Nationaal Lucht- en Ruimtevaartlaboratorium
National Aerospace Laboratory NLR

NLR-TP-2001-181

Utilizing supercomputer power from your desktop

B.C. Schultheiss and E.H. Baalbergen

Nationaal Lucht- en Ruimtevaartlaboratorium
National Aerospace Laboratory NLR

NLR-TP-2001-181

Utilizing supercomputer power from your desktop

B.C. Schultheiss and E.H. Baalbergen

The contents of this report have been initially prepared for publication as paper in the HPCN 2001 proceedings in the Springer Verlag series Lecture Notes in Computer Science.

The work reported has been executed as part of NLR's basic research programme, Workplan number I.2.A.2.

Division:	Information and Communication Technology
Issued:	7 May 2001
Classification of title:	Unclassified

Summary

Potentially unlimited computing power is provided today by computing facilities available via the Internet. Unfortunately, the facilities exhibit a variety of non-uniform interfaces to the users. Each facility has its own operating system, its own policy with respect to access and accounting, its own set of tools, and its own data. To utilize the computing power, a user has to select an appropriate facility, and has to know how to access and operate the facility, which tools to use, what options to provide for these tools, how to execute tools, how to submit a job for execution, etc..

The *Superbroker* infrastructure gives easy access to HPCN facilities in the Netherlands through a web browser, thereby taking care of the networking and system details. The user is provided with a single working environment, which gives easy access to the available, usually remote resources as if they were present on a single computer (a ”metacomputer”). Using the desktop’s native software only, the user can easily browse the working environment, start tools, and submit jobs using point-and-click and drag-and-drop operations. The working environment may be tailored for particular end users and application areas. This paper presents the design and implementation of the infrastructure.

Contents

(14 pages in total)

1 Introduction

Potentially unlimited computing power is provided today by computing facilities available via the Internet. Unfortunately, the facilities exhibit a variety of non-uniform interfaces to the users. Each facility has its own operating system, its own policy with respect to access and accounting, its own set of tools, and its own data. To utilize the computing power, a user has to select an appropriate facility, and has to know how to access and operate the facility, which tools to use, what options to provide for these tools, how to execute tools, how to submit a job for execution, etc..

Ideally, the remote computing resources should be accessible without any technical barrier: without the need to install extra software and accessible through a simple graphical user interface (GUI) with a look and feel the user is familiar with. Put in technical terms, we want the high performance computers to operate as *applications servers* that can be accessed by an end-user through a *thin client.*

First, this paper presents the requirements, design, and implementation of a general infrastructure for web-access to remote computing facilities. The key elements of the implementation are the GUI consisting of Java applets, an HTTP daemon with Java servlets, and SPINEware ([7]) object servers with CORBA interfaces. The implementation raises issues such as the communication between the Java applets and the HTTP daemon servlets, the communication between the HTTP daemon servlets and the SPINEware object servers, and launching SPINEware servers when necessary. These topics are covered in the present paper.

Next, a case study demonstrates how this general infrastructure is applied to provide the *Superbroker* working environment, for accessing the public Dutch HPCN facilities.

2 Requirements

Figure 1 presents an overview of the required functionality for the Superbroker working environment: a user easily and uniformly accesses tools and data on the HPCN facilities as if these were available from a single computer. The decisive requirements for the web-based access to the HPCN facilities are:

- Provide access to the HPCN facilities through a single network address.
- Provide access to the HPCN facilities without the need for installation of extra software on the local desktop. Also, the software running on the local desktop shall be portable. We have translated these requirements into: the only required software is a *Java 2 capable* web

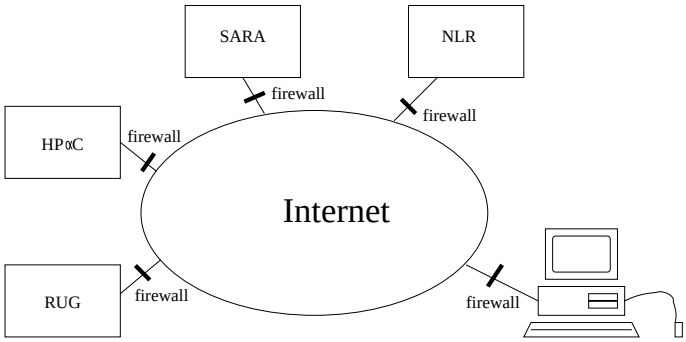


Fig. 1 Usage of SPINEware to access remote HPCN facilities

browser.

