Supporting Longevity in an Information Infrastructure Architecture

Karen R. Sollins
MIT Laboratory for Computer Science
545 Technology Square
Cambridge, MA 02139 USA
sollins@lcs.mit.edu

1 Introduction

With the growth in the Internet and network-based community, comes an interest in building applications independently of the information or data on which they will operate. In conjunction with this divergence of information from application and significant growth in the amount of information available on the net the issues of longevity are taking a more prominent position. As investment in information grows, the amount of information that has long-term value also grows. The URLs (Uniform Resource Locators) used as identifiers and access methods for Web documents will not, and in fact even now do not, suffice. Tying identification to location and access protocol leads to significant problems as information moves, underlying storage facilities are reorganized, or as different access protocols are needed. Furthermore, we will experience evolution many dimensions. Applications and the sorts of information on which they operate will evolve. Existing information is likely to evolve. The underlying transport mechanisms are continually evolving to meet higher or new demands. The question we as researchers must ask ourselves is how do we design a system that is prepared for the problems of longevity, with the long-term future in mind, rather than working only to solve the problems of the present, as we have so often done.

My current research project is the *Information Mesh*, the design of an information infrastructure architecture which has among its primary goals the support of longevity, mobility, and evolvability. In this note, I will briefly summarize the goals and highlights of the architecture of the Information Mesh. The note will conclude with a description of some of the significant ways in which it addresses mobility and evolvability. It should be noted that this work is described in more detail in [11, 9]. In this note I will only refer briefly to differences between the Information Mesh and the World Wide Web, CORBA and OLE.

2 Information Mesh

The focus of this work is the provision of the commonality needed for the growing and evolving set of global network based applications. In designing a common infrastructure, one must consider what is in it and where commonality is not valuable or feasible. Much previous work has focused on program communications or invocation models. There are those who believe in strict RPC, or various extensions of RPC providing, for example, ordering among multiple outstanding calls. Others believe in various group multicast communications paradigms. Yet others believe in message passing models. Each paradigm has its strengths and weaknesses and is probably "the right" solution to some set of problems.

Similarly, in addressing the problems of a stable infrastructure, many would suggest that persistent storage is a necessary component. Here again we see a variety of views on service required.

For some the persistent storage of a file system is adequate. In this case, information or objects will be transformed into a file format for storage, to be reconverted as the contents are needed. There are a variety of models of distributed or network based file systems, with no agreement on which semantics for distributed files is "the right" one. Others believe in various persistent object store models, with or without "pickled" forms of objects in storage and with or without various forms of transaction update mechanisms. Again, there is no common agreement on "the right" model, but rather each has its strengths and weaknesses and should be usable in appropriate situations.

Further understanding reveals a need for a uniform information infrastructure model on top of which to build either applications or programming environments with their own models of stability and permanence. We must recognize in providing a single, common information infrastructure substrate that at least some of the information will both be long-lived and of interest perhaps beyond the environment and applications within which it was created. It is useful to enable but not restrict the models needed for differing environments. We do not, in contrast, have a requirement for a single computational model. No one computational model appears at present to be the right one for all situations and problems. Nor will a single persistent storage model suffice. In addition, in considering security, accounting, and billing, etc. it is clear from the recognition that these network based activities will be occurring across administrative boundaries as well as programming model boundaries, that no single policy model, or even set of mechanisms to support policies will suffice. The work of this project is to provide a simple, extensible information infrastructure model supported below the level of any applications, which will allow objects to survive and new ones to evolve unconstrained by the infrastructure, yet supported by it.

2.1 Goals

The primary motivation of this project to provide an information infrastructure that will survive for a period measured in units of human lifetimes, significantly longer than any such architecture to date. In considering such a time frame, several issues must be addressed that often are not issues in solving immediate problems: mobility and evolution in the face of longevity. Over any significant period of time, information will move both physically, and, perhaps, administratively. For example, not only may books be moved from one shelf to another or perhaps from one building to another, but an individual may leave a book collection as a bequest to a library, or two libraries may merge, forming a new legal and administrative entity to manage the merged collection. In addition, over a long time period everything imaginable may evolve. First, the supporting mechanisms of transport and presentation protocols may evolve, either replacing or enhancing what was previously available. Second, the programming and applications models of information may evolve, requiring information using new abstraction paradigms. Third, the information itself may evolve.

