

Naming as a Fundamental Concept of Open Hypermedia Systems

Manolis Tzagarakis, Nikos Karousos, Dimitris Christodoulakis

Computer Technology Institute
Dept. of Computer Engineering & Informatics
University of Patras
26500 Rion, Greece
{tzagara, karousos, dxri}@cti.gr
Tel: +30 61 960381

Siegfried Reich

Dept. of Information
Systems
University of Linz
4040 Linz, Austria
sre@ifs.uni-linz.ac.at

ABSTRACT

Names play a key role in distributed hypertext systems, for two main reasons: Firstly, because accessing and managing system services require finding and locating the relevant components. Secondly, because managing structures between hypertext resources, such as nodes, anchors and links, requires that these resources are named and addressed.

We argue that naming services are endemic to hypertext systems and therefore, form a core part of any hypertext system's infrastructure. In particular, the current move towards interoperable component-based Open Hypermedia Systems (CB-OHS) demonstrates the need for naming components.

KEYWORDS: Naming System, Component-based Open Hypermedia System (CB-OHS), Reference Architecture.

INTRODUCTION

Research in infrastructures of hypermedia systems — particularly Open Hypermedia Systems (OHS) — focuses on the overall architecture of hypermedia systems, i.e., the identification of (common) components for various hypermedia application domains and interfaces between these components. For instance, the Open Hypermedia Systems Working Group (OHSWG) has promoted research towards interoperability amongst hypermedia systems and has come up with proposals for reference architectures and common middleware services for OHSs [11, 14, 31]. Figure 1 depicts the conceptual architecture of CB-OHSs. The three layers comprise of backend infrastructure, middleware services and front-end applications (from bottom to top). Furthermore, the opening of the middleware layer allows an open set of structure servers to co-exist within a single OHS [26].

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

Hypertext 2000, San Antonio, TX.

Copyright 2000 ACM 1-58113-227-1/00/0005...\$5.00

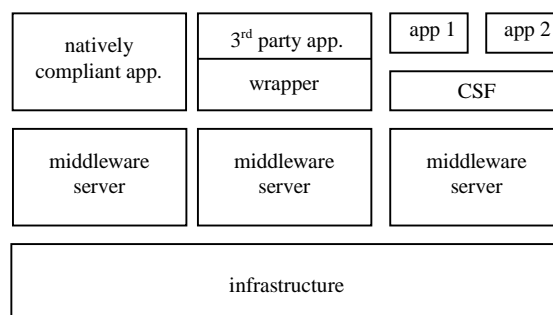


Figure 1: Conceptual Architecture of CB-OHSs [31]

In targeting these open and interoperable environments, the role of the rather invisible infrastructure of OHSs goes largely unnoticed. Yet, it is the quality of the infrastructure settings that determines to a great extent the level of interoperability and openness of OHSs [13].

In the aforementioned context, we aim to present the extent to which naming systems of OHSs can be exploited to reach the interoperability level that the OHS community is seeking. Fundamentally interoperability and openness in hypermedia systems are perceived as naming issues. Consequently, naming is considered as a technology enabling interoperability and openness. In particular, we present in detail how hypermedia resources, including hypermedia objects and services can be named, referenced and made accessible to clients and services in a standardized and seamless way. Our approach is based on the separation of operations for identifying and accessing resources.

Furthermore we analyze the information contained in a name and argue that naming is endemic to hypertext as it allows units of information to be structured and linked together. We present a conceptual architecture that utilizes the information contained in a name in order to move towards increasingly open and interoperable hypermedia systems.

The paper is structured as follows: Firstly, we investigate the role of naming in existing Open Hypermedia Systems. In the following section, we discuss objectives and requirements for naming from a conceptual point of view. We, then, introduce architectural guidelines, which we believe,

to open the way towards interoperable hypermedia systems. We finish with a summary and conclusions.

HYPERMEDIA SYSTEMS AND NAMING

All OHSs use some kind of name management. Most of them use identifiers and/or location information to refer to pieces of information in order to manage and traverse links. Yet, name management is almost never reflected *explicitly* in their infrastructure. In many cases, the management of names is either buried deep inside the application, or embedded in the hypertext data model. Naming schemes rely on simple assumptions (i.e., name is a unique sequence of symbols, is valid only for a specific host, structure server or application) in order to justify the reduced complexity of the name management system. These assumptions cause the systems to be of limited scope, thus creating distribution and interoperability problems.

Additionally, as already mentioned in the introduction, the recent move towards increasingly open, standardized and consequently interoperable components is a rather new phenomenon. Hence, many existing systems have not been built with support for exchangeable middleware or backend components in mind. Next, we will briefly report on the existing system's approaches to naming.

Most existing naming systems are functionally simple in that they rely on name / address pairs [5]. A component that wishes to communicate with or locate an object, queries a name service, which provides the object's address. This resolution process (from logical name to physical address often zooming in via various layers [16]) is referred to as "lookup". Some naming services are more elaborate in that they allow lookup of objects via attributes or characteristics. These are referred to as "descriptive naming systems" (e.g., yellow page services or X.500 [6]).

In hypertext, naming services are of particular importance because addressing is amongst the core functionalities of any hypermedia system. The World Wide Web's (WWW) hypertext functionality for instance, relies heavily on naming. Integrated in the Web's hypertext infrastructure are the Domain Name System (DNS, [19,20]), file names, and an access protocol. The resulting URLs (Uniform Resource Locators) have become an easy to use and thus popular means of exchanging information about a document's location and definitely contributed to the success of the Web as a hypertext system [12]. The basic naming mechanism allows referring to distributed resources and building associations (i.e., the links) between them. In addition, URLs have been used also to identify not only files but also a variety of other resources such as objects residing in databases or runtime entities such as sessions.