- Provide the user with a simple, predefined working environment which allows the user to browse through and access the available tools, data, and information.
- Provide support for the integration of tools in a working environment. The tools vary from compilers to Computational Fluid Dynamic applications. Tools may require a set of input parameters and files.
- Offer support for transportation of code and data files, including the required build (i.e., *make*) instructions, between the end user and the HPCN facility, if possible in compressed or encrypted form.
- Provide support for starting up (remote) jobs on the HPCN facilities.
- At job start-up, the job parameters can be specified, allowing the user to, e.g., limit the duration of a job.
- The user interface shall give a fast response (say within 0.2 seconds). Note that the action as specified using the web interface may take some time, depending on network delay.
- An authentication mechanism using user name and password shall be used. The user must be able to change the password. We translated this requirement into: the system must use the user name password mechanism of the HPCN facilities that are accessed. Data communication shall be secure.
- In order to enable users protected by a firewall to access a remote HPCN server, the communication between the user's user interface and the remote HPCN server shall be based on a widely available protocol such as HTTP or HTTPS.

3 Design

This chapter describes how services on remote hosts can be accessed using a locally running web browser.

The design must be such that the system can be used as follows. The user starts a native web browser and specifies the name (URL) of the login html page. The user selects the remote computing facility on which he or she wants to launch the initial SPINEware object server (see section 4.1 for details), and specifies the user name and password for the selected facility. A SPINEware object server will be launched. The user's web browser now provides a single working environment, which gives access to the available resources as if they were stored on a single computer (a "metacomputer") [7]. The user can browse the working environment, start tools, and submit jobs using point-and-click and drag-and-drop operations. In addition to browsing through information and launching tools on the specified host, the user may open browsers for accessing resources on other hosts as well. Data can simply be moved and copied by dragging and dropping icons from one browser window to another. The working environment may be tailored for particular end users and application areas.

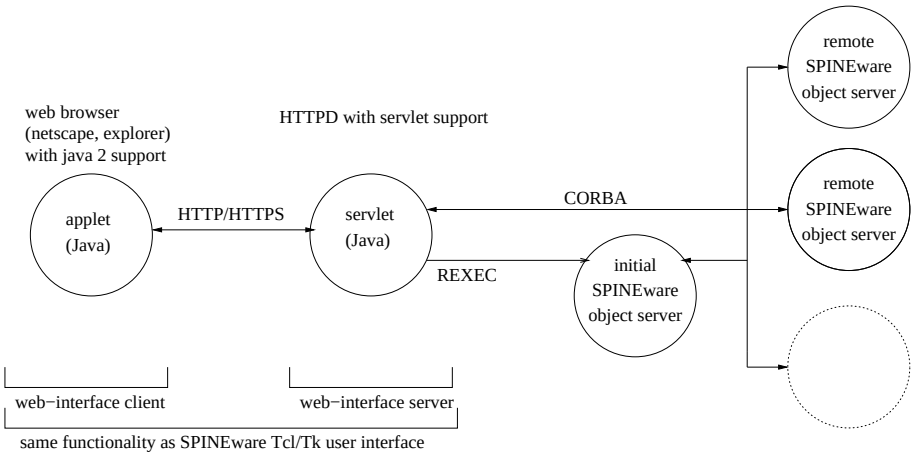


Fig. 2 SPINEware architecture

The architecture of the system is shown in figure 2. The figure displays a 3-tier model:

- The Java applets that implement the GUI.
- SPINEware object servers that carry out the requested actions and that manage the resources involved.
- A web-interface server that enables the user to obtain the applets, and that functions as a middle-tier in the communication between the applets and the SPINEware object servers.

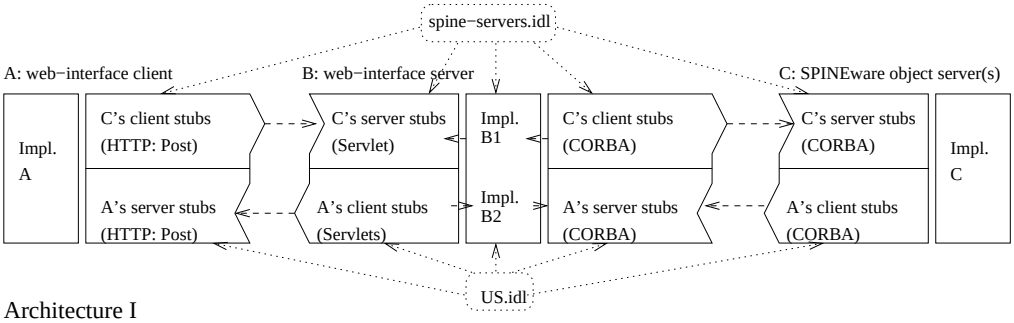
The technical details are discussed in chapter 4. Figure 3 shows two possible architectures for the communication layer between the SPINEware object servers and the local GUI applets.

