In Proc. Grace Hopper Celebration of Women in Computing Conference, September, 1997

At the Conjunction of Networking and Distributed Systems

Karen R. Sollins
MIT Laboratory for Computer Science
sollins@lcs.mit.edu

Abstract

This paper focusses on two central ideas. The first is that it is valuable and important to extend our model of the network to a higher level of abstraction and generality than has been provided. traditionally The second idea is that the level of abstraction to which the network should be extended should provide an extremely simple yet powerful object model, similar to, but not the same as, those of several existing distributed system substrates. The goals of such an infrastructure should be not only those commonly used such as ubiquity and application heterogeneity, but longevity, mobility, and evolvability. We describe the Information Mesh architecture and address some of the issues in realizing it.

1. Introduction

In their simplest form, networks enable the communication among applications or programs running on disparate computers. Each layer of network protocol provides another level of abstraction in terms of communication. For example, TCP provides reliable bit streams, and FTP, the exchange of files. In contrast, distributed systems such as Emerald [JLHB88], ANSA [VDL93], OLE [Mic93], and CORBA [OMG95] support rich object and communications models. The World Wide Web [B-L94a] provides a middle ground, in that each document is self-describing in HTML [BC95]. It supports linking as in typical hypertext systems [HS94]. There are two extremely important points to be drawn from the WWW: 1) its simplicity, and 2) its acceptance of inconsistencies in its infrastructure.

Briefly, let us consider an example. Our programmer wishes to write a conferencing tool system. Three features are worth nothing here. First, the system supports private two-way, as well as public multi-party communications. Second, the shared "white board" should be able to support composite active elements that communicate with each other. Third, our programmer can imagine that with the right equipment, the "white board" might become 3-D holograms, but when that is not available, it might be simply 2 dimensional projections on a screen. There are a number of points to be taken from this example:

- The objects to be shared in the white board may have been created independently, without having been planned to be presented in the white board. Their types may not even be known to the white board component prior to the attempt to share them in the white board.
- The location of the elements of the white board as well as the participants may be both widely dispersed and varying with time.
- There are some features, such as communications paradigms, that cannot be proscribed. No one solution will suffice.

The Information Mesh is an extensible, simply architected model of globally accessible, long-lived resources with resiliency to failure as a key aspect. As such, it is intended to provide an object based model of the network on top of which a wide variety of enhanced features, tools, and components might be provided. The remainder of this document proceeds as follows. We will begin with a enumeration of the goals of the project, in more detail. We will then describe the model, including both the naming and typing components. Our approach to linking can then be described. The succeeding sections identify a collection of core services required to provide such a facility and conclude with further related work and a summary of the contributions.

2. Goals of the Architecture

As we have described above, our objective is the design of an information infrastructure. The problem is to allow for utilization in the face of the heterogeneity and failure that is inherent in the network. We also must recognize the exponential growth in resources in the Internet,

----

This work is supported under DARPA contract no. DABT63-94-C-0072. Additionally, the Intel Corporation has provided a gift in support of this work.

with its concomitant growth in investment, both directly in those resources as well as programs to run in this new large scale, long-lived environment. In this context we identify seven major goals:

- **Longevity**: Both information and identifiers for the information must be able to survive for lifespans measured in human terms.
- **Mobility**: Information must be able to move not only among physical, but also administrative locations.
- **Evolvability**: Not only will resources themselves need to evolve to support new functionality, but the users for them may evolve with time.
- **Ubiquity**: It is necessary to support access to information distributed both physically around the world and administratively across regions of differing management policies.
- **Resiliency**: The infrastructure must be prepared for failures and unexpected behavior.
- **Heterogeneity**: The Mesh must be prepared to encompass new network services as they evolve as well as easily supporting a broad set of expectations from applications in addition to administrative controls.
- **Minimality**: In order to succeed, the Information Mesh must place a minimum of requirements and restrictions on its users. We must understand what is required of it to achieve the other goals identified here and provide no more than that minimum.

Of these goals, longevity, mobility, and evolvability are the three that distinguish this work from others. With these in mind, we can explore the architecture and then discuss the differences between this work and other apparently similar efforts.