Obviously, we are all aware of the pitfalls of location based addressing: moved or renamed documents result in invalid names that lead to unaddressable resources ("404 document not found"). The Web community has therefore come up with workarounds such as the so-called Persistent URLs (PURLs, [36]). PURLs are basically URLs. However, instead of pointing directly to the location of a resource, they point to an intermediate lookup service, which resolves the PURL to the URL, e.g. via a standard HTTP redirect.

Other approaches include the NAPTR (Naming Authority Pointer) proposal [7], which implements a resolver discovery service (RDS), i.e. a service that allows Uniform Resource Names (URNs) to be resolved to URLs. It is implemented by a new resource record in the DNS (the NAPTR).

The Chimera Open Hypermedia System [1, 2] uses object IDs as names. Anchors for instance are identified and referenced using unique identifiers. Clients are responsible for mapping an anchor identifier into a region or object of its display. Only viewers and objects can be named, whereas middleware services cannot be located and exchanged. The integration with the Web is possible. In that case, the Web's naming infrastructure is used.

HyperDisco [37] provides the notion of a naming service. In order to address the issue of Internet-wide distribution, remote workspaces can be named by binding a name to a set of attributes. These names however are explicitly reflected within the data model. Links contain a number of endpoints and consist of a triple (workspace name, node ID, anchor ID). The field *workspace name* represents the remote workspace where the node ID and anchor ID are located.

Microcosm TNG [10] bases its addressing on the syntax of the URL. However, instead of a pathname after the machine name, a unique document identifier is appended. Furthermore, as each document is associated with a hypermedia application, this information also forms part of the name. The naming protocol is Microcosm specific but adoption of the URL syntax eases integration with existing Internet services.

Hypermedia System	Resource Access Info (RAI)		
	Medium required by client	Location information	Hypermedia Protocol
HyperDisco	TCP/IP socket API	Workspace name	Proprietary, encoded protocol
Chimera	Chimera language specific API, RPC	Host name	Proprietary, XDR based marshalling
DHM	TCP/IP socket API	Host name, port number	OHP ¹
WWW	TCP/IP sockets API	Host name, port number	HTTP ²
HOSS	HCMT ³ API	HOSS hyperbase name, object identifier	Proprietary, through HCMT
Microcosm	TCP/IP socket API	Host name, port number	Proprietary encoded

Table 1: RAI for Various OHSs

HOSS [26] handles naming by assigning objects a two-part

¹ Open Hypermedia Protocol.

² HyperText Transfer Protocol.

³ Hoss Communications Model Toolkit.

identifier (global and local IDs). The global identifier uniquely specifies a hyperbase. Hyperbases must be uniquely named (e.g., using a DNS-like system). The local ID is unique per hyperbase and is assigned by the hyperbase management system itself.

The HyTime standard [23] specifies a location address module that provides multiple ways of locating and addressing data including namespace locations, coordinate locations, and semantic locations. Furthermore, a querying facility is proposed to allow arbitrary lookups of data. We believe the main contribution of HyTime to be the very flexible way in which access to data is specified. However, issues such as scalability or mobility are outside the scope of the standard and therefore not addressed.

Table 1 gives a summarizing overview over existing (Open) Hypermedia Systems and describes the characteristics of the Resource Access Information (RAI) specific to each system. The RAI is defined as the environment in which a resource resides (see also below). It can be stated that the existing systems' naming infrastructures do not address interoperability sufficiently. In the remainder of the paper we will investigate characteristics of naming and come up with an architecture and prototype implementation that addresses these issues.

WHAT'S IN A NAME?

In principle, names are textual (i.e., human readable and understandable) linguistic entities that denote resources. A resource in turn, is anything to which an identity can be attached. Names therefore serve identification, access and mnemonic purposes [32].

Users use names mostly for mnemonic purposes. Therefore they tend to use descriptive names, which almost serve the purpose of summarizing a resource's contents and characteristics, expressing that way *what* the object is about. Additionally, different users will name the same resource differently, due to contrasting perceptions of the content.

On the contrary, *hypertext systems* use names for identification and access purposes. They rely on names in order to identify resources as well as to determine how to access them.

Depending on the purpose different types of information must be reflected in a name:

- **Identification purposes** [IDENT] obligate only a unique sequence of symbols (usually in the form of simple numbers) without meaning to humans. Such identifiers (IDs) are unique within the boundaries of the information system. They are generated, assigned and interpreted by the information system itself. For example <http://www.ht00.org:80/index.html> uniquely identifies a file using the Web's URL scheme.
- **Access purposes** [ACC] demand information about the environment in which the resource resides — the so-called *Resource Access Information (RAI)* — to be included. The RAI determines how to access a resource

and can be represented in a number of ways. These include *location information* of hosts such as their address, resources available on hosts such as files, access methods to be used including protocol and port, and many more. Access methods are provided through client side APIs. These APIs embody the medium that applications need to use in order to access a hypertext system. Often a name not only identifies a file but also designates the physical location of the file (such as URLs) as well as the access method. For example, examining the URL <http://www.ht00.org:80/index.html> the host expects clients to use the HTTP protocol to access the file and to direct their requests to the default port 80, where the web server process is listening.

