

Approaches for the integration of distributed
computation into the WWW

Michael Rumpf

December 1998

Abstract

This paper gives an overview of the architectures for smoothly integrating distributed computation into the World Wide Web (WWW). Of the four approaches presented, three deal with the integration on a very basic level. The last one deals with higher level problems like referential integrity and migrational transparency.

All of them aim at leading the WWW in the future by substituting the limited view of standard and non-standard resources as it is now.

Contents

1	The World Wide Web	3
1.1	Standard resources	3
1.2	Non-standard resources	3
1.3	Disadvantages	3
2	The WWW: A platform of distributed objects	4
2.1	The benefits of a class hierarchy	4
2.2	Integration of legacy systems	5
3	CORBA	6
3.1	Why CORBA ?	6
3.2	Basic concepts	6
4	Comparison: CORBA and WWW	8
4.1	HTTP-daemon vs ORB	8
4.2	Message formats	8
4.3	Mapping to TCP/IP	8
4.4	Interface	8
5	Strategies for introducing CORBA	9
5.1	Server side	9
5.2	Network	10
5.3	User side	10
5.4	System architecture	12
6	The ANSAWeb	15
6.1	IIOP and HTTP gateways	15
6.2	Native IIOP clients and servers	15
7	The CORBAWeb	17
7.1	Structure	17
7.2	Meta-scripts	18

8	Inter Language Unification (ILU)	21
8.1	Comparison to CORBA	21
8.2	String Binding Handles	21
8.3	Communication scenarios	22
9	W3Objects	24
9.1	Broken links	24
9.2	Properties	24
9.3	Client-object communication	24
9.4	Inter-object communication	24
9.5	Client-object communication using gateways	25
9.6	Object referencing	25
9.7	Reference chains	25
10	Conclusion	29
11	Glossary	30

Chapter 1

The World Wide Web

1.1 Standard resources

The WWW depicts a collection of standard and non-standard resources. Standard resources are e.g. HTML documents, GIF pictures and other types of static data a HTTP-server (HyperText Transfer Protocol) can serve directly.

1.2 Non-standard resources

The non-standard resources can be divided into server-sided and client-sided. Server-side non-standard resources need to be processed on the server-side and the result of this processing is delivered as static data to the client. Normally CGI-scripts are used for this server-side processing. Alternatives to CGI are e.g. ISAPI, NSAPI and Servlets.

The resources the client receives appear to him as static data. They can be processed by him by means of java-applets or plug-ins, but the browser itself is only able to handle standard-resources.

1.3 Disadvantages

One of the main disadvantages of server-side CGI-scripts is that a script has to be written for each new type of data which should be accessible for a user. The fact that on each user's request a new process has to be started and the connection is closed immediately after the data is transferred is a big problem due to the statelessness of TCP/IP.

A disadvantage of client-side plug-ins is that they are usually very large and therefore it takes a long time to download them.

This separation of standard and non-standard resources cannot be overcome by partial solutions. There is a need for a comprehensive solution, addressing all the current problems with regards to the WWW.

Chapter 2

The WWW: A platform of distributed objects

2.1 The benefits of a class hierarchy

A comprehensive solution for the problems of the WWW can be achieved by establishing a unique view on resource types. An object-oriented architecture would be a good solution to achieve this goal.

All resource types are classes with distinct interfaces. The resources itself are instances of these classes. A class has several interfaces and the number depends on the resource-type the class represents. An object-oriented class hierarchy offers common functionalities in the base classes and special functionalities in the derived classes of the hierarchy. This saves the developers from the time-consuming task of writing code for each new type of resource. They only have to implement the extensions to the base class the new type derives from.

A paradigm of object-orientation is the possibility of abstracting from special interfaces by seeing only those which are important for a specific operation on the object. E.g. a superclass provides an interface for object migration. The hierarchy separates between visible and invisible resources. For the visible resources there exists an interface which exposes a function for rendering the resource on some kind of media. All the subsequent types in the hierarchy own the migration interface and the visible ones additionally own the render interface. An administration tool can now abstract from the special types visible and invisible resources and access only the migrate interface which is used to move the resources from one location to another.

2.2 Integration of legacy systems

A big challenge in establishing an object-oriented view on the WWW is to maintain backward compatibility. The process of substitution should be an evolution and not a revolution and it therefore has to take care of all the existing technologies of the WWW.

An old browser should be able to access new services via server-side gateways and new clients should be able to access old servers as well as new ones.

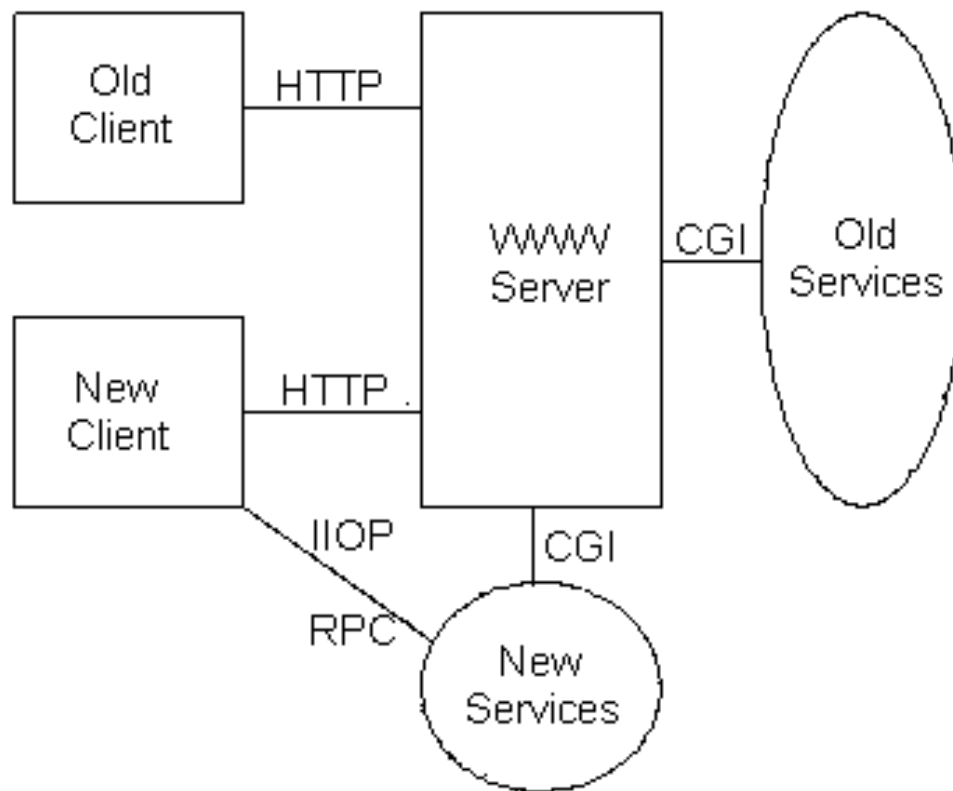


Figure 2.1: Integration of legacy systems

Chapter 3

CORBA

3.1 Why CORBA ?

Establishing such an object-oriented view on resources the underlying basis technology must be chosen with having several user requirements in mind. The most important ones are platform-availability and programming-language independence.