- Architecture I is the 3-tier architecture as presented and implemented. This architecture is required when using Java applets, because applets can only reconnect to the HTTP daemon from which they were obtained. An advantage of this architecture is that we can use our

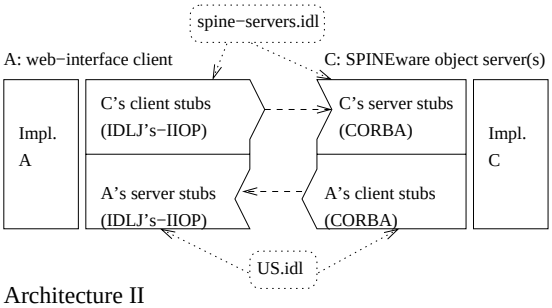
own communication layer between the Java applets and the web interface server. At this moment, we simply use the HTTP protocol or the Secure HTTP (HTTPS) procotol. HTTPS is available by default from the Java libraries. However, the HTTP daemon has to support it. Within the *Superbroker* project (see chapter 5), an Apache HTTP Daemon ([6]) with HTTPS support is used. Another advantage of the 3-tier architecture is that it is possible to monitor which users are logged on at any moment and what the users are doing, because all communication is done via the same web-interface server. An administration applet enables the administrator to do the monitoring. A disadvantage of a 3-tier system is the communication overhead.

- Architecture II shows a web-interface client which directly communicates with the SPINEware object servers. This is only possible when running the web-interface client as a stand-alone program instead of Java applets due to security restrictions imposed on Java applets. Note that architecture II does not fulfil the ”only a Java 2 capable browser is required” requirement.

As an obvious design principle, we have clearly separated the communication layer from the implementation of the GUI.



Architecture I



Architecture II

Fig. 3 Architectures I and II

4 Implementation

This chapter presents the technical details of the 3-tier model: SPINEware object servers, the web-interface server, and the Java applets.

4.1 SPINEware object servers

SPINEware exhibits an object-based model. In a working environment, all resources available from the interconnected hosts are modeled as objects. SPINEware provides ”basic” object classes such as File, Directory (for a native system’s directory or folder), ObjectFolder (a folder for organizing objects), Trashcan (”recycle bin”), Printer, AtomicTool (wrapped, ready-for-execute programs), Job (for managing the execution of AtomicTools), Workflow (for chaining tools, thereby passing output files from one tool as input files for another), and the GUI object named UserShell. Starting from the basic object classes, new classes may be added - whether or not inheriting from existing classes - and existing classes may be modified. The access operations on the objects are defined in terms of methods. Examples are *Edit* for File, *View* for Directory and ObjectFolder, and *Execute* for AtomicTool.

Each object in SPINEware has a (world-wide) unique object identifier, which contains the class (”type”), the Internet name of the host it resides on, and a host-unique identifier. The URL naming scheme is used to identify a SPINEware object. An object identifier is usually written as a so-called ”SPIRL”, a *SPI*NE *R*esource *L*ocator. For example, directory */home/user/src* on host *desk12.nlr.nl* can be identified using the SPIRL

```
spine://Directory@desk12.nlr.nl//home/user/src
```

SPIRLs are also used for specifying method invocations, such as

```
spine://Directory:View@desk12.nlr.nl//home/user/src
```

which instructs SPINEware to display the contents of the specified directory on the user’s desktop. SPINEware provides a *SPIRL broker* utility, which enables method invocations from within tools, scripts, and command-line interpreters.

The objects available from a host via a SPINEware working environment are managed by and accessible via a *SPINEware (object) server* running on the host on behalf of the user and the working environment. When a session with a working environment is started, one SPINEware object server is started initially. This server will handle all requests from the user interface. A minimum (i.e., single-host) session with a working environment involves at least a user interface and the initial SPINEware object server for accessing the host’s resources as objects. If an object on another host

is accessed, the initial SPINEware object server starts (”on demand”) a SPINEware object server on the other host, and relays all subsequent method invocations (and the corresponding results) involving objects on that host.