Thus we identify a set of goals that includes those described above, as well as other more commonly found in others infrastructure models.

- **Longevity**: The infrastructure and support of relationships among pieces of information must be able to survive for periods measured in multiple human lifetimes, possibly longer than the entities themselves.
- **Mobility**: Information must be able to move both physically and administratively among organizations.

- **Evolvability**: Information itself as well as the clients of it, often programs, in addition to the infrastructure for accessing it is likely to change with time. The infrastructure must support the ability to take advantage of such evolutionary changes.
- **Ubiquity**: The infrastructure must be global. Information and clients should be able to be anywhere and to move anywhere, both physically and administratively, restricted only by policies limiting such mobility.
- **Modularity and Abstraction**: The model of the infrastructure should support the separation of specification from implementations. This allows for heterogeneity in both the clients and the potential infrastructure on which they may be built. This in conjunction with the ability to learn about the abstraction supported permits applications or clients to use information created and managed independently of the applications or clients themselves.
- **Homogeneous abstract model**: The Information Mesh should provide a self-consistent abstract model of pieces of information and their relationships. A homogeneous abstract model allows the supporting infrastructure such as transport protocols to evolve and be used without the applications and clients necessarily needing to know about such a transition.
- **Resiliency**: Networks and network resources fail unilaterally and unpredictably. The Information Mesh will be prone to this as well and hence must provide reasonable resiliency to such failures, if it is to be useful to potential users.
- **Simplicity**: Only a simple model, not overburdened with unnecessary mechanism will survive. Simplicity is critical to both correctness and utility.
- **Support of Relationships**: The Information Mesh must support the construction of relationships among pieces of information, that will allow for the expression of the nature of a relationship, the ability to link into an object without a dependency on the internal representation of that object, the ability to link to a link, and a number of other criteria spelled out in more detail in [11].

The focus on longevity, mobility and evolution distinguish this work from other similar efforts.

2.2 Object model

The Information Mesh model is an object model, in that all components of within the Mesh are objects. To be a Mesh object, a resource must have at least one *oid* or Object Identifier and support at least one *role*. Oids with the attendant *hint* mechanism provide identification and location, while roles provide our typing model.

Historically, naming has often encompassed three basic functions, identification, access and semantic or mnemonic information. Identification is the task of distinguishing one resource from another. The accessing functionality allows us to find, get to, and perhaps use or modify an object. Finally, some naming schemes, although not all, provide human friendly semantics or at least mnemonics. URLs, as used in the World Wide Web do all three; they are used as identifiers embedded in links in long-lived objects, they specify the location and protocol for accessing the object, and they often have user-friendly semantics embedded in them. The problem with this is

that when the semantics, location or access protocol changes the URL no longer reflects what the users have come to expect from it.

We have chosen in the Information Mesh to separate these functions and not support human-friendly semantics within the Mesh. Oids are globally unique, long-lived identifiers intended not for human but for computer use. They are part of the infrastructure, on top of which applications and application domains may reside and a variety of human-friendly naming schemes can be built. It is in the context of the application domains that we expect human friendly naming schemes to be supported. In addition, location discovery is based on potentially mutable and evolving hints. When an object is to be found a set of hints will be used to suggest locations or location translation services that may know of locations. It is recommended that hints accompany the transmission of an oid. In addition, we expect there to be hint servers as fallbacks for discovering hints. When an object moves, in the simplest case this will be reported to a resolution service, rather than needing to broadcast the fact widely. If a resolution server no longer serves a particular oid, it may return to the client alternative hints of other location servers to try. Thus the set of hints can evolve and track mobility. Uniform Resource Names or URNs as defined in the Internet Engineering Task Force by Sollins and Masinter[10] reflect much of our thinking. Work is progressing both at MIT and in the IETF on URN resolution and hints. (See the following unpublished documents for the current status[3, 4, 5].)

The second major aspect of the Mesh object model is the typing model, based on roles. We call them roles because as with human playing roles, an object can play more than one at any time. In addition, the set of roles an object plays may evolve with time. Roles are defined in a hierarchy, supporting multiple inheritance, with a single root in the **object** role. There are three aspects to role definitions, a set of *actions*, a set of *parts*, and a set of *makers*. In a role definition elements of each set may be either required or optional. There may be multiple implementations of any particular role. The set of actions of a role defines the role's abstract functionality. The set of parts defines an abstract structure for an object playing the role. And the set of makers defines the abstract ability to create new objects playing the role in question. A role definition is abstract in that it does not determine the implementation or representation that will be used. The intention here is to gain the advantages of modularity by use of abstraction.