These requirements can only be guaranteed by a vendor independent technology like the Common Object Request Broker Architecture (CORBA). This is a platform for object distribution standardized by an industry consortium, the Object Management Group (OMG), having currently more than 700 members.

3.2 Basic concepts

The basic function of CORBA is provided by an Object Request Broker (ORB). The ORB is responsible for the network communication between objects.

The process of writing CORBA-objects can be divided into the following steps:

1. describe the interfaces of the server-object in the Interface Definition Language (IDL)
2. run the language compiler to generate the client-stub and the server-skeleton
3. implement the server and link the server-skeleton and the ORB to your code
4. implement the client and link the client-stub and the ORB to your code

The process of making a call to the server can be described as follows: The client-stub is the representation of the server on the client side. Instead of calling the server directly, the client calls the stub, which is located in the client's process. The stub communicates with the ORB. The ORB forwards the call over the network to the server's ORB which communicates with the server-skeleton to pass the call to the actual server-object. The marshalling and unmarshalling of the parameters of the call are totally transparent to the caller and the server and it is managed completely by the ORB.

There is yet another possibility for accessing objects. If the server-object is written after the client-object, the client cannot use the client-stub because it is not available at the time the client-object is written. The way to solve this problem is called Dynamic Method Invocation (DMI). The server registers itself in an Interface Repository (IR) which the client can browse through and search in for specific interfaces. Once he has found an interface serving his needs he can call a function of the server dynamically.

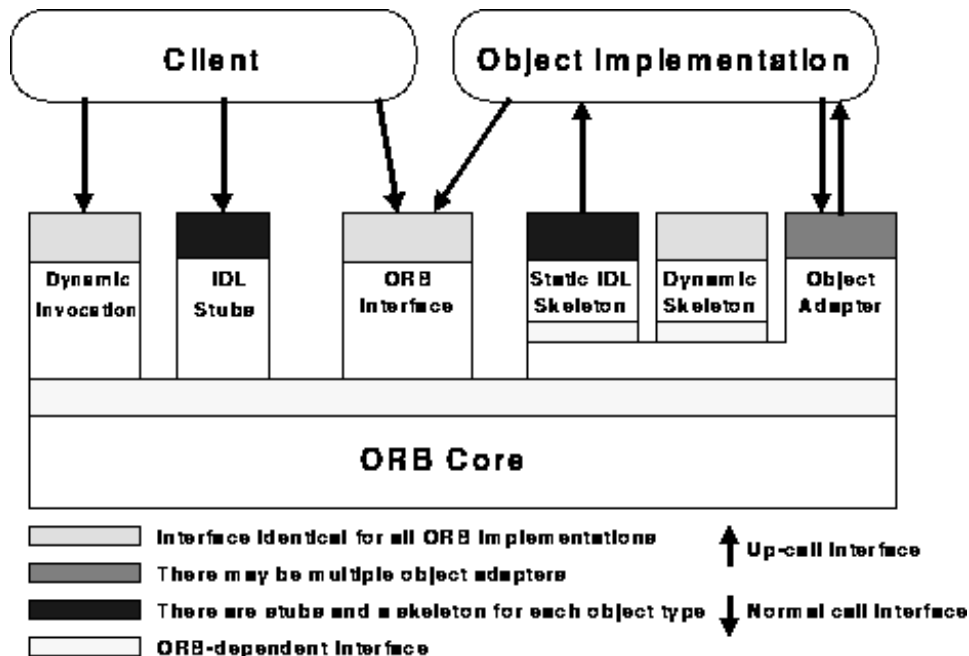


Figure 3.1: CORBA architecture

Chapter 4

Comparison: CORBA and WWW

4.1 HTTP-daemon vs ORB

The HTTP-Daemon of the WWW-server is responsible for the communication. In CORBA, this task is implemented by the ORB. It receives the requests from clients and passes back the results of an operation.

4.2 Message formats

The message format of the protocol is defined by HTTP itself whereas CORBA has a level of abstraction on top. The messages are independent from the network protocol, they are defined by the General Inter ORB Protocol (GIOP).

4.3 Mapping to TCP/IP

The way the messages are mapped into the protocol structures of TCP/IP is defined in HTTP itself. CORBA does this by specifying a concrete mapping to the actual network protocol, using the Internet Inter ORB Protocol (IIOP).

4.4 Interface

The current architecture lacks the most in HTTP and the operations defined by HTTP. All the resources have the limited interface with the functions GET, HEAD, PUT and POST. With CORBA the resources can have an appropriate interface dependent on the type.

Chapter 5

Strategies for introducing CORBA

The different strategies for the integration of CORBA into the WWW presented in this chapter are taken from [1].

Each of these solutions addresses one part of the system, i.e. the user-side, the server-side and the network infrastructure (fig. 5.1).

5.1 Server side

In figure 5.2 a specific WWW server gateway is shown. Such a gateway uses CORBA's static invocation scheme. The CORBA-objects have to be linked at the time when the gateway is compiled. To add any new objects, the gateway must be recompiled, the server needs to be stopped and started again to use the recompiled gateway.

Figure 5.3 shows a generic gateway which uses CORBA's dynamic interface invocation to access new objects when they become available. It is

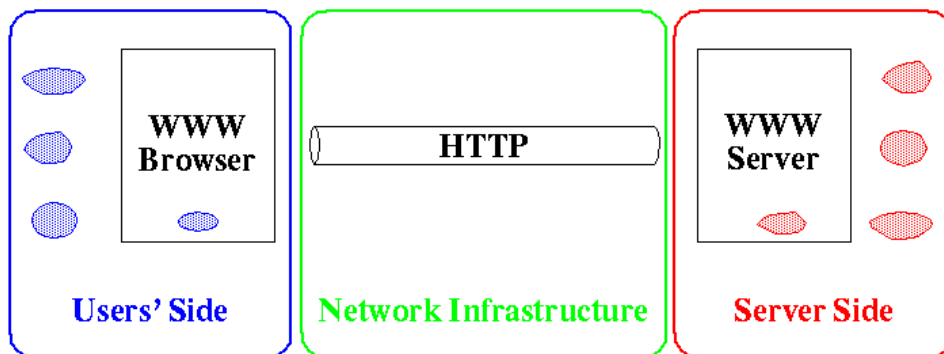


Figure 5.1: Division of the WWW into three layers

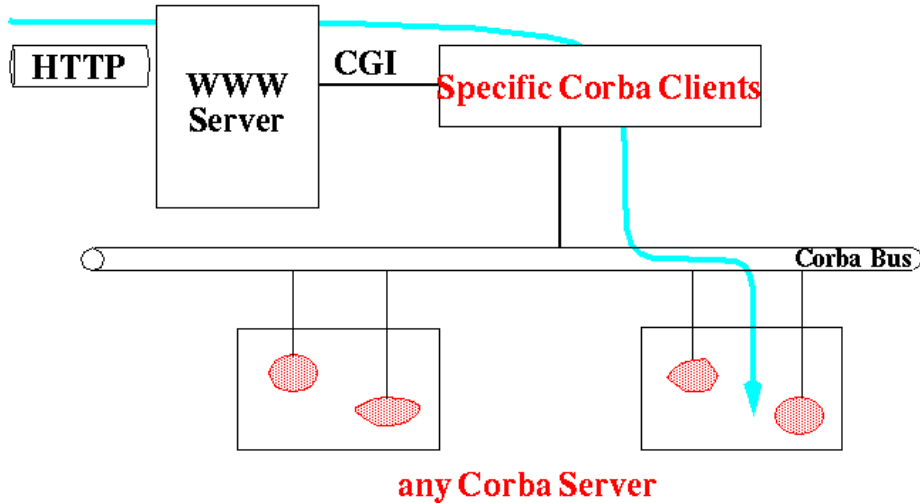


Figure 5.2: Accessing objects with a specific CORBA gateway

not necessary to change the configuration of the WWW server. New objects can immediately be introduced.