- **Mnemonic** [MNEM] purposes require logical information to be reflected. That information is given to the resource by users and is interpreted by them. Looking at the URL <http://www.ifs.uni-linz.ac.at/ifs/staff/reich/reich.html> one can expect that the resource is the home page of a *staff member* at the *University of Linz*.

The different requirements of names suggest that the functionalities of names are orthogonal to each other (Figure 2).

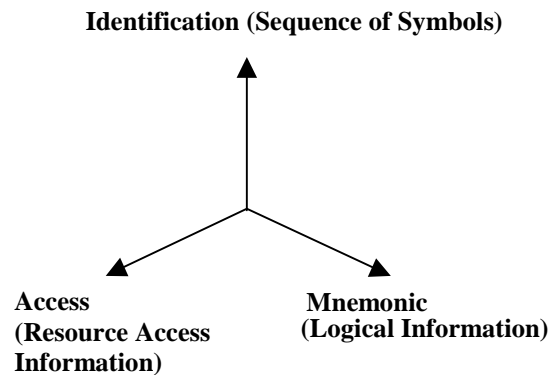


Figure 2: Orthogonal Functionalities of Names

A name management system or naming system provides the necessary structures, interfaces and services for handling the different functionalities of names. It bridges their different requirements. In order to do so the name management systems define *data contexts*. These are collections of name / resource bindings that are available for referencing within a systems name space.

Fulfilling all three purposes mentioned above, without taking into account their orthogonal functionalities, can have severe implications on the capabilities of the OHS. For instance, in cases where object IDs serve as names or in cases where only logical information is included in names, the capabilities OHSs can be limited. These systems do not scale well since no access information is present. On the other hand, putting only physical information (i.e. access information) in names makes the name dependent on the resource's RAI and decreases therefore its longevity. Some

systems attempt to condense the different functionalities into one name by encapsulating all the information in one string. The URL approach is an example of this category. To summarize, identification and proper management of the three different types of information are key to a suitable name management system.

Figure 3 depicts this relationship from a slightly different angle. The Figure expresses the relationship between logical names, physical names, access information and user context. A logical name such as “ImageOfJack” together with a context object, map to exactly one physical name (e.g., jack.jpg), which could have multiple access methods (e.g., multiple URLs).

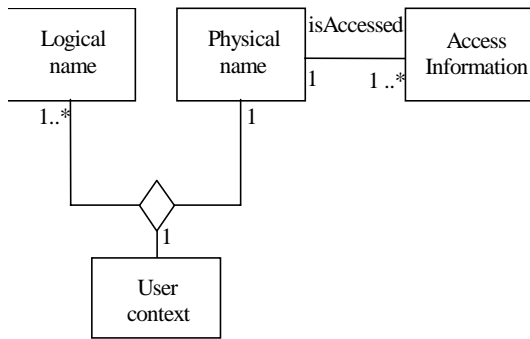


Figure 3: Relationship between Logical Names, Physical Names, Access Information and Context

Orthogonal functionalities of names can only be achieved by a naming system whose design encompasses certain characteristics. Such a set of characteristics is presented below.

Uniqueness (R1): names should be globally unique. Furthermore, the lifetime of a resource is potentially unlimited, so names have to be persistent [22, 32, 33]. *Nelson* for instance argues [22] that we should use tumblers to uniquely refer to any piece of information. He also proposes to refrain from deleting information so that all information units will always be accessible [IDENT].

Scope (R2): names should be globally valid, in order to allow named objects to be moved, their meaning should not depend on the location [33] (i.e., a resource’s name should not give any hint as to where the resource is physically located [34]) [IDENT].

Scalability (R3): there are different aspects to scalability such as number of accesses, number of resources, etc. With respect to the number of accesses, it can be said that a central broker dealing with name resolution will scale badly even when replicated; with respect to the number of resources it has to be said that resources might conceivably be available on the network forever (i.e., the number will be very high) [IDENT].

Readability (R4): names should be readable and interpretable by humans and they should be easily transcribable [3, 4, 9]. This mainly serves the mnemonic purposes [MNEM].

Multilinguality and Aliasing (R5): users will refer to the same object with different names and in different languages. This could be as simple as names for towns or countries in different languages; it will be more complex with varying cultures, etc. Aliasing allows users to create their own (personal) names for objects [MNEM].

Mobility (R6): resources might be subject to re-location at other places [15]. For instance, a set of links might be imported from a different linkbase (and naming clashes will have to be resolved) or also services might be mobile and have different locations over time. Naming should be location independent (i.e. resources should be relocatable without affecting their names) [ACC].

Legacy support (R7): names should accommodate existing name and identification schemes [29] [ACC].

Descriptiveness (R8): often users will not know the exact name of a resource so that it can be looked up. However, they will know some attributes such as author, type, size, etc. In this case it should be possible to provide the user with a mechanism to look up a name given a set of attributes [MNEM].

The sheer number of requirements indicates the complexity of naming. It should be mentioned that even though the list of requirements is rather appears to be comprehensive we have not mentioned other non-hypertext specific requirements. These include for instance trust and security [17]. Furthermore, some of these requirements are contradicting, for instance telephone numbers are globally unique but are often hard to remember. In the following sections, we will describe a naming system and its entities, which we believe to address these requirements. We will refer to the requirements by their numbers.

SPECIFYING THE RESOURCE ACCESS INFORMATION (RAI)

Providing facilities for establishing and maintaining connections to structure servers is important for client applications. The provision of such facilities by client applications is a presupposition for requesting any operation upon a resource. However, although several OHSs exist, clients integrated with one system cannot make use of the functionality of another system. This is largely due to the fact that different assumptions are made with regards to the environment in which hypermedia resources reside, leading to different requirements regarding the runtime environment of applications. Client applications assume different low-level communication mechanisms e.g. TCP/IP sockets, DDE, RPC, CORBA, DCOM, RMI or customized APIs, to name just a few (see Table 1).

