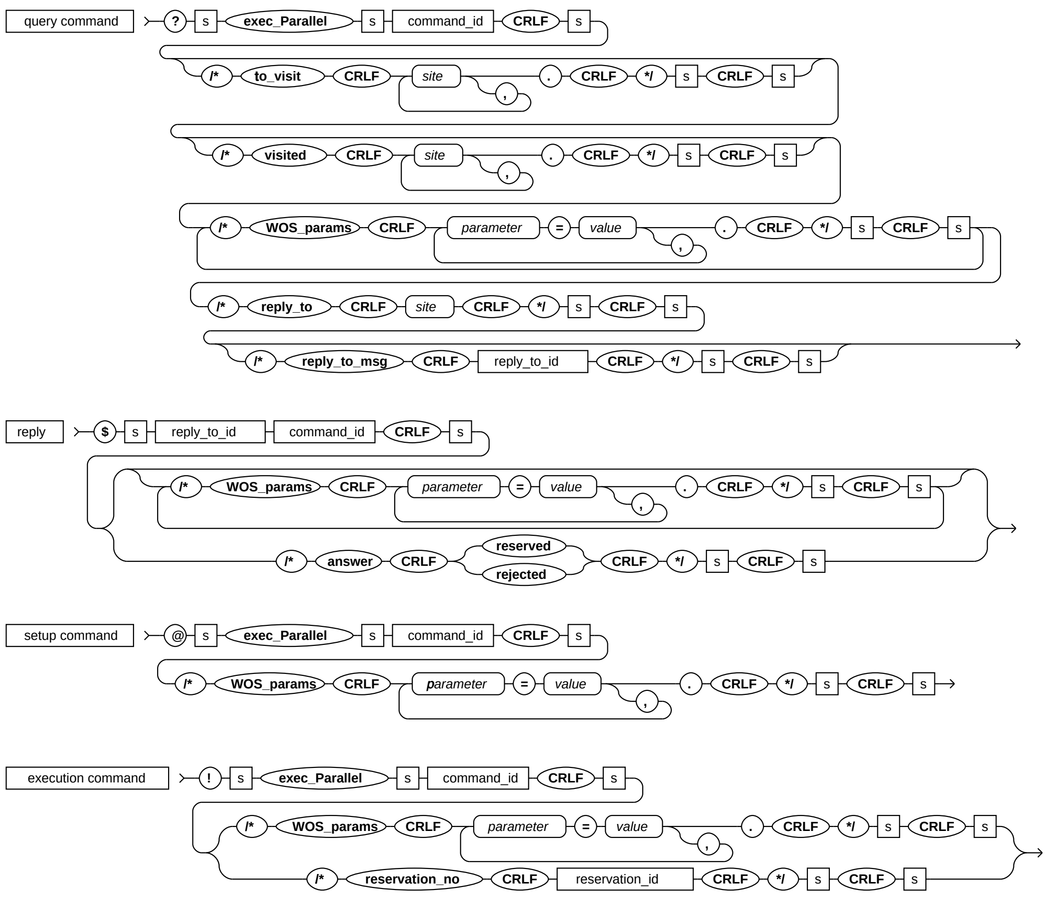


Figure 9: Syntax of the HPC-WOS version

5 Conclusion and Future Work

The Web Operating System (WOS) is at its current state of implementation useful to demonstrate the possibilities of this approach and to test its major functionality. The lack of security mechanisms and clearly defined programming interfaces foils an application of the WOS in a production environment. However, a security system based on automatic trust evaluation has been designed [26, 36] and is currently being implemented. The WOS version 1.0 is scheduled to be released in the last quarter of the year 2000. It will include the yet missing components such as the security module, a complete API and a refined job control module. The experience gained so far with the WOS system and services clearly indicate its potential for future ubiquitous computing, because any device can be a WOSNode and any service can be implemented. The concept of versions applied throughout the entire WOS system allows for the necessary flexibility required by ubiquitous computing. The section *WOS References* gives an overview of WOS related publications.