5.2 Network

The scenario in figure 5.4 addresses the network infrastructure and shows a protocol translation between HTTP and IIOP. Such a protocol-converter provides support for legacy systems which are not able to use IIOP. Any new browser or server should be aware of IIOP as a native protocol.

5.3 User side

In figure 5.5 Java applets are used to make a connection to server-side CORBA objects. The applets communicate directly with the objects on the server. For communication either HTTP or IIOP can be used. A direct IIOP connection can be selected for sending notification-messages or passing asynchronous events from the server to the user.

Figure 5.6 illustrates how Java applets access the user's desktop directly and interact with local CORBA server objects. This covers the introduction of CORBA on the user's side.

Figure 5.7 indicates the communication between applets and different browsers. This is a typical scenario for groupware applications where members of a group need to communicate and to synchronize their work over a network.

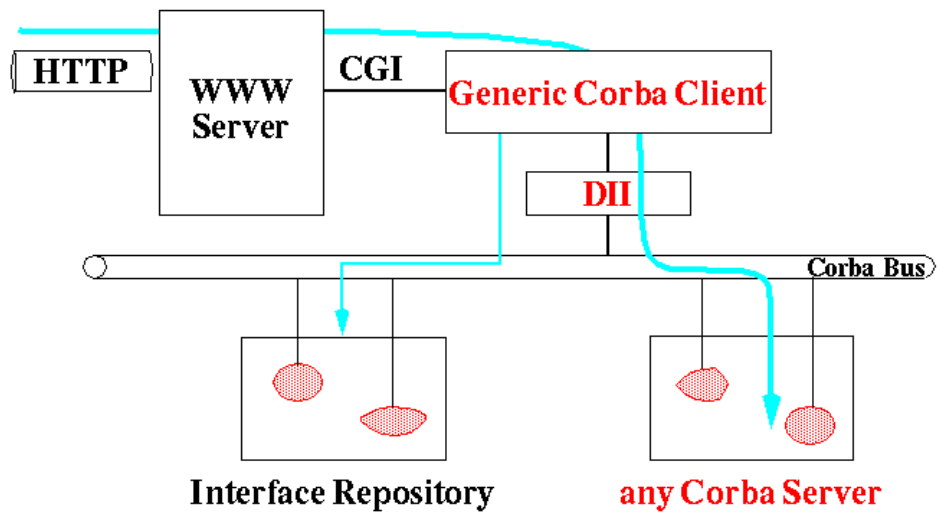


Figure 5.3: Using DII to implement a generic CORBA gateway

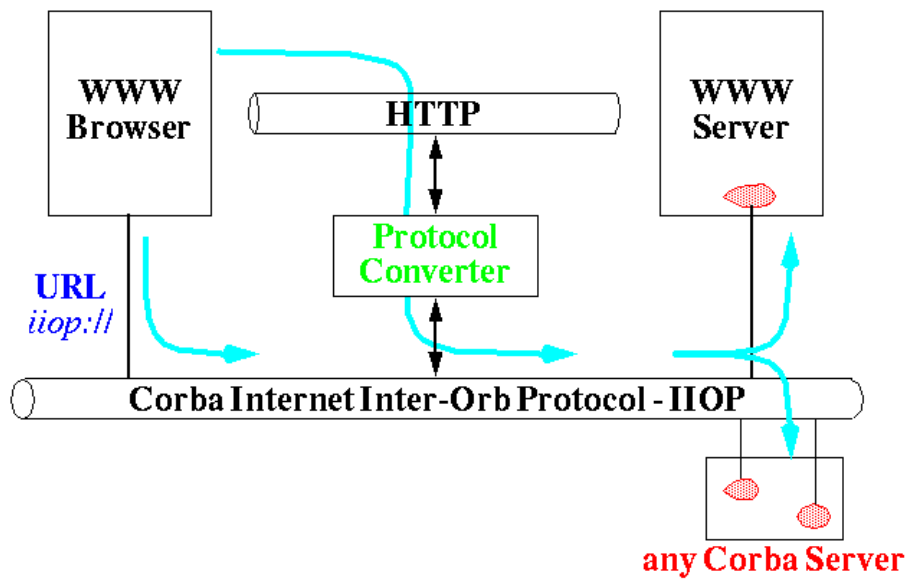


Figure 5.4: Protocol translation between HTTP and IIOP

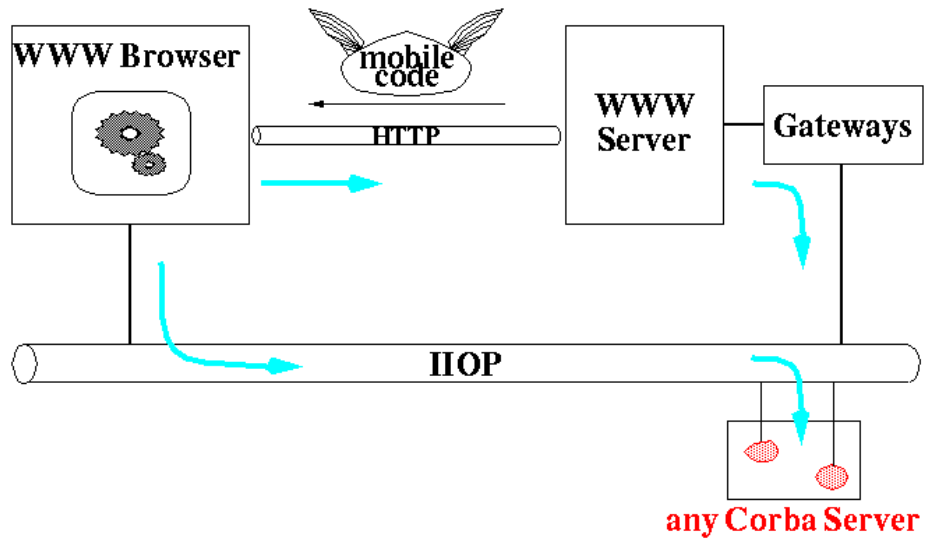


Figure 5.5: Using IIOP aware mobile code on the client-side

5.4 System architecture

Figure 5.8 describes the overall architecture after the integration of CORBA objects into the bespoke parts of the web. IIOP and HTTP serve as basic communication protocols and all connections possible can be established by using these protocols.