Roles reflect ideas from a number of different sources. Functional abstraction as reflected in our *actions* is provided by information models such as CORBA (called *methods* in CORBA *interfaces*) and languages such as Clu (called *procedures*) and C++. The term *method* has a different meaning in Lisp languages such as CLOS where it implies the implementation of a generic procedure. We have therefore chosen the new term *action*. *Parts* reflect an abstraction from the class based languages such as CLOS that specify the structure of an object. *Makers* come directly from Liskov's Theta language[6].

In comparison, different choices have been made in the World Wide Web[1], CORBA[8] and OLE[7] (and in particular COM which is the underlying object support in OLE). With respect to naming the Web uses URLs that merge the three functions of identification, access, and semantics, complicating issues of mobility and evolution. CORBA supports ORBs (object request brokers). An Orb can be distributed and supports unique names for all objects within the ORB. To name an object in another ORB, a new name is composed of the ORB's name and the name within the ORB. This has the advantage that naming can evolve completely independently within an ORB and the namespaces can be federated. It has the disadvantage that if an object wants to move across

ORB boundaries it will need a new name, negating the utility of the old name for it embedded in existing, perhaps immutable, objects, unless forwarding pointers are maintained by the original ORB forever. Objects can only easily move within an ORB. OLE at present simply does not support distributed naming. The Web model of typing is based on some combination of semantics in the URL, the protocol in use, and heuristics based on the content of the object. There is a proposal to support string labels to identify "types" with no particular model of management of the meaning of the strings. CORBA supports a typing model very similar to ours, although abstract structure is given much less importance and in the reference documents its use is discouraged, while OLE does not support inheritance, believing that the problems of versioning in a type inheritance scheme outweigh any advantages.

3 Several Issues in Realizing the Information Mesh

There are several specific issues that have arisen in providing a substrate such as the Information Mesh: oid or name resolution, links, and security. Each is addressed here briefly.

3.1 Name resolution

As suggested above, with the separation of oids or names from location information, there is a need for name resolution. Furthermore, given that objects will move and storage server may not always be available, we recognize that the location information must be both mutable and as resilient to failure as possible. Therefore location information is embodied in hints, that suggest locations which may be able to further the process of locating an object. A hint may be an address at which the object was last seen or several such addresses if it is cached or replicated. A hint may also help in finding a resolution service that may know of the location of the object. There may be both authoritative and non-authoritative resolution services. This level of indirection allows for updates of location information to be made in only one or a small number of places, but to be found by many clients. In all cases, it is possible that the response to a query will be to return to the client more hints. As a further backup mechanism, if all known hints fail, there will be a more authoritative, more complex to use, global mechanism for discovering hints. This fallback mechanism is under discussion and design both in our project and in the IETF[3, 4, 5]. In general, hints will be exchanged when references to objects are exchanged; for example, if one person recommends an object to another, not only will the oid or URN be passed but also a set of hints for finding the object should also be transferred. These can be locally cached; if a search for the object occurs, the set of hints may be modified to reflect the experience. Hence a modified set of hints may be passed along to someone else.

3.2 Links

Since links must also meet our objectives as set out in Section 2.1, the object model of the Information Mesh provides a clean, consistent basis for links; links are first class objects in the Information Mesh, and all links play the *link* superrole, and possibly subroles of it. Because links are first class objects, we can refer to a link by its oid and define the nature of its relationship by the role it plays.

Perhaps the most important feature of the role model with respect to links is the ability to define an abstract structure. The part mechanism allows us to define the abstract components of an object, that it must guarantee to support in any implementation and representation. Thus a link can refer to a part of an object and be guaranteed that that reference will remain meaningful through transitions to alternative implementations. It should be noted that this does not imply immutability in all objects to which links are created. Links may fail for several reasons. The object itself may go away or the component may go away. A book may simply be withdrawn, or its chapters may be reorganized to reduce the number of chapters. If a link has been created to "chapter 7" of the book and the book no longer has seven chapters, the link is no longer valid. On the other hand, if the chapters were originally represented by pointers into a large buffer of characters and later are represented by a set of buffers in a list, we do not need to know that, but can continue to refer to the chapters as chapters.