Furthermore, access information is often buried deep inside front-end applications, thus being not configurable at runtime. The advent of CB-OHS makes this problem more pressing. Access information about middleware services should be handled and reconfigured in a flexible and dynamic way. Additionally, one should not expect all structure servers to be implemented with the same communication mechanisms. If radical openness and

interoperability are to be achieved, there should not be any low level restrictions. The perception of the RAI, which captures vital environmental information contained in OHS's components, enables the addressing of interoperability issues as presented above.

A NAMING ARCHITECTURE FOR OPEN HYPERMEDIA SYSTEMS

In the distributed systems' literature many different approaches for addressing heterogeneity can be distinguished (see e.g., [7, 21]). These approaches are based on (programming) languages, database systems or data representation on the application level. We argue in favor of a standardized data representation accompanied by architectural guidelines for implementing the necessary components, following the approach adopted by the OHSWG to address interoperability between front-end applications and navigational middleware [31].

Following these thoughts, it can be stated that all naming systems of OHSs include a similar set of abstractions in their naming models, namely identification, access and logical information. By defining a common set of interfaces and services that implement and manage these common abstractions, OHSs will become name-wise interoperable.

In the following paragraphs we will outline an architectural model for CB-OHSs in which a naming component is encapsulated as a fundamental building block within the systems' infrastructure. The proposed architecture originates from the development of Callimachus, a CB-OHS [35]. In order to address the characteristics mentioned above, we propose the separation of naming functionalities and the merging of naming as a core component into the underlying backend infrastructure. We propose an autonomous component within the infrastructure of OHSs providing the necessary functionality that ensures this separation of naming concerns by maintaining the appropriate information in its correct place:

- unique identifiers belong to objects themselves as part of their internal data;
- logical information is encapsulated in the names and
- Resource Access Information (RAI) is available in the resolution part of the name.

Such a name management system that maintains the database of bindings allows the establishment of names for resources, the reference of these resources by names, and the control of the resources' access methods by maintaining the *Hypermedia Resource Descriptor (HRD)* of a resource. Figure 4 shows how the conceptual architecture of CB-OHSs can be extended with Open Hypermedia Naming Services (OHNS) that communicate via a dedicated protocol with components at all layers. As will be shown herein below, an inter-OHNS protocol allows name services of different CB-OHSs to communicate thus providing the infrastructure for OHS interoperability and federation.

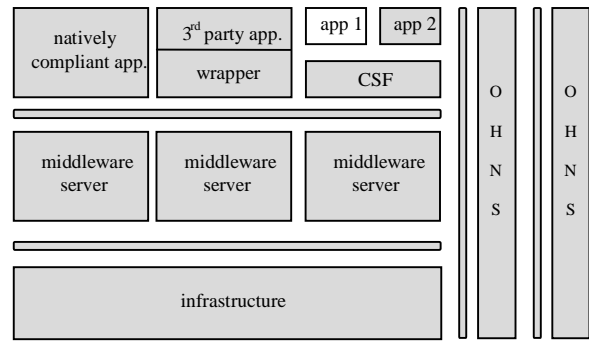


Figure 4: Naming Services are available to all Layers of CB-OHSs

The Hypermedia Resource Descriptor (HRD)

A hypermedia resource—an object or a component—is named if a name has been bound to a set of attributes of the specific hypermedia resource. These attributes describe the resources' identification as well as how the resource can be accessed within an OHS by abstracting OHS specific characteristics. We call the set of attributes to which a name is bound the *Hypermedia Resource Descriptor (HRD)*. An OHS's Name Service (OHNS⁴) maintains the independent database of bindings between names and HRDs. At runtime, applications request resolution of names from the name service using a name resolution protocol. As a reply to their requests they receive a set of HRDs.

The HRD allows resources to describe how they can be accessed by capturing information about the OHS environment. It provides the control structures that help components discover each other at runtime. Moreover, the provision of such meta-data allows applications to create access paths to resources by identifying the necessary communication medium. Thus, through the ubiquitous dissemination of HRDs, peer OHSs and applications may acquire the necessary information to interoperate. This separation of (physical) access information and (logical) names, by treating access details as first-class objects, is in contrast to the Web's approach of using URLs where access information is encoded within the name itself.

We believe that OHSs pose specific requirements to naming, such as inclusion of architectural, implementational and data model related information. Therefore DNS resource records or Uniform Resource Characteristics (URCs) are not sufficient to meet such requirements. Next we describe the structure of the hypermedia resource descriptor.

Anatomy of the Hypermedia Resource Descriptor (HRD)

Hypermedia structure elements and components require different types of attributes to express their characteristics. These attributes take character data as values that are predefined and have a precise semantic meaning. The application is responsible for interpreting these values in the

⁴ Hereafter referred to as *name service*.

correct way⁵. We identify two types⁶ of HRDs, those for structure elements and those for structure servers.