The communication among the SPINEware objects is based on CORBA, and is implemented using an off-the-shelf CORBA implementation, *ILU* [4]. This product supports CORBA-based inter-object communication over local networks as well as the Internet, and is capable of being used in combination with firewalls.

4.2 Web-interface server

The web-interface server is the middle tier of the system, see figure 2. This middle tier is implemented using an HTTP daemon (*httpd*) with Java servlets.

The servlets are invoked by applets (see chapter 4.3) when the GUI wants to invoke an object method via the SPINEware object server. For example, if the user wants to see the contents of a job queue, the *getContents* method of the Job object is invoked.

When the user logs on to the system, the *Login servlet* is invoked. The Login servlet takes care of publishing a new instance of the SPINEware UserShell object to the CORBA naming service for the user who logs on, and launches a SPINEware object server on the specified host using *rexec* and the user name and password provided by the end user. For example, if the SPINEware Tool Editor (started upon invocation of the *Edit* method of the AtomicTool object) needs to display an error message, it invokes the *ErrorMessage* method of the UserShell object. This results in invocation of the ErrorMessage implementation in the web-interface server via its CORBA interface. The web-interface server ensures that the correct applet code is run. This applet method invocation is, due to security restrictions on applets, implemented by a blocking poll for a command by the applet.

The CORBA interface of the webinterface server has been built using Java’s *idlj*. So, the protocol used between the SPINEware object server and the webinterface server is CORBA’s *IIOP*.

4.3 Web-interface Java applets

The web-interface client (i.e. the GUI) is implemented by Java 2 applets. All communication with the web-interface server is implemented using HTTP or HTTPS *Post* commands. Consequently, if a firewall is between the end user’s system and the web-interface server, this poses no problem as long as the firewall allows HTTP (or HTTPS). Also, the applets always initiate the communication. As explained in the previous chapter, the Java applets poll the web-interface server for commands

to be executed.

The webserver will send an `SC_UNAUTHORIZED` message if the user name and password are not specified in the servlet request, and thus obtains the username and password. In response, the web browser will display a login window, prompting the user to specify the user name and password of the system he or she wants to access initially.

Any customization information for the GUI (e.g., which browsers to open, colors, character sets, etc.) will be stored at the accessed remote host, and will be obtained from the SPINEware object server at login.

To access local files from the GUI, Java Web Start [8] is supported. The GUI could also be run as a Java standalone program instead of using the web browser, thus enabling access to the local file system.

5 A case study: Superbroker

The managing institutes of the major HPCN facilities in the Netherlands (NLR, SARA, HPC-RuG, and HPaC/TUD) have developed a uniform infrastructure named *Superbroker*, that provides users from small and medium-sized enterprises (SMEs), government, and industry, with easy access to the HPCN facilities in the Netherlands and support for using those facilities. The activities for realizing this infrastructure have been carried out in the scope of the HPCN 2000 Infrastructure project, which was funded by the Dutch HPCN Foundation.

The Superbroker web site (`http://www.superbroker.nl`) provides the user with access to information on the HPCN facilities, enables the user to obtain accounts on the HPCN facilities, and offers working environments to facilitate access to data and tools on the HPCN facilities. Figure 4 shows the Superbroker architecture: it is a specialization of the general SPINEware architecture as shown in figure 2.

To interactively access an HPCN facility with a predefined working environment, start a web browser on your local workstation or PC, and provide the *Superbroker* URL. After successful registration and login, a file browser for the top-level ObjectFolder of the working environment pops up, as shown in figure 5. From this very moment on, you may browse through the working environment by opening Directories and ObjectFolders via double clicking on icons. The tree-shaped organization of the working environment together with the on-line help information will