The part model also allows a link to contain parts, easily distinguishing the different endpoints of a link. Thus, for example, a multi-part link such as the relationship involved in purchasing a car may involve the buyer, the seller, and the loan organization helping the buyer pay for the car. Links that are first class objects with names, roles, and the ability to link into the abstract structure of the endpoints provide a richer and much more long-lived infrastructure than a simple hypertext model such as the HTML model of the World Wide Web. We have published more detail on this issue[11].

3.3 Security

Security as mentioned earlier represents a dilemma. On the one hand, there can be no one policy or set of mechanisms that will suffice. Simultaneously, if an infrastructure into which access control can be fit is not provided, a model such as ours will not succeed because security will be modelled completely differently in each locality. In addition, the encapsulation and abstraction paradigm of the Information Mesh has the problem that it leads to a model in which each object is provides a security perimeter of its own. This is contrary to the current trend of providing firewalls to guarantee a consistent and managed security perimeter around an organization or community.

Again, the object model of the Information Mesh provides us with a solution. If each security domain is considered to be an object, it can support policies using the mechanisms it chooses. Other security domains can be nested within it or cooperate if an object is in two overlapping security domains. The object model provides for an abstract interaction among security domains. The particular mechanisms used by a specific domain can evolve with time by providing new implementations, while supporting an abstract interface to the outside world. This work was explored in the context of our project by Condell[2].

4 Conclusion: Resiliency to Mobility and Evolution

We conclude by highlighting how it is that the Information Mesh addresses the problems of mobility and evolution. If we do not support mobility and evolution, we will discover significant problems down the road, as we make a growing investment in information that becomes unfindable or inaccessible.

The separation of identification from discovery of location allows for much greater flexibility with respect to mobility. An object must report a new location to some service in order for it to be found there, but does not need to report that to every site that has a reference to it. The evolvable hint mechanism allows for that. Thus, for example, links remain valid, even if their endpoints move.

With respect to evolution, the role model allows objects to evolve, implementations to evolve, and the application space to evolve to use new sorts of objects. By requiring that every role definition include the action of **get-roles**, an object will always be able to report which roles it plays. Since all objects must play the **object** role at least the question can be asked in that context. If the representation of an object changes, since links will be able to link only to abstract parts of an object, links will remain valid.

In conclusion, we have built a prototype system, but are now exporting the ideas to the Web by building services to provide these functions in the Web context.

References

[1] T. Berners-Lee et al., *The world wide web*, **Communications of the ACM**, 37(8):76-82, August, 1994.

[2] M. Condell, **A Security Model for the Information Mesh**, MIT/LCS-TR-691, June 1996. Also thesis for the Master's of Engineering degree.

[3] L. Daigle, P. Faltstrom, R. Iannella, **A Framework for the Assignment and Resolution of Uniform Resource Names**, Internet Draft draft-daigle-urnframework-00.txt, June, 1996.

[4] R. Daniel and M. Mealling, **Resolution of Uniform Resource Identifiers using the Domain Name System**, Internet Draft draft-daniel-naptr-00.txt, June, 1996.

[5] L. Girod and K. Sollins, **Requirements for URN Resolution Systems**, Internet Draft draft-girod-urn-res-require-00.txt, June, 1996.

[6] B. Liskov et al., **Theta Reference Manual**, Programming Methodology Group Memo 88, February, 1995.

[7] Microsoft OLE 2.0 Design Team, *Microsoft OLE 2.0 Design Specification*, **Microsoft Development Library**, April, 1993.

[8] Object Management Group, **CORBA: Architecture and Specification**, The Object Management Group, August, 1995.

[9] K.R Sollins, **Extending the Network Model to an Information Infrastructure: The Information Mesh**, unpublished, available from <http://ana-www.lcs.mit.edu/anaweb/ps-papers/sollins-extend.ps>, January, 1996.

[10] K.R. Sollins and L. Masinter, *Requirements for Uniform Resource Names*, Network Working Group RFC 1737, February, 1995.

[11] K.R. Sollins and J. R. Van Dyke, *Linking in a Global Information Infrastructure*, **Proc. Fourth World Wide Web Conference**, Boston, MA, December, 1995.