HRD for Structure Servers. This type of component is used to abstract information about the components — regardless of their implementation details — that are available within an OHS, providing structuring services. The following attributes describe this type of HRD:

- *Hypermedia application domain information:* a name to denote the hypermedia domain the component is servicing. This attribute can have values such as “navigation”, “taxonomic”, “spatial” etc. and allows components to be associated with a domain (see [24, 25] for an overview of hypertext application domains). We envisage the Open Hypermedia Systems Working Group (OHSWG) to agree on the most commonly used domains such that components can rely on an agreed set of domains.
- *Hypermedia protocol information:* the name of the protocol to be used in order to communicate with the component accompanied by a version identifier. Example field values could include “HTTP 1.0” or “OHP-Nav 1.3”.
- *Component framework information:* expresses the type of the component as well as the technology upon which this component is built. Components can be simple executables running as operation system services or daemons. Their implementation can be based on simple communication technologies such as TCP/IP and UDP or more advanced object busses such as CORBA, DCOM or RMI (cf. Requirement R7).
- *Component location information:* denotes the host on which the component resides and the port on which it is (eventually) listening for application requests.

HRD for Structure Elements. This type of Hypermedia Resource Descriptor (HRD) is used to describe how to access hypermedia structure elements such as nodes, links, or anchors. The following attributes describe this type of HRD:

- *Structure element identification:* the identification of the structure element as assigned by the data model of the OHS. For example, this field will have a simple ID, if the OHS that has created the object uses IDs as names, or a URL, if the data model uses the URL naming scheme (R1, R2).
- *Identification type:* the type of the element’s identification. This is important in order to determine how to use the HRD. Simple strings such as “URL”, “URN” or “PROPRIETARY” constitute the values of this attribute. The value “PROPRIETARY” is

used to identify a proprietary identification/naming scheme within an OHS, e.g. IDs serving as names or some other proprietary formalism (R7).

- *Access broker information:* The name of the component that has to be accessed in order to request an operation upon that hypermedia structure element. This component name has to be resolved further to obtain an access path to it.
- *Name server connection information:* connection information about the name server that maintains the HRD of this component.

In HRDs for hypermedia structure elements not all attributes have to be explicitly present. This depends on the *identification type* of the resource. For example, if URLs are used to identify a resource (the TYPE attribute has value “URL”), access broker information is obsolete since it is encoded within its identification (the URL itself). The TYPE attribute serves primary purposes of legacy support (R7). Figure 5 gives an example of a typical resource description specified in XML (Extensible Markup Language). A Hypermedia Resource Descriptor (HRD) of a structure element having ID “14” is shown. The “proprietary” value indicates that the OHS uses a proprietary naming/identification scheme. The particular resource is handled by a component named “Server1”. The component HRD is maintained by the specified name service.

OPEN HYPERMEDIA NAME SERVICES: A COLLECTION OF NAME MANAGEMENT SYSTEMS

Within an OHS the purpose of the name service is to maintain a database of bindings between names and HRDs (i.e., resources). It provides operations through which resources can be abstracted by HRDs, names can be bound to HRDs (and thus to resources) and HRDs can be looked up given their names. Names and HRDs have no meaning for the name service. Thus, they become an autonomous repository for hypermedia resources and can be exchanged similar to the way URLs are exchanged today.

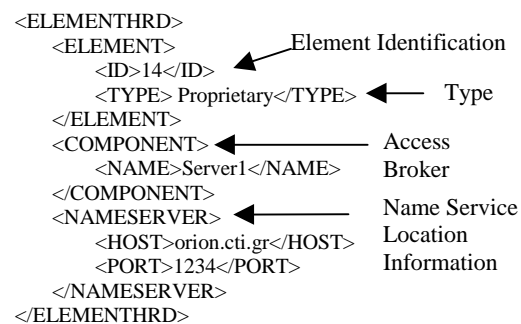


Figure 5: Example HRD for elements specified in XML

In the proposed architecture, there exists at least one name service in each OHS. A name can be bound to the connection information of an OHSs name service. In such cases we consider that the name identifies the OHS in which that name service operates. This binding is maintained by other OHSs and used in inter-name service communication. Such “chaining” of name services permits peer OHSs to communicate and HRDs of resources to be disseminated in

⁵ The name space of HRD metadata is not fixed. This has been chosen in order to ensure support for new OHS architectures and models.

⁶ We identify two types of HRDs since they differ substantially in their internal structure.

a pervasive way.

The name service of each OHS provides a set of operations through the use of a common protocol called OHNSP (Open Hypermedia Name Service Protocol).

OHNS Names

We call OHNS names those names that can be interpreted by the proposed name service. OHNS names encapsulate logical information and are structured according to a certain syntactic pattern which can be compared to the naming service in CORBA [28]. They consist of two parts: a handle and a remainder separated by a delimiter (e.g. “.”). The handle identifies the name service (and thus the OHS) in which the remainder has to be resolved. If a name has no remainder, it has to be resolved by the name service of the OHS in which the application is currently operating. For example the name A.B denotes that the remainder “B” has to be resolved further within an OHS whose name service is identified by the name “A”. This is to address requirement R3, scalability. Name resolution over multiple levels is possible.

Name Service Operations

Within any OHS, each name service provides the following operations:

- *Name binding*: To bind a name to the HRD of a resource. Users can name the structure elements they create, in any way they like (R4, R5). Yet, within each OHS these names have to be unique (R1).
- *Name aliasing*: To bind an alias to a name. Users may alias any name. This allows them to include logical information in a name that reflects their personal conception of the content. Aliases are also unique (R5).
- *Name resolution*: To return the HRD of a hypermedia resource given its name.
- *Name unbinding*: To delete a binding between a name and a HRD.
- *HRD update*: To update all or part of the attribute values of a HRD to which a name is bound.
- *Name querying*: to query the name service for the name that has bound to it an HRD with specific values in its attributes fields (R8).

