Network Working Group
Request For Comments: 1862
Category: Informational

M. McCahill
University of Minnesota
J. Romkey, Editor
M. Schwartz
University of Colorado
K. Sollins
MIT
T. Verschuren
SURFnet
C. Weider
Bunyip Information Systems, Inc.

November 1995

Report of the IAB Workshop on Internet Information Infrastructure, October 12-14, 1994

Status of this Memo

Abstract

This document is a report on an Internet architecture workshop, initiated by the IAB and held at MCI on October 12-14, 1994. This workshop generally focused on aspects of the information infrastructure on the Internet.

1. Introduction

The Internet Architecture Board (IAB) holds occasional workshops designed to consider long-term issues and strategies for the Internet, and to suggest future directions for the Internet architecture. This long-term planning function of the IAB is complementary to the ongoing engineering efforts performed by working groups of the Internet Engineering Task Force (IETF), under the leadership of the Internet Engineering Steering Group (IESG) and area directorates.

An IAB-initiated workshop on the architecture of the "information infrastructure" of the Internet was held on October 12-14, 1994 at MCI in Tysons Corner, Virginia.

In addition to the IAB members, attendees at this meeting included the IESG Area Directors for the relevant areas (Applications, User Services) and a group of other experts in the following areas:

gopher, the World Wide Web, naming, WAIS, searching, indexing, and library services. The IAB explicitly tried to balance the number of attendees from each area of expertise. Logistics limited the attendance to about 35, which unfortunately meant that many highly qualified experts were omitted from the invitation list.

The objectives of the workshop were to explore the architecture of "information" applications on the Internet, to provide the IESG with a solid set of recommendations for further work, and to provide a place for communication between the communities of people associated with the lower and upper layers of the Internet protocol suite, as well as allow experience to be exchanged between the communities.

The 34 attendees divided into three "breakout groups" which met for the second half of the first day and the entire second day. Each group wrote a report of its activities. The reports are contained in this document, in addition to a set of specific recommendations to the IESG and IETF community.

2. Summary

Although there were some disagreements between the groups on specific functionalities for architectural components, there was broad agreement on the general shape of an information architecture and on general principles for constructing the architecture. The discussions of the architecture generalized a number of concepts that are currently used in deployed systems such as the World Wide Web, but the main thrust was to define general architectural components rather than focus on current technologies.

Research recommendations include:

- increased focus on a general caching and replication architecture
- a rapid deployment of name resolution services, and
- the articulation of a common security architecture for information applications.

Procedural recommendations for forwarding this work in the IETF include:

- making common identifiers such as the IANA assigned numbers available in an on-line database
- tightening the requirements on Proposed Standards to insure that they adequately address security

- articulating the procedures necessary to facilitate joining IETF working group meetings, and

- reviewing the key distribution infrastructure for use in information applications

3. Group 1 report: The Distributed Database Problem

Elise Gerich, Tim Berners-Lee, Mark McCahill, Dave Sincoskie, Mike Schwartz, Mitra, Yakov Rekhter, John Klensin, Steve Crocker, Ton Verschuren

Editors: Mark McCahill, Mike Schwartz, Ton Verschuren

3.1 Problem and Needs

Because of the increasing popularity of accessing networked information, current Internet information services are experiencing performance, reliability, and scaling problems. These are general problems, given the distributed nature of the Internet. Current and future applications would benefit from much more widespread use of caching and replication.

For instance, popular WWW and Gopher servers experience serious overloading, as many thousands of users per day attempt to access them simultaneously. Neither of these systems was designed with explicit caching or replication support in the core protocol. Moreover, because the DNS is currently the only widely deployed distributed and replicated data storage system in the Internet, it is often used to help support more scalable operation in this environment -- for example, storing service-specific pointer information, or providing a means of rotating service accesses among replicated copies of NCSA's extremely popular WWW server. In most cases, such uses of the DNS semantically overload the system. The DNS may not be able to stand such "semantic extensions" and continue to perform well. It was not designed to be a general-purpose replicated distributed database system.