3. The Architecture

The Information Mesh model is one that extends the model of the network upwards to provide an extremely simple, powerful, and expressive network-based entity model. It specifies resources that are named and typed, and among which relationships can be represented. There are significant components that might be considered part of a distributed system that are not provided here, because there appears to be no single solution that will suffice, in particular in the areas of communications, security, and transport.

The question of how programs communicate with each other is traditionally a significant part of any distributed system. SunRPC [Sun88] and DCE [Joh91] support various forms of Remote Procedure Call. ISIS[BSS91], for example provides causal invocation and group communications. Message passing is more common in strict object oriented systems, such as ANSA[Ree93,VDL93] or CLOS [Kee88]. Each one provides a distinct and valuable communications paradigm and no one is right for all situations. Hence we have chosen not to make a single choice in this regard.

The same sort of problem exists with respect to security. No one policy is suitable for all situations. Furthermore, no one mechanism is acceptable to all individuals and communities. We recognize the need for various components, such as the tools for authentication, keys and certificates, and the ability to express security constraints, but do not recognized any one mechanism as adequate for all situations.

The problem also arises with respect to supporting transport and presentation protocols. The WWW and Prospero[Neu92a, Neu92b] take advantage of whole suites of protocols, such as HTTP [FGMFB97] , FTP [PR85], and Gopher [Ank93] with good reason. The set of transport protocols that is appropriate for different sorts of entities varies widely, is growing and will certainly evolve with time. Again, no one answer is the correct one.

With these issues in mind, our architecture is object-based, with three aspect being particularly important to describe here: naming and name resolution, typing or roles, and enabling an infrastructure of links.

3.1 Naming and name resolution

In general, there are three functions for which names are used: identity, semantics, and access or location. The identity function is that of distinguishing one entity from another by means of the name. Thus two objects intended to be equal, by a definition of equality determined by the name assigner, will have the same name, whereas those intended to be distinct will have different names. For a name to be useful for identification, it must survive at least as long as the object being named and never be reassigned to something else. The reason is that it may be embedded in other objects that themselves have longer lifetimes.

The semantic function is one of embedding knowledge about an object in its name. For such information to be useful, one must be able to verify or trust the correctness of it. This becomes particularly problematic in an evolving universe, where characteristics of an object may change with time. An obvious example of this problem arises when a program is recoded in a different programming language.

The third function, that of access or location is often achieved by embedding location or access information in a name. This has the same problems as embedding semantics. Any long-lived object in a system such as ours is likely to move during its lifetime. Hence, including valid location or access protocol information in a name will mandate that the name must change, thus negating its utility for identification.

Thus, the naming model separates identify from resolution information (location) and semantics. A name is permanent and globally unique. Once assigned to a resource, it will never be reassigned to something else. It contains no semantics, including location information. Location is discovered through a flexible, resilient set of mechanisms that provide hints for location. Other semantic or meta-information about a resource will be handled separately. A realization of this work is occurring within the Internet Engineering Task Force, with Uniform Resource Names (URNs)[SM95], Uniform Resource Locators (URLs)[B-L94b, Kun95], and Uniform Resource Characteristics, which are discussed in some of these documents but not completely defined as yet.

The work that is currently in progress is hint discovery and resolution. Hints may be addresses of resources themselves or addresses or names of resolution servers. Furthermore, as their name implies, they may not be correct. In order to provide resiliency to such failures (the object moved and only reported its new location to some resolver not known to the client), a resolution discovery service is also needed. This work is ongoing both in the IETF[Sol97, DM97] and within our project [Slo97].

Thus the naming model supports longevity, ubiquity, mobility, heterogeneity of supporting transport protocols, and resiliency to failures.

3.2 Role playing

It is important to be able to recognize the behavior of an object. Any object may exhibit more than one behavior. It may be a book to one client and a location service to another, as, for example a phone book is. Thus it may play more than one *role* at a time, and furthermore the set of roles it plays may change with time. Roles are defined hierarchically and allow for multiple inheritance. Because of the singly rooted hierarchy in the `object-role`, all roles must support the `roles-played` action, as the `object-role` does.