Table 2 summarizes the operations of a name service. Whenever applications are requesting services from an OHS they maintain internally location information of the respective name service.

COMMON NAME RESOLUTION PROCEDURE

Names have to be resolved in order to identify the hypermedia resource they represent. Applications request resolution of names from a name service and get the HRD of that resource in return (see also Figure 6). When applications need to operate upon a hypermedia object, they send its name to the name service to which they are currently connected (see ① in Figure 6). The name is split

CreateBinding (name, HRD)	Binds <i>name</i> to the HRD of a structure element or resource
HRD = ResolveName (name)	Returns HRD for object identified by <i>name</i>
UpdateBinding (name, HRD)	Updates the HRD of a resource to which the name is bound
DeleteBinding (name, HRD)	Deletes a binding
CreateAlias (name, alias)	Creates an alias for a resource referenced by <i>name</i>
DeleteAlias (name, alias)	Deletes an alias
[HRD List] = GetComponent-ByHRD (HRD)	Returns a list of component HRDs that match the fields in the given HRD

Table 2: Operations of the Name Service

into the handle and the remainder. From the handle, a new context derives and the remainder is sent — through the interconnected name services — to the name service of the derived OHS to be resolved (② and ③). This operation cascades until the name cannot be split any further. At this point, the HRD of the resource is looked up (from the database of bindings of the name service) and the resource’s HRD is sent back to the application (④). In that way, the application obtains the required information to establish an access path to the resource. If no remainder is present, the name is resolved by the name service the application is currently connected to.

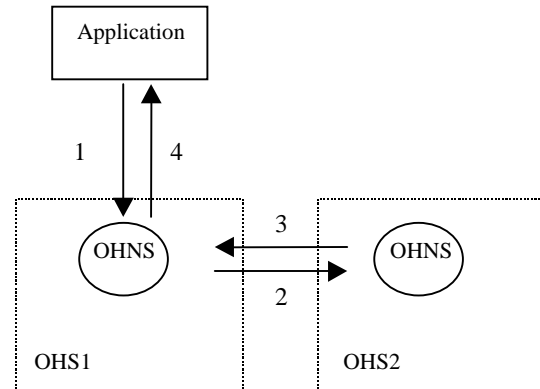


Figure 6: Name Resolution Procedure

SCENARIOS

We describe, herein under several scenarios clarifying the architecture described above. We assume that the OHS name servers are located on a specific host listening — on a specific port — for client requests and that applications have at any point in time a name service, which they are currently working with. Furthermore, components are autonomous entities located somewhere within a networked environment. Their configuration and initial data are

Name Service Operation	Comment
------------------------	---------

managed centrally only by authorized users. Applications contact components to obtain structuring services.

Creating Component Binding

Component binding is accomplished in a semi-automatic manner. Manual intervention by authorized users occurs only in case of name collisions.

1. When a new component is executed for the first time it registers its name / HRD binding. From a component-side initialization file applications obtain the required information: the values of their HRD, the name to which the HRD will be bound as well as location information (host and port) of the name service that will hold the name / HRD binding. Furthermore, the initialization file indicates if the component has been already named by the particular name service. If it is already named by the particular name service, the component signals only its presence to the name service, which in turn updates the proper attributes of its HRD.
2. In case the initialization file indicates that the component is not named by the name service, it issues a *CreateBinding* operation to the specified name service supplying the values of the initial HRD accompanied by their name. The name service records the name and the HRD in its database of bindings. After successful naming the component is notified. At this point, the component is *registered* within the OHS. On the contrary, if a component with the same name already exists, an exception is raised notifying the component that the naming procedure failed. During such name collisions authorized users supply a new name to the component by editing their initialization file.
3. When the name/HRD binding for a component is deleted from the database of bindings of a name service, the component is *unregistered* from that OHS. Its services are no longer available within the OHS.

Locating Components

1. During startup, the application obtains the location information of the initial name service from a client side initialization file. The application is aware of the naming service and implements the name protocol. The application is also aware of its hypertext application domain, e.g., navigation, the hypermedia protocol it is using as well as the component framework it is built upon.
2. The application connects to the name service and issues a *GetComponentByHRD* operation supplying a HRD that contains the criteria that have to be matched. These criteria denote the requirements of that particular application and are formed from the initial data.
3. The name server looks up its local database of bindings and retrieves the component names that match the criteria. Consequently, the name service further resolves the component names and returns to the applications the HRDs of the registered component along with their

names in that OHS. Applications record internally the returned data in order to re-use it in successive operations.

4. The application receives the list of HRDs and attempts to detect components that match its needs. If more than one component match the requirements, one is selected at random. More elaborate methods to select a specific component can be built if more information is contained within its HRD (e.g., availability). If no component matches the criteria, the application is *hypertext unaware* with regards to this OHS.

If no criteria are provided with the HRD, a name service returns a list of the HRDs of all available components within an OHS. In such cases, a list of component names accompanied by the domain that it supports is presented to the user who, in turn, selects the desired component to work with.

Locating hypermedia elements

In case of a link traversal, for instance, names of destination structure elements have to be resolved first. The resolution will reveal the access path to the resource. The lookup procedure can be described as follows:

1. The user opens an initial document. The names of the source anchors for this document are fetched from link service. The hypermedia elements participate in the structuring information with their names rather than with their identification.
2. When the user follows a link by clicking at a selection, the destination element names (e.g. nodes, anchors) are fetched from the link server.
3. If more than one destination elements are determined, the user is asked to select a specific destination. The name of the selected destination element is sent to the current name service in order to be resolved.
4. The name service first examines whether this name is an alias. In such cases it determines the real name, which is in turn resolved as described in the above paragraphs. The element's HRD together with the HRD of the component - handling that element - are then returned to the application.
5. The application uses the returned HRDs and requests from located component an operation upon the specified element e.g., by launching a browser window and displaying the element.