There are many examples of systems that need or would benefit from caching or replication. Examples include key distribution for authentication services, DHCP, multicast SD, and Internet white pages.

To date there have been a number of independent attempts to provide caching and replication facilities. The question we address here is whether it might be possible to define a general service interface or protocol, so that caches and replica servers (implemented in a variety of ways to support a range of different situations) might

interoperate, and so that we might reduce the amount of wasted re-implementation effort currently being expended. Replication and caching schemes could form a sort of network "middleware" to fulfill a common need of distributed services.

It should be noted that it is an open question whether it would be feasible to define a unified interface to all caching and replication problems. For example, very different considerations must go into providing a system to support a nationwide video service for 1,000,000 concurrent users than would be needed for supporting worldwide accesses to popular WWW pages. We recommend research and experimentation to address this more general issue.

3.2 Characteristics of Solutions

While on the surface caching and replication may appear to occupy two ends of a spectrum, further analysis shows that these are two different approaches with different characteristics. There are cases where a combination of the two techniques is the optimal solution, which further complicates the situation.

We can roughly characterize the two approaches as follows:

Caching:

- a cache contains a partial set of data
- a cache is built on demand
- a cache is audience-specific, since the cache is built in response to demands of a community

Replication:

- replicated databases contain the entire data set or a server-defined subset of a given database
- a replicated database can return an authoritative answer about existence of an item
- data is pushed onto the replicating server rather than pulled on demand

While there are important differences between caches and replicated databases, there are some issues common to both, especially when considering how updates and data consistency can be handled.

A variety of methods can be used to update caches and replicas:

- master-slave
- peer-to-peer
- flooding techniques (such as that used by NNTP).

Which strategy one chooses influences important characteristics of the cache or replicated database, such as:

- consistency of data
- is locking used to achieve consistency? this influences performance...
- are there a priori guarantees of existence of an item in the database (is the answer authoritative, do you detect conflicts after the fact, or is there no guarantee on authoritativeness of the answer?)

Consistency guarantees depend on the granularity of synchronization (ms, sec, hr, day), and there are cases where it is acceptable to trade consistency for better performance or availability. Since there is a range of qualities of service with respect to consistency and performance, we would like to be able to tune these parameters for a given application. However, we recognize that this may not be possible in all cases since it is unlikely one can implement a high performance solution to all of these problems in a single system.

Beyond simply performing replication or caching, there is a need for managing cache and replication servers. There are several models for organizing groups of caches/replication servers that range from totally adaptive to a rigidly administered, centrally controlled model:

- a club model. Minimal administrative overhead to join the club. Participation is a function of disk space, CPU, available network bandwidth.
- centrally coordinated service. Here administrators can take advantage of their knowledge of the system’s topology and the community they intend to serve. There may be scaling problems with this model.
- hybrid combinations of the club and centrally coordinated models

There are a couple of models for how to organize the management of a group of cooperating servers, but this does not address the question of what sorts of commands the manager (be it a person or a program) issues to a cache or replicated server. A manager needs to be able to address issues on a server such as:

- control of caching algorithms, defining how information is aged out of the cache based on disk space, usage demands, etc. This is where you would control time-to-live and expiry settings.

- flushing the cache. There are circumstances where the information source has become inaccessible and the normal cache aging strategy is inappropriate since you will not be able to get the information again for an indeterminate amount of time.

- management control might also be a way for information providers to control how information is pushed on servers for maintaining data consistency, but this raises tricky problems with trust and authentication.

Given a common set of management controls needed, a common protocol would allow for simplified management of a collection of caching and replicating servers since you would be able to both control them with a single set of commands and query them about their capabilities. A common language/protocol would also allow different implementations to interoperate.