References

[1] J. Baldeschwieler, R. Blumofe, and E. Brewe. ATLAS: An Infrastructure for Global Computing. In *7th ACM SIGOPS European Workshop on System Support for Worldwide Applications*, 1996.

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

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

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

[5] H. Casanova and J. Dongarra. NetSolve: A Network Server for Solving Computational Science Problems. *International Journal of Supercomputer Applications and High Performance Computing*, 3(11):212–223, 1997.

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

[7] I. Foster and C. Kesselman. Globus: A Metacomputing Infrastructure Toolkit. *Supercomputer Applications*, 2(11):115–128, 1997.

[8] A. S. Grimshaw, W. A. Wulf, J. C. French, A. C. Weaver, and P. F. Reynolds. A Synopsis of the Legion Project. Technical Report CS-94-20, University of Virginia, 1994.

[9] Sun Microsystems Inc. Jini Specification. www.javasoft.com/products/jini/specs, 1999.

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

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

WOS References

[12] S. B. Lamine, J. Plaice, and P. Kropf. Problems of Computing on the Web. In SCS A. Tentner, editor, *High Performance Computing Symposium*, pages 296–301, Atlanta, GA, 1997.

[13] P. Kropf, J. Plaice, and H. Unger. Towards a Web Operating System. In *WebNet '97*, Toronto, 1997.

[14] G. Babin, P. Kropf, and H. Unger. A Two-Level Communication Protocol for a Web Operating System (WOS). In *IEEE 24th Euromicro Workshop on Network Computing*, pages 934–944, Sweden, 1998.

[15] S. B. Lamine and J. Plaice. Simultaneous Multiple Versions: The Key to the WOS. In *Distributed Computing on the Web (DCW'98)*, pages 122–128, Rostock, Germany, 1998.

[16] G. Babin. Requirements for the Implementation of the WOS. In *Distributed Computing on the Web (DCW'98)*, pages 129–133, Rostock, Germany, 1998.

[17] H. Unger and P. Kropf. An Approach for the Resource Scheduling in the WOS . In *Distributed Computing on the Web (DCW'98)*, pages 134–140, Rostock, Germany, 1998.

[18] M. Wulff, G. Babin, P. Kropf, and Q. Zhong. Communication in the WOS . Technical report, PARADIS Laboratory, Université Laval Canada, 1998.

[19] H. Unger, P. Kropf, G. Babin, and T. Böhme. Simulation of Search and Distribution Methods for Jobs in a Web Operating System (WOS). In SCS A. Tentner, editor, *High Performance Computing 1998 ASTC*, pages 253–259, Boston, MA, 1998.

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

[21] P. Kropf. Overview of the WOS Project. In SCS A. Tentner, editor, *High Performance Computing 1999 ASTC*, San Diego, CA, 1999.

[22] J. Plaice and P. Kropf. WOS Communities – Interactions and Relations Between Entities in Distributed Systems. In *Distributed Computing on the Web (DCW'99)*, pages 163–167, Rostock, Germany, 1999.

[23] H. Coltzau, H. Unger, and D. Berg. Implementation of a WOS-Prototype. In *Distributed Computing on the Web (DCW'99)*, Rostock, Germany, 1999.

[24] I. Banicescu and H. Unger. Running Scientific Computations in a Web Operating System Environment. In SCS A. Tentner, editor, *High Performance Computing 1999 (ASTC)*, San Diego, CA, 1999.

[25] M. Wulff. Implementation of the Service Search in the WOSNet. In *Distributed Computing on the Web (DCW'99)*, Rostock, Germany, 1999.

[26] H. Unger. A New Security Mechanism for the Use in Large Distributed Systems. In SCS A. Tentner, editor, *High Performance Computing 1999 (ASTC)*, San Diego, CA, 1999.