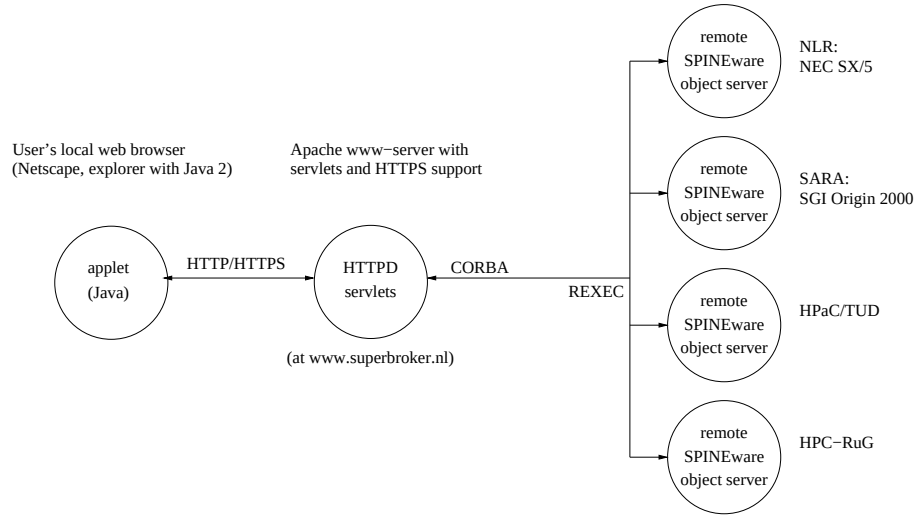


Fig. 4 Access to HPCN facilities through a web interface using SPINEware

help you to find your way around in the working environment and to locate and use the resources you need.

The *Superbroker* presently provides means to upload program code and data, to build executable programs and tools from the program code on the HPCN facilities, to run existing as well as your own programs, and to download results. The SPINEware object classes Queue (for Job queues) and Workflow (for chaining tools) offer useful possible extensions for working environments based on the Superbroker concept. The Queue class supports specification of jobs to be run on supercomputers, and to be managed by supercomputer-native job management systems, such as NQS and Codine, both of which are supported by SPINEware. The Workflow class supports definition of tool chains, which allow the use of tools and data involved to be specified in the form of a graph and to be reused by other users as well.

6 Conclusions

This paper described a general infrastructure for easy and uniform access to remote computing facilities from the desktop computer, using the desktop's native web browser as GUI. This infrastructure, SPINEware, has been applied to instantiate the *HPCN 2000 Superbroker* working environment.

The GUI is simply started by entering the URL of the start page in the native web browser. The GUI has the look and feel of a graphical file-system browser, enabling the user to browse