Multiple Name Services

Names can also be bound to remote name services. That can increase the name space of a given OHS. This means that more structure elements are available for referencing. Given the resolution procedure described previously, the HRD of the resource residing within the remote OHS can be transferred to applications, thus establishing an access path to it.

Relocating Resources

In case a resource is moved within an OHS, only updating the database of bindings where the resource's HRD resides is needed. Other structure elements that use the name to refer to it (e.g. documents) are not affected (R6). We have not yet addressed migrating resources between different name services. Simply forwarding requests might scale badly in particular for constantly moving resources such as resources residing on mobile devices [29, 30]. In those cases we can imagine an approach where the (mobile) component itself registers with a local name service that forwards the registration request to the components "home registry", similar to the way mobile phones work today.

In those cases that the identification type is "PROPRIETARY" (i.e., a custom naming scheme is used), all attributes of the HRD are utilized to access the resource. The *component location information* is used to locate the component and to request a component identified by its *structure element identification* using the *hypermedia protocol*. Furthermore, the *component framework attribute* is employed to determine if the application is "equipped" accurately. If the application is unable to recognize the *hypermedia protocol* or the *communication framework* an exception is raised, indicating that the application is hypertext unaware with regards to this OHS.

PROTOTYPE IMPLEMENTATION

A prototype of the proposed naming system has been implemented in the framework of the Callimachus CB-OHS [35]. In particular, two instances of the Callimachus CB-OHS allow multiple user communities to structure their data in an autonomous way. The naming system is employed to allow links to cross these instances in a transparent way. Since each instance belongs to a different administrative domain components can be relocated and exchanged at will by authorized users. The proposed naming system provides a way to address these issues.

Each name service has been implemented as a server process built upon the TCP/IP protocol. The database of bindings has been implemented on top of a relational DBMS. A proprietary application layer protocol using XML for encoding messages allows applications to request services from an name service.

SUMMARY AND FUTURE WORK

A new record type has been proposed — the Hypermedia Resource Descriptor (HRD) — which implements a separation of the three main naming functions. Furthermore, we have proposed architectural extensions to the existing reference models identifying name services as part of the infrastructure of OHS. In this paper, we have demonstrated the need for naming components, which will open the way for truly interoperable component-based open hypermedia systems. We have argued how an advanced naming system can provide the foundations upon which interoperable systems can be built. We have identified naming as an endemic property of hypertext and we have investigated characteristics of naming. The initial design and use of the name service within Callimachus gave us the opportunity to investigate important issues. These include:

- HRD attributes: The attributes included within a HRD are not sufficient for all frameworks. For example, these attributes are currently able to describe components that base their communication on plain TCP/IP sockets, as opposed to more advanced object buses such as CORBA, DCOM or RMI. TCP/IP sockets were selected because of their popularity.
- Discovery of components: Currently applications can query for components only in the OHS to which they are connected. The interconnection of the OHSs as well as the presence of HRDs will allow such queries to propagate to remote OHSs. This will enable the discovery of components that can support the application needs for a specific hypermedia domain.

In conclusion we would like to stress the importance of flexible naming for hypertext systems. The move towards increasingly open and interoperable OHSs requires that name management issues are reconsidered. A properly designed naming component encapsulated within the infrastructure of OHS will provide the adequate ground to move forward from traditional OHS to CB-OHS.

ACKNOWLEDGEMENTS

We wish to thank the anonymous referees for their comments, which greatly helped in improving the paper.

REFERENCES

1. ANDERSON, K. M. *Integrating open hypermedia systems with the world wide web*. In Proceedings of the '97 ACM Conference on Hypertext, April 6-11, 1997, Southampton, UK (Apr. 1997), pp. 157-166.
2. ANDERSON, K. M., TAYLOR, R. N., AND WHITEHEAD, E. J. *Chimera: Hypertext for heterogeneous software environments*. In ECHT '94. Proceedings of the ACM European conference on Hypermedia technology, Sept. 18-23, 1994, Edinburgh, Scotland, UK (1994), pp. 94-197.
3. 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*. Internet RFC 1630 (Informational), June 1994.
4. BERNERS-LEE, T. *The myth of names and addresses*. Tech. rep., World Wide Web Consortium, Dec. 1996. Available as <http://www.w3.org/DesignIssues/NameMyth.html>.
5. BOWMAN, M., DEBRAY, S. K., AND PETERSON, L. L. *Reasoning about naming systems*. ACM Transactions on Programming Languages and Systems 15, 5 (1993), 795-825.
6. CCITT INTERNATIONAL TELECOMMUNICATIONS UNION, Geneva, Switzerland. Recommendation X.500: *The Directory Overview of Concepts, Models and Service*, 1988.
7. DANIEL, R., AND MEALLING, M. *Resolution of Uniform Resource Identifiers Using the Domain Name System*. Internet RFC 2168 (Experimental), June 1997.
8. EMMERICH, W. *Engineering Distributed Objects*. Jon Wiley & Sons, Chichester, 2000.