[27] S. A. Hopper, A. R. Mikler, P. Tarau, F. Chen, and H. Unger. Mobile Agent Based File System for the WOS: An Overview. In SCS A. Tentner, editor, *High Performance Computing 1999 (ASTC)*, San Diego, CA, 1999.

[28] S. Schubiger, O. Krone, and B. Hirsbrunner. WebComs: Transactions as Object-Flow Networks for the WOS. In *Distributed Computing on the Web (DCW'99)*, pages 31–38, Rostock, Germany, 1999.

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

[30] H. Unger. The Adaptive Warehouse Concept for the Resource Management in the WOS. In *Distributed Computing on the Web (DCW'99)*, Rostock, Germany, 1999.

[31] G. Babin, H. Coltzau, M. Wulff, and S. Ruel. Application Programming Interface for WOSP/WOSRP. In P. Kropf et al., editor, *Distributed Communities on the Web 2000*, LNCS 1830, pages 110–121. Springer, 2000.

[32] S.A. Hopper, A. Mikler, and J. Mayes. Design and Implementation of a Distributed Agent Delivery System. In P. Kropf et al., editor, *Distributed Communities on the Web 2000*, LNCS 1830, pages 192–201. Springer, 2000.

[33] O. Krone and A. Josef. Using Corba in the Web Operating System. In P. Kropf et al., editor, *Distributed Communities on the Web 2000*, LNCS 1830, pages 133–141. Springer, 2000.

[34] O. Krone and A. Josef. Integrating CORBA into the Web Operating Systeem: First Experiences. In SCS A. Tentner, editor, *High Performance Computing 2000 (ASTC)*, pages 207–212, 2000.

[35] M. Wulff and H. Unger. Message Chains as a New Form of Active Communication in the WOSNet. In SCS A. Tentner, editor, *High Performance Computing 2000 (ASTC)*, pages 219–224, 2000.

[36] H. Unger. *Resource Managment in Large Distributed Systems.* Habilitation thesis, University of Rostock, Germany, 2000. In German: *Untersuchungen zum Ressourcenmanagement in grossen verteilten Systemen.*

[37] P. Kuonen, G. Babin, N. Abdennadher, and P-J. Cagnard. Intensional High Performance Computing. In P. Kropf et al., editor, *Distributed Communities on the Web 2000*, LNCS 1830, pages 161–170. Springer, 2000.

[38] N. Abdennadher, G. Babin, P. Kropf, and P. Kuonen. A Dynamically Configurable Environment for High Performance Computing. In SCS A. Tentner, editor, *High Performance Computing 2000 (ASTC)*, pages 236–241, 2000.

[39] J. Plaice, P. Svoboda, and A. Alammar. Building Intensional Communities Using Shared Contexts. In P. Kropf et al., editor, *Distributed Communities on the Web 2000*, LNCS 1830, pages 55–64. Springer, 2000.

[40] J. Plaice and P. Kropf. Intensional Communities. In *Intensional Programming II*, Singapore, 2000. World Scientific Press.

[41] M. Wulff, P. Kropf, and H. Unger. Message Chains and Disjunct Path for Increasing Communication Performance in Large Networks. In P. Kropf et al., editor, *Distributed Communities on the Web 2000*, LNCS 1830, pages 123–132. Springer, 2000.

[42] N. Abdennadher, G. Babin, and P. Kuonen. Combining Metacomputing and High Performance Computing. In *2000 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'2000)*, Las Vegas, Nevada, 2000.

[43] S. Schubiger. A Resource Classification System for the WOS. In P. Kropf et al., editor, *Distributed Communities on the Web 2000*, LNCS 1830, pages 74–81. Springer, 2000.

[44] H. Unger. Distributed Resource Location Management in the Web Operating System. In SCS A. Tentner, editor, *High Performance Computing 2000 (ASTC)*, pages 213–218, 2000.