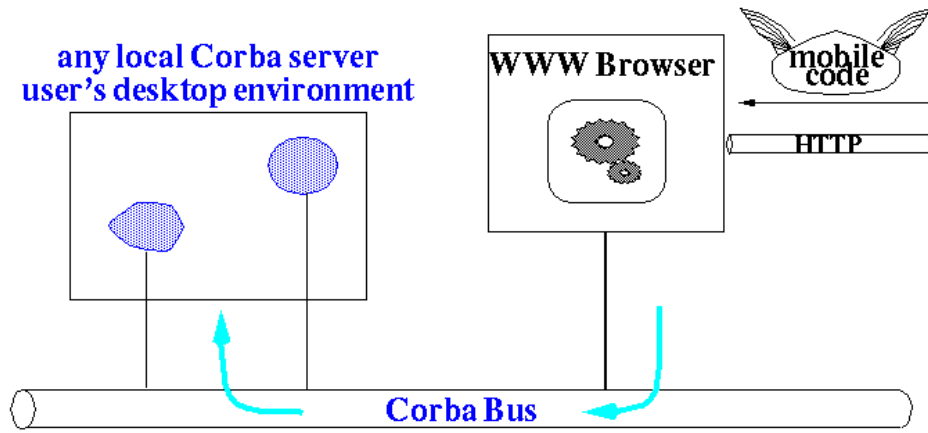


Figure 5.6: Accessing the user's desktop environment

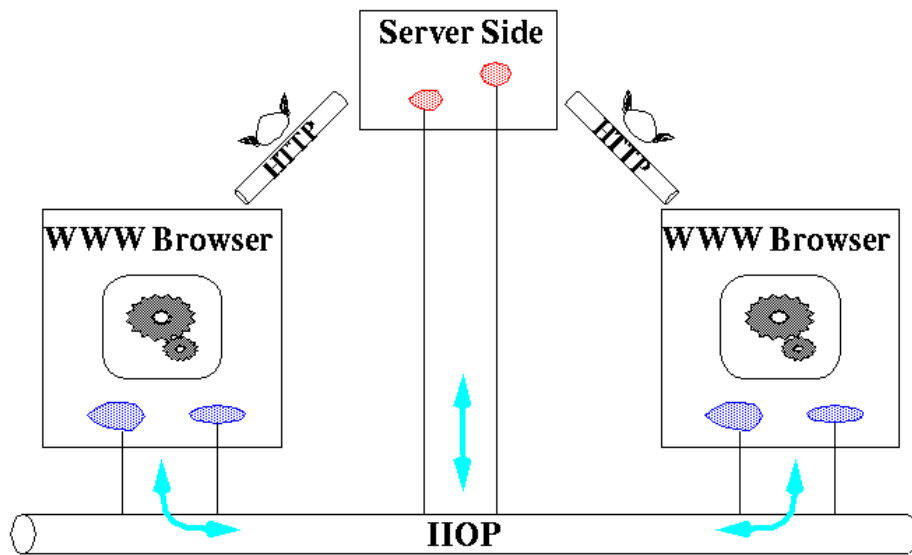


Figure 5.7: Communication between browsers

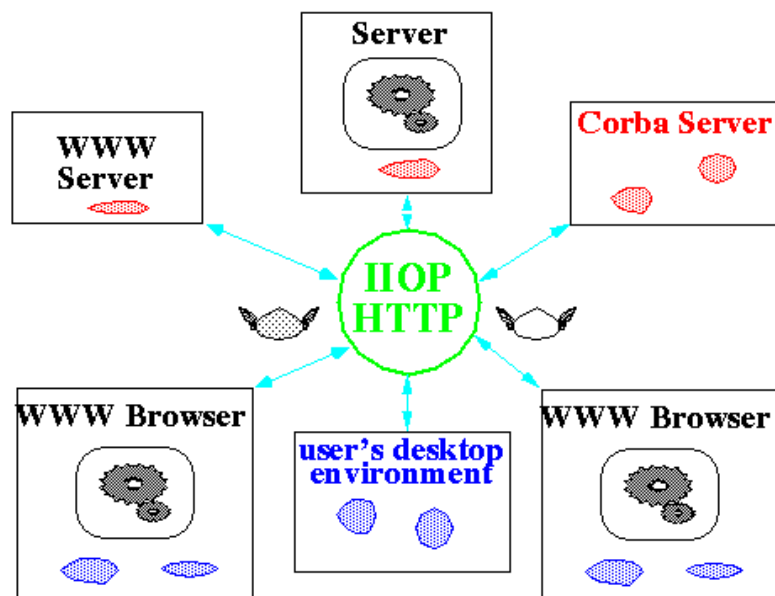


Figure 5.8: Overall architecture

Chapter 6

The ANSAWeb

6.1 IIOP and HTTP gateways

This solution by APM Ltd. addresses the network layer. ANSAWeb (Advanced Network System Architecture) introduces IIOP to HTTP and HTTP to IIOP converters. The first step of the proposed architecture is to substitute the HTTP connection between a normal browser and a normal HTTP-server by the protocol converters shown in figure 6.1.

The HTTP2IIOP bridge is located on the machine where the browser is running and the IIOP2HTTP bridge should run on the HTTP-server. This is the first step of the integration. The HTTP2IIOP bridge could be used as a HTTP-proxy which decides which route to choose and which protocol to use. If the server has no IIOP2HTTP bridge, the proxy has to use HTTP and in this case it acts as a normal HTTP-proxy. If the server has a IIOP2HTTP bridge, the HTTP2IIOP bridge can use IIOP instead. But how does the HTTP2IIOP bridge know which protocol the server supports? This information is provided by the locator, as shown in figure 6.2, which is implemented as a separate service.

The question which remains to be solved is how the locator gets the data. One possible way is to try IIOP first and fall back to HTTP if the server cannot understand IIOP. Another possibility is to send an extra header in HTTP responses. The disadvantage is that the server has to be modified. The third way would be to ask a Trader whether there is a suitable IIOP service available.

6.2 Native IIOP clients and servers

The second stage of the integration is to implement native IIOP servers and clients. The results of this architecture are shown in figure 6.3.

