Reworking OHP: the Road to OHP-Nav

D. E. Millard, S. Reich, H. C. Davis
Multimedia Research Group
The University of Southampton
SO17 1BJ, Southampton UK
{dem97r, sr, hcd}@ecs.soton.ac.uk

Abstract

The original OHP specification was intended to define a simple but comprehensive ASCII protocol to deal with hypertext objects. However, this was too broad an issue to be addressed by a single protocol. Therefore, we have focused on navigational hypertext and produced a specification that is more powerful within this more specific domain. This paper describes the design of OHP-Nav, defined both in IDL as an interface and in XML as a tagged ASCII protocol. We believe that this reworked definition can form a basis for focusing on the real issues of interoperability in open hypermedia systems.

Introduction

The Open Hypermedia Systems Working Group (OHSWG) has promoted research towards interoperability amongst hypermedia systems and integration with the World-Wide Web. A number of sub groups have been established to address the various issues.

In this paper we describe the development and current status of OHP-Nav, a standardized set of services for retrieving and navigating hypermedia objects that evolved from OHP, the open hypermedia protocol, first presented at the 2nd Workshop on Open Hypermedia Systems at Hypertext '96 in Washington, DC (Davis et al., 1996) and later refined at the OHSWG meeting the following November in Southampton, England (Davis, 1996). Here further issues were identified and work began on both the protocol and a reference architecture. At the last OHSWG meeting in Aarhus, Denmark, in September 97, it was decided that we should be considering a suite of interfaces, rather then a single protocol (Wiil, 1997). What used to be OHP has become the OHP Navigational Interface (OHP-Nav for short). The latest version of the interface definition (Davis et al., 1997) was considered complete enough for implementations to begin, with the aim of exploring the model.

This paper describes the current set of components that have been implemented by the Multimedia Research Group in Southampton, UK. It explains the decisions made for developing an XML conformant document type definition for talking OHP-Nav messages as XML documents and shows how we have mapped the ASCII protocol onto a CORBA compliant interface definition in IDL. Finally we illustrate conclusions and further issues that have come up during development.

Current State of the Southampton Foo

The system being developed by the Multimedia Research Group in Southampton currently comprises of an OHP-Nav aware jpeg viewer, the CSF itself and the linkserver (see Fig. 1). All communication is done through TCP/IP sockets, using a 19 byte leading integer to indicate the length of the message stream. All testing has been done on Windows NT platforms but the three main components are written in Java and should be portable to other systems with the minimum of effort.

The jpeg viewer is an application that supports filesystem and Internet retrieval of documents, the following of links and the creation of new endpoints. It does not have a cache or any editing facilities, allthough these could be added at a later date.

The linkserver uses JDBC into an Access Database, which stores OHP-Nav objects (endpoints, nodes etc).

The CSF we have implemented uses a toast-rack architecture of slot in communication mechanisms to talk a variety of network protocols, either to client or server sides. Currently we have implemented the TCP/IP sockets mechanism and a CORBA compliant mechanism is also planned in line with the OHSWG proposals for a component based approach (see the OHSWG framework pages). It also functions with other protocols such as the CSF protocol described below.