Replicating or caching information immediately raises issues of billing, access control and authentication. Ignoring authentication and access control issues simplifies the replication and caching problem a great deal. Exactly who is running the replication or caching server makes a big difference in how you approach this issue. If the information publisher runs a set of servers, they can easily handle billing and authentication. On the other hand, if an organization is running a cache on its firewall (a boundary cache), and purchasing information from a vendor, there are sticky issues regarding intellectual property in this scenario.

Selecting an appropriate cache or replica of a database is simple in the case of a captive user group (for instance a company behind a firewall). In this case, configuring the user’s software to go through one or more boundary caches/replication servers directs the users to the closest server. In the more general case, there are several replicated/cached copies of an object, so you may receive several URLs when you resolve a URN. How do you select the best URL?

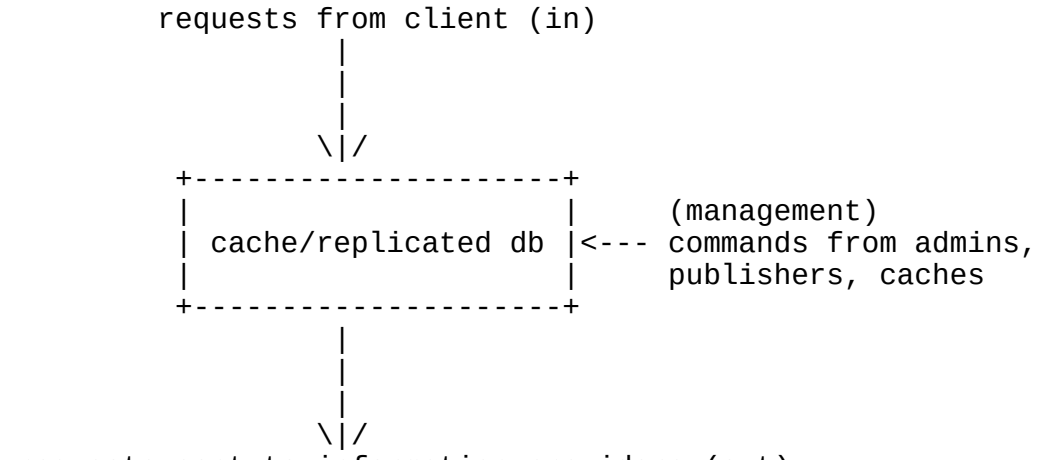
Either client developers create ad hoc performance metrics or (in an ideal world) the lower level protocols would give the client

application some guidance about the "closest" copy of the object. In other words, if better information about network performance was available from lower levels of the protocol stack, applications would not have to build ad hoc models of network topology

We did not model the functions of a cache/replication server in detail, but we did an (incomplete) model of some of the functions (see Figure 1). The idea here was to start work on a general form which might include features such as a push function for use in both maintaining consistency and in preloading information that the information publisher believes will be requested in the near future.

Preloading information via a push command might be a function of observed behavior patterns (when you ask for A you'll probably want B and C). The decision about what to preload can be made either by the information publisher or by the cache server. The cache server has the advantage that it has better knowledge of the use patterns of its community. The distributed nature of links to other servers also limit the knowledge of a single information publisher. In any case, being able to accurately predict usage patterns can result in significant performance enhancements for caches.

Figure 1: a rough cut at functions



in: (requests from a client)

- give me meta-info about cached object (how up-to-date, ttl, expiry, signatures/checksum, billing information)

- give me the object

- go get the object from the net

- cache, what objects should I pre-fetch?
  (this assumes that the client software believes that the cache/replica has some knowledge of use patterns and can predict what the user will do next)

out: (requests sent to an information publisher or a cache further up the food chain)

- server, do I have latest copy of this object?