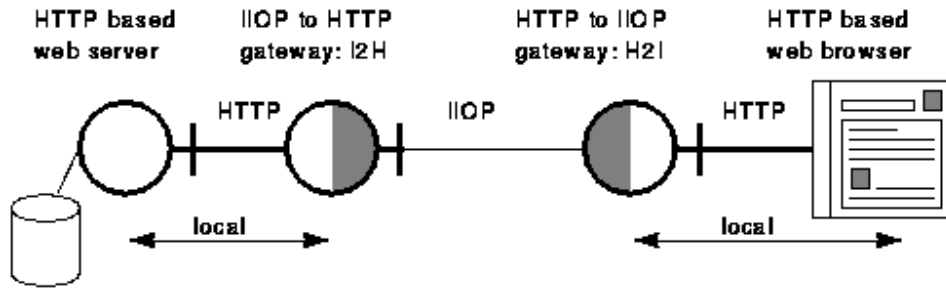


Figure 6.1: Introduction of gateways for protocol translation

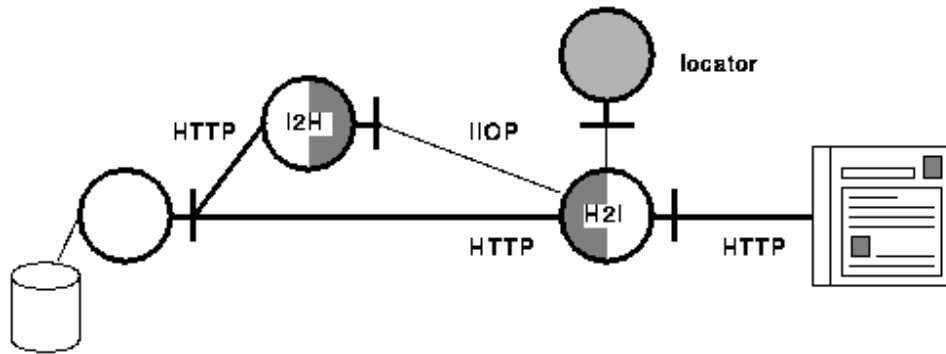


Figure 6.2: Using locators as a lookup service

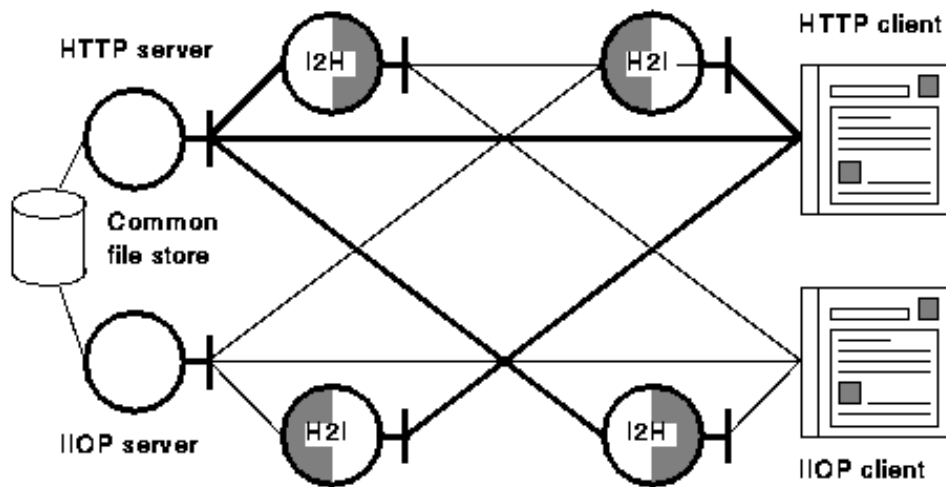


Figure 6.3: Interaction of IIOP and non IIOP clients and servers

Chapter 7

The CORBAWeb

7.1 Structure

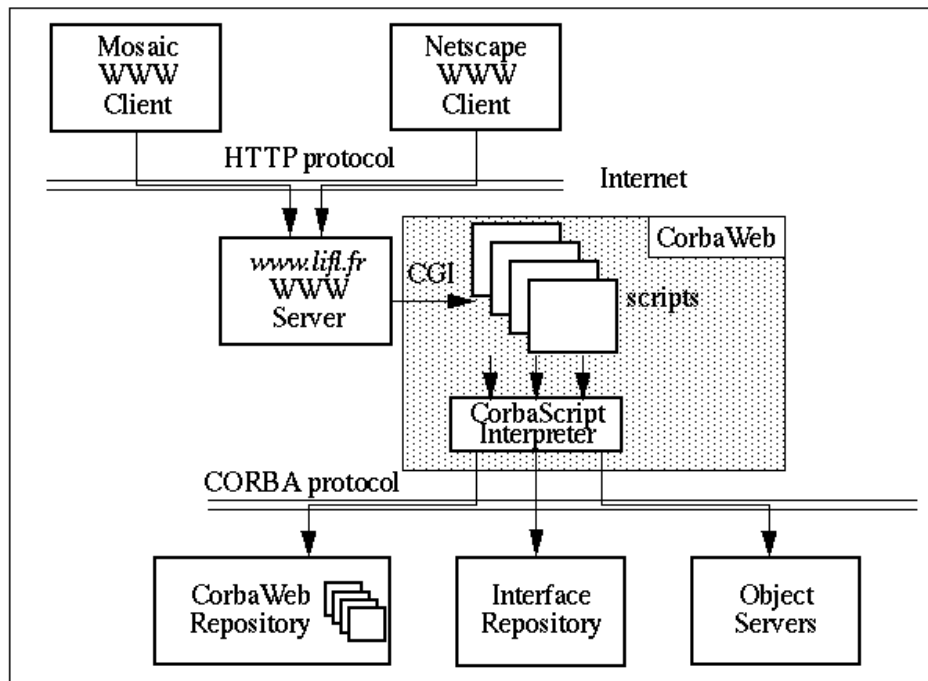


Figure 7.1: Gateway architecture

The CORBAWeb of the University of Lille in France addresses server-side processing. Accessing CORBA-objects is achieved by using a special scripting language, called CORBAScript. In figure 7.1 an overview of the system architecture is presented. The main components are the CORBAScript interpreter and the meta-script repository. The interface repository and the object-servers itself are CORBA components.

The CORBAScript interpreter executes the scripts from the meta-script repository. The main reason for inventing a completely new language is to hide the complexity of using the DII in a common programming language like C++ or Java. Compared to traditional CGI-programming a new language only is not a satisfactory solution because a script has to be written for each object like in any other language. This problem can be solved by providing so-called meta-scripts which allow to call methods on CORBA-objects in a generic way.

7.2 Meta-scripts

There are three different types of meta-scripts, the exec-, the interface-, and the view meta-script. The following piece of code shows the exec-script which is used to invoke an operation on a CORBA-object. The syntax of this new language is similar to common programming languages. Additionally, the `execute` function performs the dynamic interface invocation and the `EXCEPTION` variable contains the error information if an error occurred during the function call. If the operation succeeds, the `execute` function generates a HTML page presenting the result as shown in figure 7.3.

```
#!/usr/bin/cwsh
echo "Content-Type: text/html\n\n<HTML>\n"
if $QUERY_STRING <> "" then
    echo "<TITLE>", $QUERY_STRING, "</TITLE>\n<BODY>\n"
    try
        execute ($QUERY_STRING)
    catchany
        echo "Exception raised: ", $EXCEPTION
    end
    echo "</BODY>\n</HTML>\n"
else
    # generates HTML Form for user's input
end
```

The interface-script generates an HTML page with a form for each method and attribute a CORBA object exposes to the client (see figure 7.2). The form contains an input field for each method parameter and a button to call the method. The action parameter of the `<FORM>`-tag contains the exec-script which will be executed when the appropriate button is pressed (e.g. `<FORM ACTION='.../interface.csh'>...</FORM>`).

The view meta-script provides a view of the current state of an object. A user profile is used to show only those parts of an object's state which are accessible for a specific user.

With these scripts a user can browse through CORBA objects in the same and easy way as browsing through the WWW.