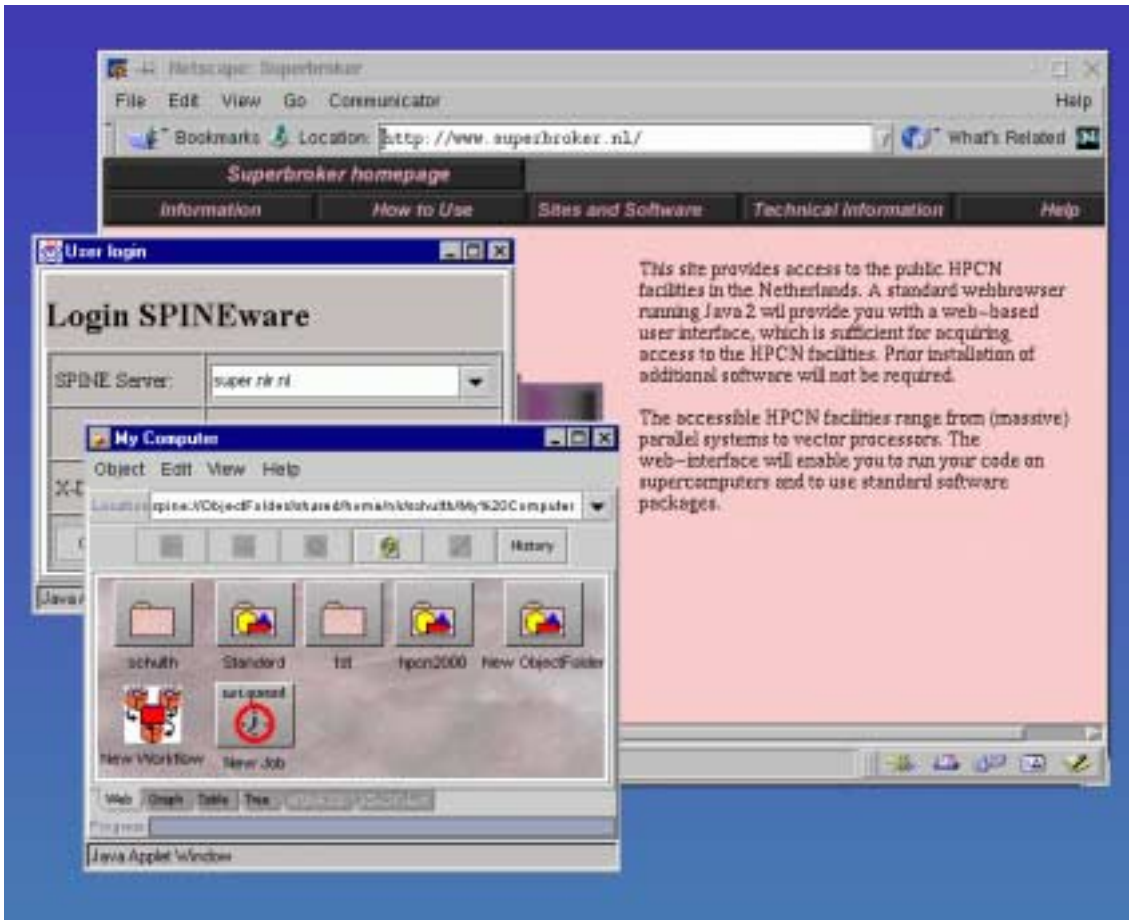


Fig. 5 Accessing HPCN facilities through the superbroker web site

through and access the available resources, such as files, directories/folders, and tools. Tools can be started, and jobs can be submitted using intuitive point-and-click and drag-and-drop operations. SPINEware takes care of the system details.

The end user only requires a Java-2 enabled web browser to access the defined working environments. The end user is able to enhance the working environments by integrating new tools, or adding new tool chains.

Behind the screen, a web-interface server (an HTTP daemon with Java servlets) functions as an intermediate between the end user's GUI (Java applets) and the SPINEware object servers on the remote hosts. Communication between the web-interface server and the Java applets is based on HTTP or HTTPS, thus minimizing firewall problems. The web-interface server communicates with the SPINEware object servers through CORBA. SPINEware object servers are automatically

launched on the hosts that are accessed during a session.

7 References

1. Orfali, R and D. Harkey, *Client/Server programming with Java and Corba*, ISBN 0-471-16351-1
2. Eckel, B, *Thinking in Java*, ISBN 0-13-659723-8, http://www.bruceeckel.com
3. *Java Servlets and Serialization with RMI*, http://developer.java.sun.com/developer/technicalArticles/RMI/rmi/
4. *ILU 2.0alpha14 Reference Manual*, Xerox Corporation. Can be obtained via ftp://beta.xerox.com/pub/ilu/ilu.html
5. Hunter J.; W. Crawford; *Java Servlet Programming*, ISBN 1-56592-391-X, O'Reilly & Associates.
6. *Apache HTTP server project*, http://www.apache.org/httpd.html.
7. Baalbergen, E.H. and H. van der Ven, *SPINEware - a framework for user-oriented and tailorable metacomputers*, in: Future Generation Computer Systems 15 (1999) pp. 549-558, NLR-TP-98643.
8. *Java Web Start*, http://java.sun.com/products/javawebstart/

Appendices

A Acronyms

CODINE	COmputing in DIstributed Network Environment, a job management system from GENIAS.
CORBA	Common Object Request Broker Architecture, see *http://www.omg.org*.
GUI	Graphical User Interface.
HPaC/TUD	High Performance Applications Center at Delft Technical University.
HTTP	HyperText Transfer Protocol.
HTTPD	HTTP Daemon or webinterface server.
HTTPS	Secure HTTP.
IDL	CORBA's Interface Definition Language.
IDLJ	Java's IDL-to-Java Compiler to generate Java bindings from a given IDL file.
IIOP	CORBA's Internet Inter-ORB Procotol. This is the protocol that must be used to provide interoperability with other CORBA implementations.
ILU	The Inter-Language Unification system, a free CORBA implementation from Xerox.
NLR	Dutch National Aerospace Laboratory.
NQS	Network Queing System, a job management system.
REXEC	Protocol to remotely start a process.
RUG	Rijksuniversiteit Groningen.
SARA	Academic Computing Services Amsterdam.
SME	Small and Medium Enterprises.
SPINEware	SPINEware is a system for the construction and operational use of functionally-integrated working environments in a computer network.
SPIRL	SPINEware resource locator.
URL	Universal Resource Locator, unique address of a document or a resource on the the internet in the form protocol://server domain name/pathname.