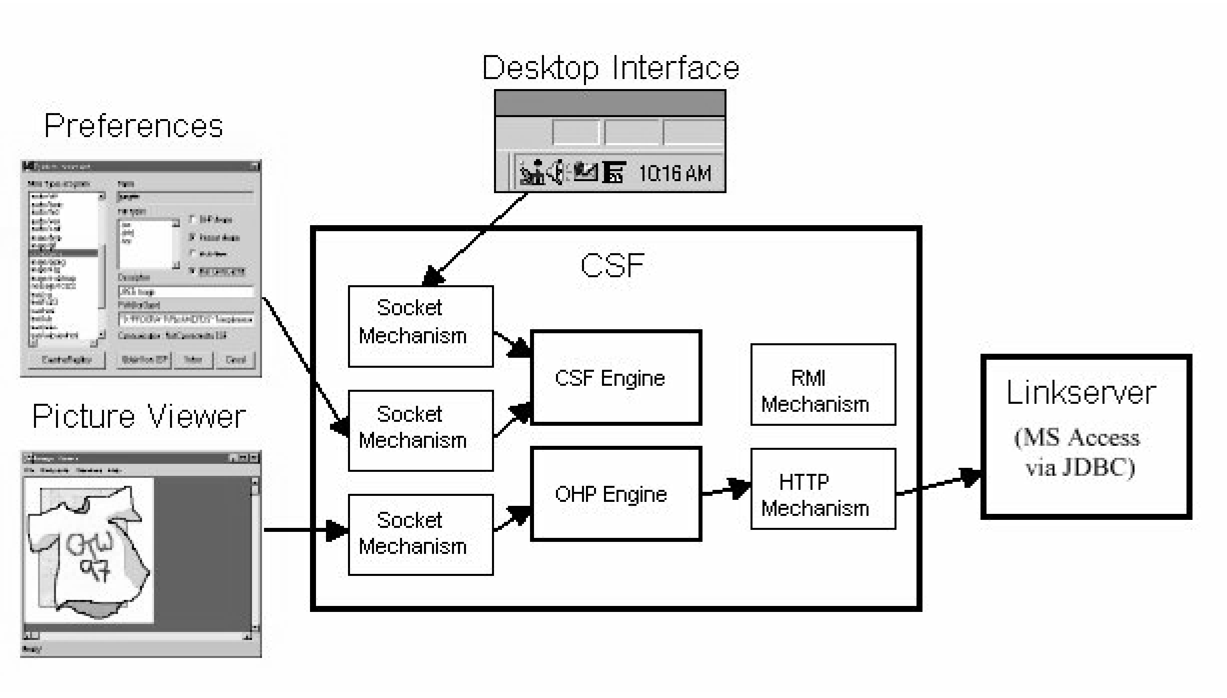

Fig 1. The Southampton Foo Configuration.

The definition of OHP-Nav in XML

The implementation of the prototypes as described above has raised several issues regarding the parsing of the tagged ASCII protocol definition. These issues included some inconsistencies in the definition among them the naming of tags and the handling of objects. Also, the message header contained insufficient information about the content of the message to allow efficient routing. Most importantly, the message definition was non deterministic which made message parsing difficult.

For these reasons we decided to define an XML (Bray et al., 1998) conformant document type definition called the OHP-Nav.dtd (the complete DTD is available as http://www.ecs.soton.ac.uk/~sr/ohs/ohpindex.html). XML also has the advantage of being a standard and it allows the re-use of a variety of existing tools and packages. Definition and naming of the dtd were a result of the move to consider OHP as a suite of protocols rather than a single protocol. Therefore, we can envisage future DTDs for other domains of hypertext, e.g. taxonomic hypertext (Parunak, 1989) or spatial hypertext (Marshall & Shipman, 1995).

In OHP-Nav hypertext objects are actually defined as XML element types which means that they exist as coherent objects. Therefore, passing whole objects becomes possible. Experience has shown that referring to objects via IDs results in a considerable amount of message traffic as each ID has to be resolved by another message. We can now avoid this by passing the entire object where necessary. OHP-Nav objects are held on the server side. Therefore, the server passes objects to the client while the client only needs to pass IDs to the server, which can resolve them directly.

The Definition of OHP-Nav in IDL

Besides the viewpoint that OHP is a suite of protocols rather than a single protocol it has also been envisaged that the navigational protocol can be 'spoken' over different communication mechanisms. With respect to this idea, we have also been working on an Interface Definition Language (IDL) definition which maps the existing OHP-Nav XML DTD to a CORBA compliant IDL (OMG). The IDL is called OHP-nav.idl and a simple implementation of the interface, as well as a sample server and client, can be downloaded from http://www.ecs.soton.ac.uk/~sr/ohs/ohpindex.html.

There are basically three different ways of implementing a mapping of the asynchronous ASCII based navigational protocol to a CORBA interface definition.

1. OHP-Nav messages passed as ASCII strings: ASCII messages could be parsed between communication objects that would support the navigational interface. The advantage of this approach is a precise mapping of messages between the XML definition and the IDL definition. Only the communication mechanism would be different. The disadvantages of this approach include the need for message parsing and the additional overhead for naming (see naming issue below). Additionally, we lose many of CORBA's benefits such as typed objects and regular flow of execution.
2. OHP-Nav messages as typed method calls on communication objects: This approach requires that OHP-Nav messages are mapped onto IDL method definitions. The advantage of this approach is that the IDL mapping is still very close to the XML specification and that at the same time we receive the benefits of CORBA such as typed objects and a high level of communication abstraction. The disadvantage is that dedicated communication objects are still needed for communicating messages. These objects have to be 'exported' and 'known' by all the participating components.
3. Pure CORBA implementation: This approach, which takes full advantage of CORBA, is characterised by defining all hypertext objects and their behavior as interfaces and by implementing a set of components that publish and support these interfaces. Therefore OHP-Nav clients, link servers, client side foos and server side foos (see below) will all support a different set of interfaces. For example, a server would support all hypertext object interfaces, the notification interface, etc. While a client would support only the notification interface.