9. ENGELBART, D. C. *Knowledge-domain interoperability and an open hyperdocument system*. In CSCW 90. Proceedings of the Conference on Computer-Supported Cooperative Work (Oct. 1990), pp. 143-156.
10. GOOSE, S. *A Framework for Distributed Open Hypermedia*. PhD thesis, Department of Electronics & Computer Science, University of Southampton, UK, 1997.
11. GOOSE, S., LEWIS, A. J., AND DAVIS, H. C. OHRA: *Towards an open hypermedia reference architecture and a migration path for existing systems*. Journal of Digital Information (JoDI). Special Issue on Open Hypermedia Systems 1, 2 (1997).
12. GRØNBÆK, K., AND TRIGG, R. *Toward a dexter-based model for open hypermedia: Unifying embedded references and link objects*. In Proceedings of the '96 ACM Conference on Hypertext, March 16-20, 1996, Washington, D.C. (1996), pp. 149-160.
13. GRØNBÆK, K., AND TRIGG, R. *From Web to Workplace: Designing Open Hypermedia Systems*, MIT Press, 1999.
14. GRØNBÆK, K., AND WIIL, U. K. *Towards a reference architecture for open hypermedia*. Journal of Digital Information (JoDI). Special Issue on Open Hypermedia Systems 1, 2 (1997).
15. GUENTHER, R. *Naming conventions for digital resources*. Tech. rep., Library of Congress. 1998. Available as <http://lcweb.loc.gov/marc/naming.html>.
16. KHARE, R. *Anatomy of a URL (and Other Internet-Scale Namespaces, Part 1)*. IEEE Internet, pp. 78-81, Sep. 1999.
17. KHARE, R. *What's in a Name? Trust. Internet-Scale Namespaces, Part II*. IEEE Internet, pp. 80-84, Nov. 1999.
18. MILLARD, D. E., REICH, S., AND DAVIS, H. C. *Reworking OHP: the road to OHP-Nav*. In Proceedings of the 4th Workshop on Open Hypermedia Systems, ACM Hypertext '98 Conference, Pittsburgh, PA, June 20-24 (June 1998), U. K. Wiil, Ed., pp. 48-53.
19. MOCKAPETRIS, P. *Domain names — concepts and facilities*. Internet RFC 1034 (Obsoleted), Nov. 1987.
20. MOCKAPETRIS, P. *Domain names — implementation & specification*. Internet RFC 1035 (Obsoleted), Nov. 1987.
21. MÜHLHÄUSER, M., AND SCHILL, A. *Software Engineering für verteilte Anwendungen*. Springer, Berlin/Heidelberg, New York, 1992.
22. NELSON, T. H. *Managing immense storage*. Byte (Jan. 1988), 225-238.
23. NEWCOMB, S. R., KIPP, N. A., AND NEWCOMB, V. T. *The HyTime hypermedia/time-based document structuring language*. Communications of the ACM 34, 11 (1991), 67-83.
24. NÜRNBERG, P. J. *HOSS: An Environment to Support Structural Computing*. PhD thesis, Department of Computer Science, Texas A&M University, College Station, TX, 1997.
25. NÜRNBERG, P. J., GRØNBÆK, K., BUCKA-LASSEN, D., PEDERSEN, C. A., AND REINERT, O. *A component-based open hypermedia approach to integrating structure services*. New Review of Hypermedia and Multimedia (1999). Accepted for publication.
26. 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).
27. NÜRNBERG, P. J., LEGGETT, J. J., AND WIIL, U. K. *An agenda for open hypermedia research*. In Proceedings of the '98 ACM Conference on Hypertext, June 20-24, 1998, Pittsburgh, PA (1998), pp. 198-206.
28. OMG — OBJECT MANAGEMENT GROUP. CORBA services, Chapter 3, *Naming Service Specification*. OMG Document No. 97-07-12, 1997.
29. PASKIN, N. *Toward Unique Identifiers*. Proceedings of the IEEE 87, 7 (1999).
30. PERKINS, C. *IP mobility support*. Internet RFC 2002 (Standards Track), Oct. 1996.
31. REICH, S., WIIL, U. K., NÜRNBERG, P. J., DAVIS, H. C., GRØNBÆK, K., ANDERSON, K. M., MILLARD, D. E., AND HAAKE, J. M. *Addressing interoperability in open hypermedia: The design of the open hypermedia protocol*. New Review of Hypermedia and Multimedia (1999).
32. SOLLINS, K., R., VAN DYKE, J., R., *Linking in a Global Information Architecture*. In Proceedings of the fourth International World Wide Web Conference. Boston, MA (1995).
33. SOLLINS, K., AND MASINTER, L. *Functional requirements for uniform resource names*. Internet RFC 1737 (Informational), Dec. 1994.
34. TANENBAUM, A. S. *Distributed Operating Systems*. Prentice-Hall International, Englewood Cliffs, New Jersey, 1995.
35. TZAGARAKIS, M., VAITIS, M., PAPADOPOULOS, A., AND CHRISTODOULAKIS, D. *The Callimachus approach to distributed hypermedia*. In Proceedings of the '99 ACM Conference on Hypertext, February 21-25, 1999, Darmstadt, Germany (Feb. 1999), pp. 47-48.
36. WEIBEL, S., JUL, E. AND SCHAFFER, K. *PURLs: Persistent Uniform Resource Locators*. OCLC Newsletter, November/December 1995. Available as <http://purl.oclc.org/OCLC/PURL/SUMMARY>.
37. WIIL, U. K., AND LEGGETT, J. J. *The HyperDisco approach to open hypermedia systems*. In Proceedings of the '96 ACM Conference on Hypertext, March 16-20, 1996, Washington, D.C. (1996), pp. 140-148.