- give me object x and the meta data for object x

- I have a copy of object x (announcing you have a copy of object x to other caches or URN to URL server)

- info publisher, what objects should I pre-fetch?
  (this assumes that the information publisher has some knowledge of use patterns and can predict what the user will do next)

management: (commands from administrators, other cooperating caches, and object publishers)

- turn parameters (e.g. consistency) on/off

- flush the cache

- there's a new version of object x, take it

3.3 Recommendations

Caching and replication are important pieces of Internet middleware, and solutions need to be found soon. Caches and replicas have different performance characteristics, and there are cases where a combination of the two provides the best solution. There are also many strategies for updating and maintaining consistency of caches and replicated databases, and we do not believe any single implementation can suffice for the broad range of needs in the Internet. One possible solution would be to define a general protocol for a replicated distributed database and for caching so that different information application implementations can interoperate and be managed via a common management interface. A common protocol would provide a framework for future protocols (e.g., URN2URL, DHCP) or existing protocols (e.g., Gopher or WWW) that presently lack a consistent solution.

4. Group 2A report: Building an Information Architecture

Karen Sollins, Abel Weinrib, Barry Leiner, Clifford Neuman, Dan LaLiberte, Erik Huizer, John Curran, John Klensin, Lixia Zhang, Michael Mealling, Mitchell Charity, Mike St. Johns, Paul Mockapetris

This group took as its central agenda exploring an information architecture, the services that would instantiate such an architecture, and the functional interfaces between a realization of such an architecture and both layers on which it would sit and the layers that would sit on it. In order to describe an architecture, one must describe not only what it includes, but also what it excludes.

4.1. The core model and service structure

The general architecture has as its centerpiece objects, or as they are known in the Uniform Resource Identifier Working Group, resources. An object in this architecture has several characteristics. First, it has an identifier, assigned within the context of some namespace. Such an identifier is globally unique and will not be reassigned to another object. Thus, it can be said to be globally unique for a long time. Because such an identifier must remain unique for all time, it cannot contain location-relevant information ... locations can and will be reused. Also, since resources may appear in zero, one, or many locations simultaneously, location-dependent information can lead to a vast number of identifiers for an object, which will make it difficult to identify separately retrieved copies of an object as being the same object. These locations are defined by the supporting layers that provide transport and access. Therefore the definition of locations is not within the architecture, although their existence is accepted. Second, an object will support one or more abstract types. Further determination beyond this statement was not made. One can conclude from these two points that an object cannot be part of such an architected universe without having at least one such identifier and without supporting at least one type if it has at least one location.

In addition, the architecture contains several other components. First, there will be a prescribed class of objects called links that express a relationship among other objects including the nature of that relationship. It is through links that composite objects composed of related objects can be created and managed. Finally, there is a need for several sorts of meta-information, both in order to discover identifiers (e.g. for indices and in support of searching) and to aid in the process of mapping an identifier to one or more potential locations. Both of these sorts of meta-information are associated with objects, although they will be used and therefore

most likely managed differently, to support their distinctive access and update requirements.

Given this architecture of information objects, one can identify several boundary points. First, something that does not have an identifier or type is outside the architecture. Second, the architecture does not, at this point, include any statement about computations, or communications paradigms other than second-handedly by assuming that traversal of links will occur. Third, although pre-fetching, caching, and replication are important, such details may be hidden from higher level software components, and thus are not part of the data model exposed to the application in the normal case (though some applications may want to specify such characteristics).

Now one can ask how such a model fits into a layered network model, how it might be modularized and realized. We envisioned this information layer as an information "wholesale" layer. It provides the general, broad model and provision of shared, network-based information. Above this sit the "retailers," the marketers or providers of information to the marketplace of applications users. Below the "wholesalers" lie the providers of "raw materials." Here will be the provision of supporting mechanisms and architecture from which information objects can come.

The remainder of this group’s report describes the modular decomposition of the wholesale layer, including the interactions among those modules, separate discussions of the interactions first between the retail and wholesale layers and then between the wholesale and raw material layers. The report concludes with recommendations for where the most effective immediate efforts could be made to provide for the wholesale layer and make it useful.