There are three aspects of the definition of a role: *actions*, *parts*, and *makers*. In each of these categories some components may be required and others optional. The actions define the abstract functionality of the role, and as such are defined in terms of the signatures of the functions. They are abstract in that the choice of an action does not imply the choice of an implementation of that function. Parts define the abstract structure of an object playing a role. By defining parts abstractly, they are not constrained by any particular representation. Thus, for example, the table of contents of a book might be represented by an array in one implementation and by a linked list in another. Some parts may occur more than once in a role, thus requiring *selectors* to identify individual parts. These may be ordered, unordered but from a restricted set, or completely arbitrary. Lastly we separate out the activity of creating new objects to play a particular role. These functions fall into the group of makers. They do not operate on objects that play the role in question, but rather may take other sorts of arguments (or none) to create a new object playing the specified role.

With this broad definition of role, evolvability and heterogeneity can be well supported.

3.3 Links

Building on this simple model, creating relationships among objects becomes a matter of applying the object model appropriately. The solution for links is that they are first class objects, that play subroles of the superrole `link`. [SV95, VD95]

A link consists of one or more endpoints as defined parts. The realization of an endpoint is a *descriptor* that includes a URN and a *qualifier* for the URN. The qualifier may specify the whole object or some view or component exported through a part of one of the roles the object plays. A *tag* in the qualifier specifies which part of which role is being identified and a *selector* distinguishes the particular part if more than one is available.

By using URNs problems of identity, longevity and mobility are addressed. Since links are first class objects, they have long-lived identities and therefore can be used as the targets of links. In addition, since URNs are independent of location, unlike the URLs used in WWW links, the endpoints are free to move as needed without voiding the meaning of the endpoints and therefore the link. Again, because links are first class objects, we have a simple mechanism for distinguish an extensible set of kinds of relationships. A full analysis of linking models is presented in {SV95, VD95].

4. Core services and functionality

There are two approaches to realizing the Information Mesh architecture. The first is to design from scratch a new universe as a replacement for the World Wide Web. Although this is impractical it is worth understanding what would be needed to do this. The second approach is to understand how one could approach the architecture by modifying the web as it exists at present.

To achieve the first goal a collection of services must be made available within what the applications might view as "the network". The basic services needed are:

- **Name assignment**: this service has the authority to create new names as needed.
- **Name resolution**: once a name has been assigned and distributed to others, they need to be able to reach the object. The resolvers and resolution discovery service are parts of such a service.
- **Object management**: an object management service handles the business of creating, storing, and providing access to an object.
- **Role library**: a role library provides storage and access to role definitions. It may also provide implementations of roles.
- **Authentication distribution service**: this service provides access to such security components as public keys and certificates. It may also provide verification of certificates and other signatures and assignment of new keys as needed.
- **Initialization**: In order for a new participant host to join the Information Mesh, it must learn such information as the location of instances of the above services. This can be done either by the use of well known addresses or by a boot service that provides it on demand, presumably from some well-known address.

Each category describes a class of services, with many possible realizations. It is worth noting here that each of these functions has a significant problem of information management and distribution in a global environment. We do not wish to down-play this problem, but do not have the space here to discuss those problems.

The other alternative, that of modifying the existing infrastructure, is in fact the direction we are moving for now, having built demonstration versions of most of the services above. Our current effort is focussed most directly on naming, name resolution, and the attendant need for public key and certificate distribution, with one name resolution design and prototype completed {Slo97] and a second in the design stages, based on more efficient, but not yet widely deployed multicast protocols.

5. How this fits in

There are a wide variety of related work, and hence we will only refer to a sampling. We do not have space to discuss the numerous references on naming, typing and linking models.

The most obvious, and previously mentioned are the network based information models, such as the World Wide Web, Gopher, and Prospero. None of these addresses the problems of longevity, mobility, and scaling as are done in this architecture.

Moving to the distributed systems end of the spectrum, we find such systems as CORBA, ANSA, and OLE. These sorts of systems move the application further from the network, providing not only an information base but also communications invocation models. The application builder then has very little choice in the matter of paradigms for these. They provide more of a "network operating system" model, rather than an enhanced model of the network.