In our definition of the OHP-Nav.idl we basically followed the second approach, i.e. we mapped OHP-Nav messages to IDL methods. However, in CORBA we can return objects from method calls. This means, for many of the methods, that we do not have to rely on a return message. For example, a 'createEndPoint()' call will return an endpoint object directly to the caller. This is different to the purely asynchronous way ASCII messages are being communicated over sockets. In this case the 'createEndPoint()' message would be sent from a client to a server and the server would answer it with an 'endPointDef()' message.

By using this technique we are somewhere in between the second and third approaches. This enables us to imitate the XML definition without sacrificing all of CORBA's benefits. Before we can follow the third approach entirely we, as a community, have to agree on further issues such as the naming of objects and the structure of the object model.

Findings and Issues

In completing the specifications and prototype implementations we encountered a number of issues that have since partially been resolved. These issues include the number of messages being sent (OHP-Nav Traffic), Routing, the CSF protocol, naming and architectural issues.

OHP - Nav Traffic

We found that using the last OHP specification (Davis et al., 1997) the number of messages being sent was disproportionately high in comparison with the task being carried out. This was because the protocol had been written to handle ID values (so as to mimic the 'pass by reference' structure of component frameworks such as CORBA). This means that ID values were continually having to be resolved and re-resolved. By using XML and by defining objects as element types we enable the link server to pass entire objects to the client while allowing the client to send object IDs to be directly resolved at the server side. Thus the overhead of sending messages for ID resolution is avoided.

Routing - The Return of the Channel?

Another issue we encountered was within the routing system of the CSF. We found that the CSF needed to record which applications sent which message IDs, so that when a message was answered the RID value of the return message could be used to find the original application. As OHP messages no longer carry any channel information (as defined in the original proposal (Davis et al., 1996)) we needed to 'hijack' the message ID on the way out so that it included the necessary information to identify it on its return as an RID. This avoided the problem of having to maintain possibly lengthy tables with time-outs for all the values.

Although this was successful we feel that the inclusion of an *Origin* field within the header of a message would make things simpler.

CSFP - Yet Another Protocol

Our major work with the CSF has been in the area of handling its complexity. The CSF as a nexus within the system is in an ideal position to perform a number of useful tasks. Configuring it to do these and adding any application information and options, requires a great deal of time. To cope with this we implemented a second, lightweight protocol to run alongside OHP-Nav. We called this the *CSF Protocol (CSFP)* and it deals with passing client side only information between client side software. For example a message for informing the CSF which applications deal with which mime types. The protocol also allowed us to build towards a more modular client system, using CSFP to communicate between components (see http://www.ecs.soton.ac.uk/~dem97r/csfprot.html for further details on the protocol definition). A dedicated protocol for inter-components communication called SIM Protocol (Server Information Manager Protocol) is proposed in (Nürnberg & Leggett, 1997).

Naming

As with any distributed system, using CORBA naming is an important issue and there has been little discussion about this in the OHSWG so far (Nürnberg & Legget, 1997). When using CORBA all the objects that support an interface are named. Thus a program that wishes to use an object that exists on the system can request it by using this name. Our current IDL specification uses the principle of typed method calls replicating each message (although we still utilise the ability to return an object). In this case the server must support an interface that defines all these method calls. Also the client must support an interface that defines all the return method calls (e.g. displayEndpoint).

To do this both client and server export a named object that supports the required interface. The client must know the name of the servers exported object and vice versa. This means that these names must be defined.

In the case of a pure CORBA implementation all the hypertext objects within a system have associated interfaces that allow direct manipulation of those objects. Each individual object in the system (e.g. an OHP-Nav node) must be exported as a CORBA object, and thus must be given a name. An important issue then, is exactly how this naming will work and if the object names will directly reflect the object's OHP-Nav id.

If this were to be the case, then the adoption of a global naming system would produce many benefits. In the same way that a URI can uniquely identify any 'document' on the WWW (Berners-Lee, 1994) an OHP unique identifier could identify uniquely any of the hypertext objects used, e.g. nodes, endpoints, pspecs. The identifier would be used as the OHP ID for the object and the CORBA name. Thus we would have the situation where an OHP unique ID within a global distributed hypertext system could be sent via email and still be valid.