4.2. The Wholesale Layer

In order to realize the information architecture in the network a variety of classes of services or functionality must be provided. In each case, there will be many instances of a sort of service, coordinating to a lesser or greater degree, but all within the general Internet model of autonomy and loose federation. There also may be variants of any sort of service, to provide more specialized or constrained service. In addition, services may exist that will provide more than one of these services, where that is deemed useful. Each such service will reside in one or more administrative domains and may be restricted or managed based on policies of those domains. The list of core services is described below. Because there are many interdependencies, there may often be forward references in describing a service and its relationships to other services.

* RESOURCE DISCOVERY: Much of the activity of resource discovery, indexing and searching, will be in the domain of the retailers, although there are supporting hooks that can be provided by the wholesaler layer as well. A resource discovery service will hold mappings from descriptions to identifiers of objects. They will need to be queried. Thus there is a general functionality for a wholesale layer service that answers queries formulated in certain ways and responds with identifiers. The business of on what basis indices are computed or how they are managed will be domain specific.

* NAMING or IDENTIFICATION: There are two aspects to assigning an identifier to an object, one in the wholesale layer, and one, arguably, in the retail layer. In the wholesale layer, one can generate identifiers that are guaranteed to be unique. In the retail layer one might ask the question about whether two objects are the same or different by the rules of an identification authority that therefore would determine whether they should bear the same or different identification from that authority. It should be noted that the URI Working Group has included these two functions in the requirements document for URNs.

An identification service will obviously provide functionality to the uniqueness authority. It will also provide identification in the process of publication of objects, as will be discussed below, in the management of resource discovery information, object location and storage services, as well as cache and replication management.

* NAME or IDENTIFICATION RESOLUTION: Since identifiers are presumed to be location independent, there is a need for a resolution service. Such a service may sometimes return other identifiers at this same level of abstraction (the equivalent of aliases) or location information, the information delivered to a transport service to access or retrieve an object.

* OBJECT RETRIEVAL: Object retrieval is tightly coupled to resolution, because without resolution it cannot proceed. Object retrieval provides the functionality of causing a representation of an object to be provided locally to the requester of an object retrieval. This may involve the functionality of object publication (see below) and object storage, caching and replication services as well as the supporting transport facilities.

* OBJECT PUBLICATION: When an object comes into existence in the universe of the information infrastructure, it is said to be "published." There will be two common scenarios in publication. One will be the use of tools to directly enter and create the information that comprises an object in the information infrastructure. Thus there may be object creation tools visible to users in applications.

In contrast there may also be tools outside the information infrastructure (for example word processing or text editing tools) that provide for the entry of data separately from the operation of assigning an object an identifier and causing it to support information infrastructure definitions of objects. Thus, there will also be visible at the interface between the wholesale and retail layers the ability to cause some pre-existing data to become one or more objects. In addition to interacting with the identification service, publication is likely to cause interaction with object storage, and possibly caching and replication.

* DEFINITIONS: If the information infrastructure is to both survive and evolve over a long time period, we must be prepared for a wide variety and growing number of different sorts of information with different functionalities that each supports. For objects available on the net, the functionality that each provides must be exposed or able to be learned. To do this objects must be able to indicate by name or identifier the types of functionality they are supporting. Given such an identifier, an object is only useful to a client, if the client can discover the definition and perhaps a useful implementation of the type in question. This will be acquired from a definitions service, which will be used in conjunction with applications themselves directly, object publication, and object retrieval.

* ATTRIBUTE MANAGEMENT: The attributes considered here relate to policy, although any understanding of that policy will be above the wholesale level. There are, for example, access management and copyright attributes. There is a question here about whether there is or should be any access time enforcement or only after the fact enforcement. The information is likely to be in the form of attribute-value pairs and must be able to capture copyright knowledge effectively.

* ACCOUNTING: An accounting service provides metering of the use of resources. The resources wholly contained in the wholesale layer are the services discussed here. It will also be important to provide metering tools in the wholesale layer to be used by the retail layer to meter usage or content access in that layer. Metering may be used for a variety of purposes ranging from providing better utilization or service from the resources to pricing and billing. Hence accounting services will be used by object storage, caching and replication, lower layer networking services, as well as pricing and billing services. In the form of content metering it will also interact with attribute management.