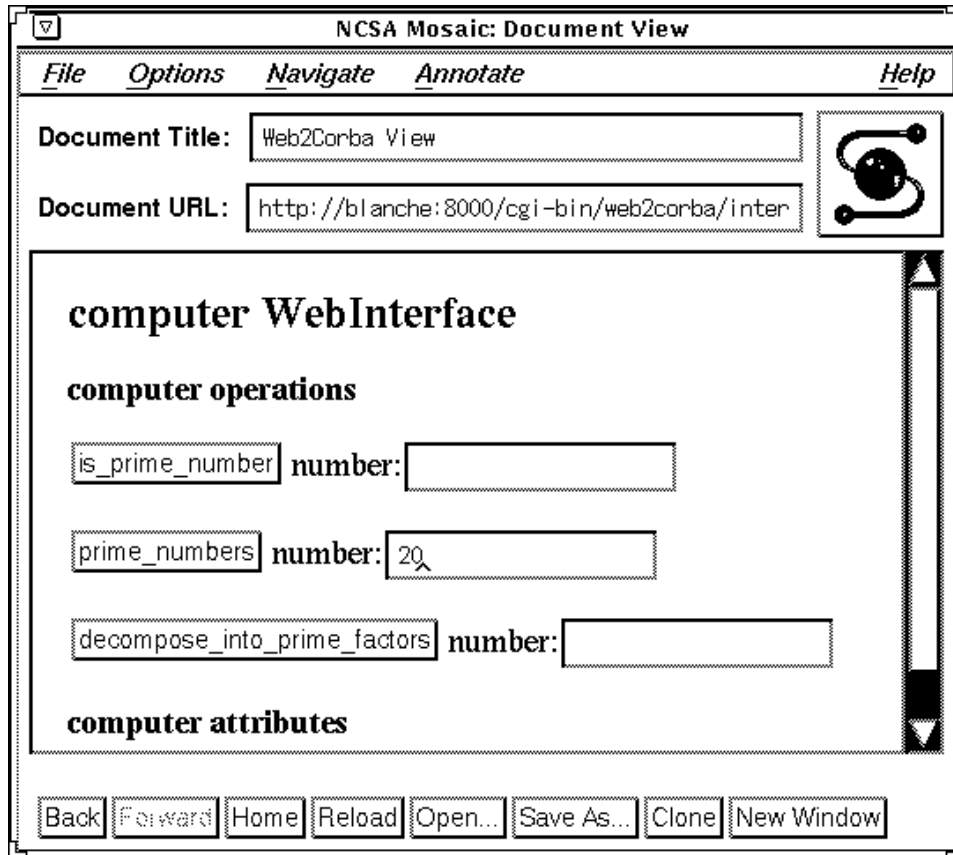


Figure 7.2: A HTML-form with an object's interface

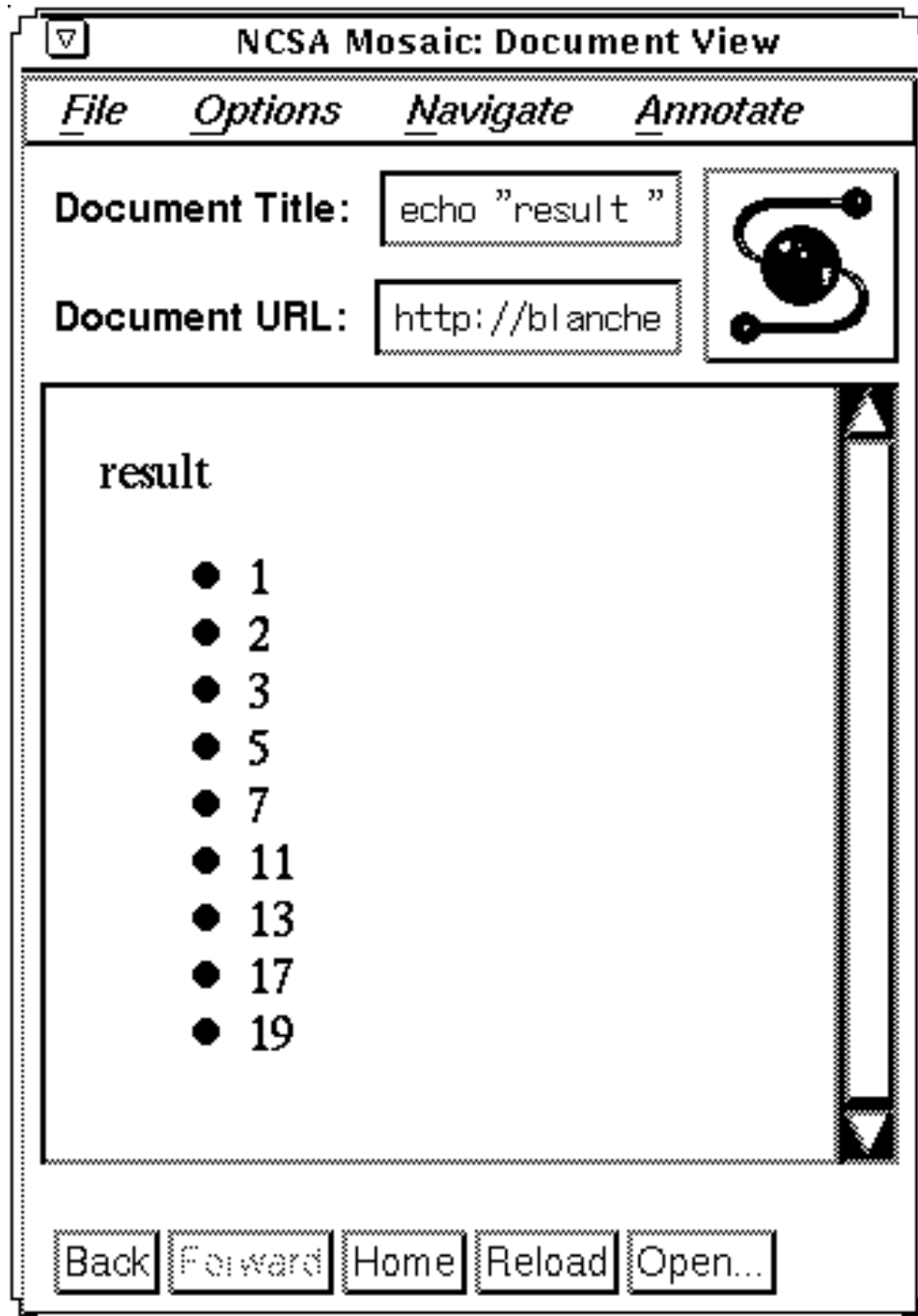


Figure 7.3: The result of a function call

Chapter 8

Inter Language Unification (ILU)

8.1 Comparison to CORBA

ILU is based on the CORBA but offers the following features additionally:

- ILU conforms to the CORBA specification and it is available in binary and in source-code form at no costs
- a lot of different protocols are supported, like SUN's ONC RPC, Xerox's Courier, HTTP, IIOP, TCP and UDP
- the Interface Specification Language (ISL) is more powerful than CORBA's IDL; especially Singletons which allow legacy services to be treated as objects and Siblings which force objects to be executed on the same machine
- most CORBA environments support only the most common programming languages. ILU can be used with nearly all programming languages, even in the same address space.
- ILU runs on any UNIX and Windows operating system
- ILU provides efficient intra-process communication and distributed garbage collection, which means that server objects are shut down if no client objects are keeping references anymore

8.2 String Binding Handles

The key to the integration of objects into the WWW is the similarity between a String Binding Handle (SBH) and a Uniform Resource Locator (URL).