We believe that any naming scheme would have to be built upon the URL concept. Although the complete definition of such a naming scheme is beyond the scope of this paper we envisage an OHPID similar to the following example:

Domain + Process + Context + Type + ObjectID

- The *Domain* Internet Domain Name of the machine, e.g. www.ecs.soton.ac.uk
- The *Process* identifies an individual process or system on that machine.
- The *Context* (or Linkbase) identifies which Linkbase the object resides on.
- The *Type* defines the type of the object, e.g. DR for a DataRef, EP for an endpoint.
- The *ObjectID* is unique to that set of objects within the linkbase.

An example of such an OHPID would be:

ohs.ecs.soton.ac.uk/4578/MCM34/EP/000567034

It is our belief that such a definition needs to be considered by the entire community with the aim of producing a complete definition.

Architectural Issues

Addressing interoperability not only requires some common understanding of the data model; also the underlying architecture has to be defined. This is because the data model usually makes assumptions about the architecture and because functionality and behavior are to a large extent based on the architecture. Promising work in the area of reference architectures already exists (Goose et al., 1997, Grønbæk & Wiil, 1997), however an assumed architecture grew alongside OHP-Nav and is shown in Figure 2.

From the diagram it is obvious that the architecture is not even. Why do we have a Client Side Foo (CSF) and not a Server Side Foo (SSF)? Many scenarios have shown the CSF communicating with a single linkserver (see Scenarios on the OHSWG web site). In this case there is no need for a nexus similar to the CSF on the server side, no routing is needed. However in the situation where several linkservers are all residing on a single machine or network then the role of an SSF becomes clearer, many of the functions it should perform are analogous to those on the client side. The concept of a server information manager (SIM) which collects server name and location information from servers and distributes it to clients and therefore is related to the idea of a Server Side Foo is presented in (Nürnberg & Leggett, 1997).

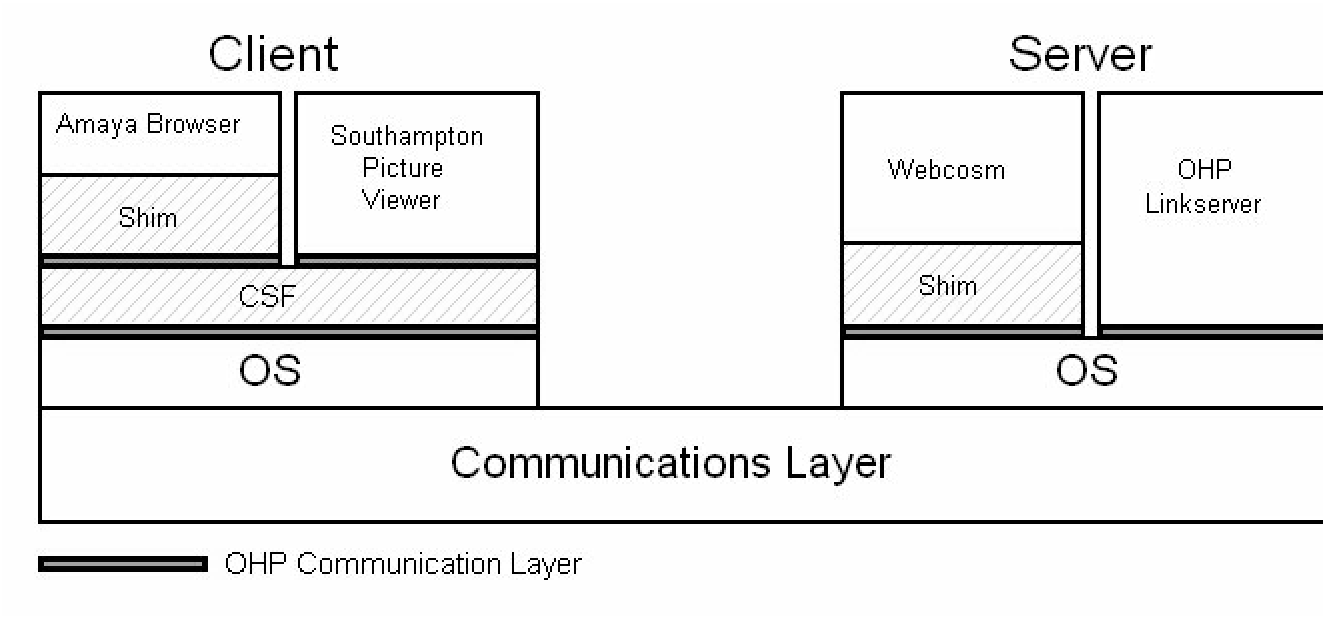

Fig 2. Current OHP-Nav Architecture

We believe that the following functionality is essential and will probably be realised by distinct subsystems within an SSF:

- Message Routing
- Support of (Multiple) Protocols
- Synchronization and Compilation of Results
- Transaction Handling
- Concurrency Control
- Notification Control

Additionally we think that this list could be optionally extended to include scripting engines, versioning control, configuration tools, and security and user components. It can be reasonably assumed that since a protocol / interface is required on the client side (i.e. the CSFP) to organise these modules, one will also be required on the server side (e.g. an SSF Protocol, SSFP).

Conclusion

Over the last few months we have undertaken work to prove the viability of the current OHP-Nav proposal, with the objective of showing interoperability between systems and to help eventually refine the specification. We have redefined OHP as OHP-Nav, a interface for navigating hypermedia objects. The specification is presented here both as an XML conforming document type definition and a CORBA conforming interface definition. We have shown the issues involved in matching these two very different specifications. We have also identified the need to extend the reference architecture to include a *Server Side Foo* that would in many ways mirror the activities of the CSF. We have also listed the functionality that could be expected from the SSF.

We believe that with the current specification of OHP-Nav in XML we are on the right road to interoperability. However, there are many more miles yet to travel.

Acknowledgments

The authors would like to thank the OHS community, all of whom have contributed to this paper, but special thanks are due to Pete Nürnberg, Ken Anderson and Uffe Wiil. We would also like to thank

Dave Chandler for implementing the parser and viewer and Jon Griffiths for implementing the navigational server. The work of Sigi Reich has been supported by the Austrian Fonds zur Förderung der wissenschaftlichen Forschung (FWF, grant No. J1507-INF).

References

- Berners-Lee, T. Universal resource identifiers in WWW. A unifying syntax for the expression of names and addresses of objects on the network as used in the World-Wide Web. Tech. rep., Internet RFC 1630, June 1994.
- Bray, T., Paoli, J., Sperberg-McQueen, C. M.: Extensible Markup Language (XML) 1.0, W3C Recommendation 10-February-1998, Available as http://www.w3.org/TR/1998/REC-xml-19980210.
- Davis, H. C. 2.5 Open Hypermedia System Working Group Meeting. (Southampton, England, December 1996). Available as http://www.ecs.soton.ac.uk/~hcd/research/invitation.htm.
- Davis, H., Lewis, A. & Rizk, A. OHP: A Draft Proposal for a Standard Open Hypermedia Protocol (Levels 0 and 1: Revision 1.2 - 13th March. 1996). 2nd Workshop on Open Hypermedia Systems, Washington, March 1996. Available as http://www.ecs.soton.ac.uk/~hcd/protweb.htm.
- Davis, H., Reich, S., Millard, D.: A proposal for a common navigational hypertext protocol. Presented at 3.5 Open Hypermedia System Working Group Meeting. Aarhus University, Denmark. September 8-11, 1997. Available as http://www.ecs.soton.ac.uk/~hcd/ohp/ohp35.htm.
- Goose, S., Lewis, A. & Davis, H. OHRA: Towards an Open Hypermedia Reference Architecture and a Migration Path for Existing Systems. *Journal of Digital Information (JoDI). Special Issue on Open Hypermedia 1*, 2 (1997).
- Grønbæk, K. & Wiil, U.K. Towards a Reference Architecture for Open Hypermedia. *Journal of Digital Information (JoDI). Special Issue on Open Hypermedia 1*, 2 (1997).
- Marshall, C. and Shipman, F. M.: Spatial hypertext designing for change. *Communications of the ACM 38*, 8 (Aug. 1995), 88—97.
- Nürnberg, P. J., and Leggett, J. J. A Vision for open hypermedia systems. *Journal of Digital Information (JoDI). Special Issue on Open Hypermedia Systems 1*, 2 (1997).
- Parunak, H. Van D.: Don't Link Me In: Set-Based Hypermedia for Taxonomic Reasoning. In Proceedings of Hypertext '91, San Antonio, Texas (1989), 233—242.
- Wiil, U. K. 3.5 Open Hypermedia System Working Group Meeting. (Aarhus, Denmark, September 1997). Available as http://www.daimi.aau.dk/~kock/OHS3.5/.