To summarize, the contribution of this work is a global information infrastructure architecture, the Information Mesh, designed to account for issues of longevity, mobility, evolvability, and resiliency.

References

[Ank93] F. Anklesaria, et al. *The Internet Gopher Protocol (a distributed document search and retrieval protocol)*, IETF RFC 1436, March, 1993.

[B-L94a] T. Berners-Lee et al., *The world wide web*, **Communications of the ACM, 37**(8) (August, 1994), pp. 76-82.

[B-L94b] T. Berners-Lee, *Uniform Resource Locators: A Syntax for the Expression of Access Information of Objects in the Network*, IETF RFC 1738, December, 1994

[BC] T. Berners-Lee and D. Connelly, *Hypertext Markup Language - 2.0*, MIT/W3C September, 1995. Also IETF RFC 1866, November, 1995.

[DM97]R. Daniel, and M. Mealling, *Resolution of Uniform Resource Identifiers using the Domain Name Service*, IETF RFC 2168, June 1997.

[FGMFB97] R. Fielding et al., *Hypertext Transfer Protocol - HTTP /1.1*, IETF RFC 2068, January, 1997.

[HS94] F. Halasz and M. Schwartz, *The Dexter Hypertext Reference*, **Communications of the ACM, 37**(2) (February, 1994), pp. 30-39.

[Joh91] B.C. Johnson, *A Distributed Computing Environment Framework: An OSF Perspective*, Open Software Foundation, DEV-DEC-TP6-1, June 1991.

[JLHB] E. Jul, H. Levy, N. Hutchinson, and A. Black, *Fine-Grained Mobility in the Emerald System*, **ACM Transactions on Computer Systems, 6**, 1 (February, 1988), pp. 109-133.

[Kee88] S. Keene, **Object-oriented programming in Common Lisp: a programmer's guide to CLOS**, Addison Wesley, Reading, MA 1988.

[Kun95] J. Kunze, *Functional Requirements for Internet Resource Locators*, IETF RFC 1736, February, 1995.

[Mic93] Microsoft OLE 2.0 Design Team, *Microsoft OLE 2.0 Design Specification*, **Microsoft Development Library**, April, 1993.

[Neu92a] B.C. Neuman, *Prospero: A Tool for Organizing Internet Resources*, **Electronic Networking: Research, Applications and Policy, 2**(1) (Spring, 1992), pp. 30-37

[Neu92b] B.C. Neuman, *The Prospero File System: A global file system based on the Virtual System Model*, **Computing Systems, 5**(4) (Fall, 1992), pp. 407-432.

[OMG95] Object Management Group, **CORBA: Architecture and Specification**, The Object Management Group, August 1995.

[PR85] J. Postel and J. Reynolds, *File Transfer Protocol (FTP)*.. IETF RFC 959, October, 1985.

[Ree93] R.T.O. Rees, **The ANSA Computational Model**, Architecture Projects Management, Ltd, AR.001.01, February, 1993.

[Slo97] E. C. Slottow, **Engineering a Global Resolution Service**, MIT/LCS/TR-712, June, 1997.

[SM95] K.R. Sollins and L. Masinter, *Requirements for Uniform Resource Names*, IETF RFC 1737, February, 1995.

[Sol97] K.R. Sollins, *Guidelines and a Framework for URN Resolution Systems*, unpublished, currently available from the IETF as draft-ietf-urn-req-frame-02.txt, June, 1997.

[Sun88] Sun Microsystems, *RPC: Remote Procedure Call Protocol Specification Version 2*, IETF RFC 1057, June 1988.

[SV95] K.R. Sollins and J.R. Van Dyke, *Linking in a Global Information Architecture*, Proc. Fourth World Wide Web Conference, Boston, !995, pp. 493-508.

[VD95] J.R. Van Dyke. *Link Architecture for a Global Information Infrastructure*, MIT/LCS/TR-659, June, 1995.

[VDL93] R. J. van der Linden, **An Overview of ANSA**, Architecture Projects Management, Ltd, AR.000.00, May 1993.