SBH example:

```
ilu:TimingTest.figtree.parc.xerox.com/0;ilu%3Ac0FGHuC8UdTA0+
ETWz1N15RjWa3;sunrpc_2_0x61a78_1139713249@sunrpcrm=tcp_13.1.
100.126_1588
```

URL example:

```
http://www.parc.xerox.com:80/index.html
```

Description:

- the keywords `ilu:` and `http:` refer to the scheme identifier which indicates what protocol to use
- the object's server ID is `TimingTest.figtree.parc.xerox.com` which addresses one server entity i.e. a container for single objects
- the address `www.parc.xerox.com` is the host name which is mapped to an IP address using a Domain Name Service (DNS)
- the number `0` denotes an object inside the server and `80` is the TCP port number of the HTTP connection
- `ilu%3Ac0FGHuC8UdTA0+ETWz1N15RjWa3` is a code for the object's base type.
- `sunrpc_2_0x61a78_1139713249` denotes the protocol and additional parameters used to operate on an object
- `sunrpcrm=tcp_13.1.100.126_1588` identifies the protocol used for connecting to the object, SUN RPC and specifies the location of the object at IP address `13.1.100.126` on TCP port `1588`
- the filename `/index.html` indicates the resource on which an HTTP operation should be performed

8.3 Communication scenarios

An ILU server-object which understands HTTP can easily be implemented. A HTTP request to this server, like in the URL example above, will be redirected to the object `/index.html` instead of `/0`. The object itself exposes an interface with the common functions GET, HEAD, POST and PUT. The method implementation and therefore the result, i.e. the HTML output, is determined by the programmer and differs from one HTTP resource to another.

The three different inter-object communication scenarios are: ILU client to ILU server, ILU client to HTTP server, and HTTP client to ILU server.

1. **ILU client - ILU server**

The server-object is represented by a kind of proxy, a so-called surrogate object, on the client-side. The surrogate object manages all the necessary network communication and the marshalling of the method parameters

2. **ILU client - HTTP server**

If the server understands only HTTP the generated surrogate transforms the client's request into a HTTP request and the result of the HTTP server back into a form the client is able to understand

3. **HTTP client - ILU server**

The client passes only HTTP requests to the server. The ILU server object needs to expose a HTTP interface to understand these requests

Chapter 9

W3Objects

9.1 Broken links

This approach addresses problems not discussed in previous chapters. It is no further distributed object technology. CORBA or ILU can be the basis technology for W3Objects.

The current WWW lacks referential integrity, better known as the problem of ‘broken links’. This problem is caused by a lack of migrational transparency. Each time a WWW resource is deleted or moved to another place and a reference to this resource still exists, a broken link is the result.

9.2 Properties

The specification of W3Objects has identified a set of properties. These properties have the same basic functionality like the CORBA services defined by the OMG. If W3Objects are implemented on top of CORBA, the CORBA services can easily be used instead.

9.3 Client-object communication

Figure 9.1 shows a W3Objects server and two clients accessing the server. The server can serve RPC and HTTP clients simultaneously. It exposes a HTTP communication end point where the HTTP protocol is converted into internal method invocations. A RPC client calls the W3Object’s server functions immediately.

9.4 Inter-object communication

The architecture also supports inter-object communication. It does not make any difference if the objects are located in the same W3Objects server,

in different servers on the same machine or in different servers on different machines (see figure 9.2). W3Objects servers can contain stubs which are responsible for forwarding method invocations to another stub or to the real object. This forwarding mechanism can be used for referencing, migration, caching, and replication.

9.5 Client-object communication using gateways

Figure 9.3 explains the need to have different servers with different types of W3Objects. This is a result of the fact, that not all object types can be identified at compilation time of a server.

When introducing new objects, the old version of the server has to be recompiled and started again. To reduce the downtime of a server, it is more recommendable to launch a second server with the new types of objects and to use a HTTP gateway which redirects the requests to the desired object.

9.6 Object referencing

Objects in a W3Object server are referenced from a root within the server. If a stub, a synonym for a hypertext link, is created, he performs an explicit *bind* operation on the object the link refers to. When the stub is deleted, an *unbind* operation is invoked on the referenced object. If the reference counter is equal to zero, the object is available for deletion. This method ensures that once an object is deleted no other links are pointing to it.

9.7 Reference chains

The problem of migrational transparency can be solved by using reference chains. Each time an object is moved, a stub is created and resides on the old

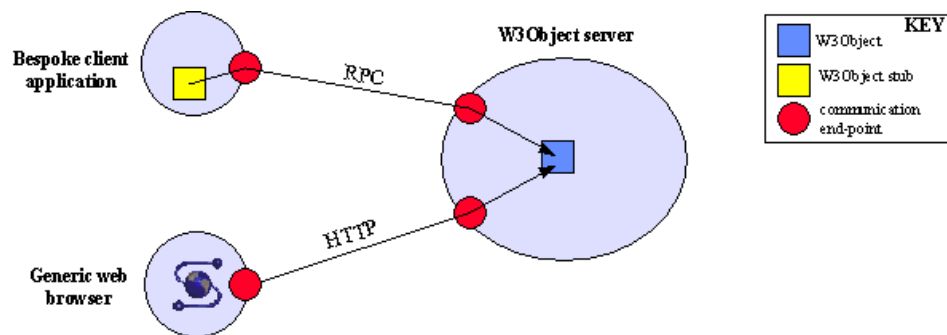


Figure 9.1: Client-object communication

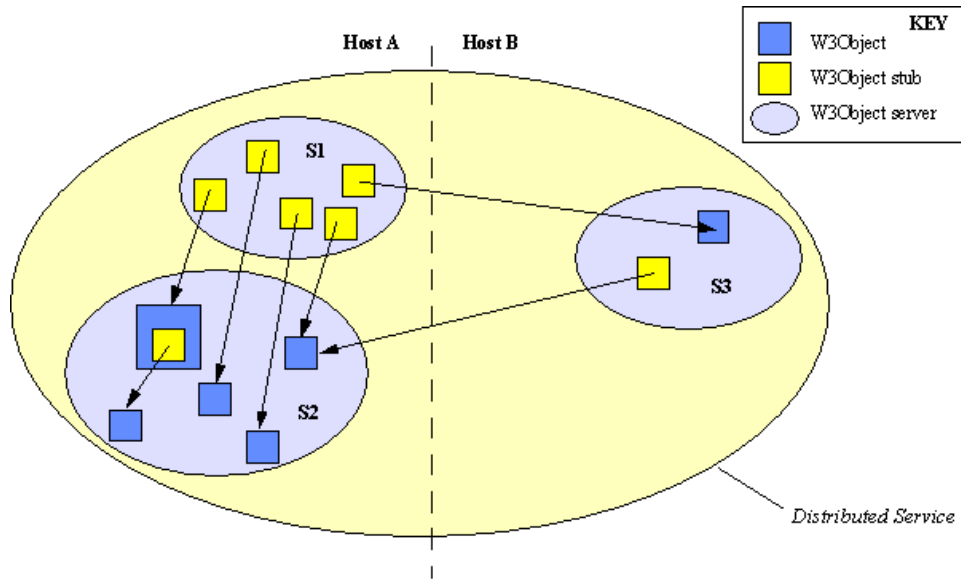


Figure 9.2: Inter-object communication

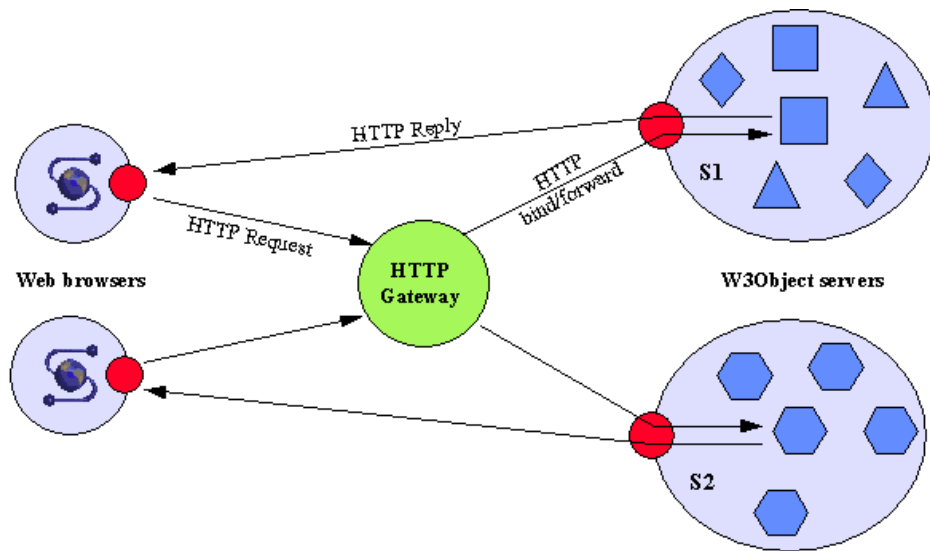


Figure 9.3: Client-object communication using gateways

Figure 9.4: Object referencing

location. When a call to the old location is performed, the reference is short-cut and it is updated on the client side. When all the clients have accessed the object at its new location at least once, the references are automatically updated and the stubs can be deleted.

Figure 9.5: Reference chains

Chapter 10

Conclusion

The first three solutions address only the issue of integrating objects into the WWW. CORBA and ILU provide a basic architecture of distributed objects. For CorbaWeb, ANSAWeb and W3Objects it is possible to use CORBA or ILU as an implementation technology. The last one, W3Objects, tries to solve the most immanent problems like referential integrity and migrational transparency.

All the projects have already been presented at conferences a few years ago and are still under improvement. Even prototypes for each approach are available, but a real breakthrough is not in sight. The drawback is that the object oriented view on a resource has not yet been established. The eXtended Markup Language (XML) and the Document Object Model (DOM) may have a big influence on the further development because each document is being transformed into an object which is accessible through a set of interfaces, exposed by the DOM.

The approaches presented can be used to develop a better solution for an object oriented WWW by taking the main components out of each project. The protocol gateways from ANSAWeb, the object browsing capability of CorbaWeb, the naming scheme of ILU and the solution for referential integrity and migrational transparency from W3Objects. A profound architecture will be the result and will help to simplify the WWW for both, users and web-developers. Users will no longer waste bandwidth and time because of broken links and WWW developers do not need to worry about broken links when a resource is moved.

Chapter 11

Glossary

ANSA: The Advanced Network System Architecture is a network-protocol layer solution from APM Ltd. for the integration of CORBA objects into the WWW.

CGI: The Common Gateway Interface is a server side solution to extent the static delivery of documents by dynamic processing capabilities.

CORBA: Common Object Request Broker Architecture (CORBA) is a vendor neutral standard for object distribution, proposed by the OMG.

DII: Dynamic Interface Invocation (DOM) is a system to invoke methods without further knowledge of the object's type.

DOM: The Document Object Model (DOM) is an API for accessing documents written in HTML, XML or SGML.

GIF: The Graphics Interchange Format (GIF) is a widely used graphics format on the WWW.

GIOP: The General Inter Object Protocol (GIOP) is a CORBA protocol of object

HTML: The HyperText Markup Language (HTML) is a subset of SGML and is used for writing documents for the WWW.

HTTP: The Hypertext Transfer Protocol is a protocol especially used for transferring HTML documents with TCP/IP over the internet.

IIOP: The Internet Inter Object Protocol defines the mapping of the GIOP to TCP/IP.

ILU: Inter-Language-Unification is a solution for object distribution, similar to CORBA.

IR: The Interface Repository (IR) is a repository where server-objects can register their interfaces to make it possible for older clients to get access to them.

ISAPI: Information Server API (ISAPI) is an API for integrating server-side computation into the Microsoft Internet Information server.

NSAPI: Netscape Server API (NSAPI) is an API for integrating server-side computation into the Netscape WWW server.

OMG: The Object Management Group (OMG) is a group of independent companies standardizing the Common Object Request Architecture.

ORB: The Object Request Broker (ORB) is responsible for the network communication and all the related tasks.

RPC: A Remote Procedure Call (RPC) is a call to a function which is implemented in an application that is running in a different process or on a different machine.

SBH: A String Binding Handle (SBH) is a naming component of ILU, which is comparable to a URL.

SGML: Standard General Markup Language (SGML) is the superset of XML and HTML, offering complex document management features.

TCP/IP: This stateless protocol is the standard protocol of the internet.

URL: A Uniform Resource Locator (URL) is a name for a resource on the internet.

WWW: World Wide Web

XML: The eXtended Markup Language (XML) is a subset of SGML providing special functionality for WWW documents.

Bibliography

- [1] P. Merle: *Integrating Corba Objects in the WWW*, at the Fifth International World-Wide Web Conference on Distributed Object Technology and the Web, Paris, France, May 1996, URL: http://corbaweb.lifl.fr/papers/96_WWW5/panel/slides.html
- [2] P. Merle, C. Gansart and J.M. Geib: *Future Trends for a Global Object Web*, in Joint W3C/OMG Workshop on Distributed Objects and Mobile Code, Boston, Massachusetts, June 24-25, 1996 (June 1996), OMG/W3C, URL: http://corbaweb.lifl.fr/papers/96_OMG-W3C.html
- [3] P. Merle, C. Gansart and J.M. Geib: *CorbaWeb: A Generic Object Navigator*, at the Fifth International World-Wide Web Conference on Distributed Object Technology and the Web, Paris, France, May 1996, URL: http://corbaweb.lifl.fr/papers/96_WWW5/paper/Overview.html
- [4] D.B. Ingham, M.C. Little, S.J. Caughey and S.K. Shrivastava: *W3Objects: Bringing Object-Oriented Technology to the Web* URL: <http://www.w3j.com/1/Ingham.141/paper/141.html>
- [5] D. Larner: *Migrating the Web toward Distributed Objects*, URL: <ftp://ftp.parc.xerox.com/pub/ilu/misc/webilu.html>
- [6] O. Rees, N. Edwards, M. Madsen, A. McGlenaghan: *A Web of Distributed Objects*, at the Fourth International Web Conference, December 1995, URL: <http://www.w3j.com/1/rtor.085/paper/085.html>
- [7] APM Ltd.: *Advanced Network System Architecture*, URL: <http://www.ansa.co.uk/ANSA/ISF/overview.html>
- [8] Object Management Group: *CORBA Overview*, URL: <http://www.infosys.tuwien.ac.at/Research/Corba/OMG/arch2.html>