* PRICING, BILLING and PAYMENT: Pricing and payment services straddle two layers in the information infrastructure. Servers that maintain account balances and with which users interact to retrieve and edit account information are applications that will be built on top of wholesale layer services. Pricing will be determined in the applications environment for application level activities. However, it must be possible for middle layer services to process payment instruments analogous to cash, credit card slips, and checks, without an understanding of the specific implementation of the payment mechanism. Application programming interfaces supporting payment should be provided, and a common tagged representation of payment instruments should allow instruments from a variety of payment systems to be presented within middle layer protocols.

* OBJECT STORAGE, CACHING and REPLICATION: There is a recognition that caching and replication are important, but the discussion of that was left to another group that had taken that as the focus of their agenda. Object storage will take an object and put it somewhere, while maintaining both the identity and nature of the object. It is tightly coupled to caching and replication, as well as accounting, often in order to determine patterns of caching and replication. It is also tightly coupled to object publication, translation, and provides interfaces to both supporting storage facilities such as local file systems, as well as direct access from applications, needing access to objects.

* TRANSLATION: A translation service allows an object to behave with a nature different than that it would otherwise support. Thus, for example, it might provide a WYSIWYG interface to an object whose functionality might not otherwise support that, or it might generate text on the fly from an audio stream. Translation services will be used by object publication (allowing for identification of an object including a translation of it) and with object storage, providing an interface only within the wholesale or to the retail layers.

* SERVER AND SERVICE LOCATION: It will be necessary as part of the infrastructure to be able to find services of the kinds described here and the servers supporting them. This service has direct contact with the lower layer of raw materials, in that it will provide, in the final analysis, the addresses needed to actually locate objects and services using lower level protocols, such as the existing access protocols in use today, for example FTP, SMTP, HTTP, or TCP. This service will provide functionality directly to resource discovery as well as remote object storage services.

* ADAPTIVE GLUE: This is not a single service as much as a recognition that there must be a path for a flow of information between the network layers and the applications. The application may have constraints, based both on its own needs as well as needs of the objects in the wholesale layer. Only the application can really know what compromises in services provided below are acceptable to it. At the same time, the supporting network layers understand what qualities of service are available at what price. Hence there is the potential for flow of information both up and down through the wholesale layer, perhaps mediated by the wholesale layer. Hence the adaptive glue has hooks into all three levels.

* SECURITY: Security services will be a critical piece of the infrastructure architecture. For any real business to be conducted, organizations must make their information available over the network, yet they require the ability to control access to that information on a per user and per object basis. To account properly for the use of higher level services, organization must be able to identify and authenticate their users accurately. Finally, payment services must be based on security to prevent fraudulent charges, or disclosure of compromising information.

The two biggest problems in providing security services at the wholesale layer are poor infrastructure and multiple security mechanisms that need to be individually integrated with applications. The poor state of the infrastructure is the result of a lack of an accepted certification hierarchy for authentication. A commonly held position is that there will not be a single hierarchy, but there must be established authorities whose assertions are widely accepted, who indirectly certify the identities of individuals with which one has not had prior contact.

Integration with applications is made difficult because, though security services are themselves layered upon one another, such services do not fit into the information architecture at a single layer. By integrating security services with lower layers of the information infrastructure, security can be provided to higher layers, but some security information, such as client’s identity, may be needed at higher layers, so such support will not be completely transparent. Further, the security requirements for each middle layer information service, and of the application itself, must be considered and appropriate use must be made of the middle-layer security services applied.

Integration with applications will require user demand for security, together with common interfaces such as the GSS-API, so that applications and middle layer information services can utilize the security services that are available, without understanding the

details of the specific security mechanism that is employed.

* BOOTSTRAPPING: In order for a newly participating machine to join the infrastructure, it must have some way of finding out about at least one instance of many of the services described here. This can be done either by providing it with some form of configuration provided by the human bringing it up or by a bootstrapping service. The bootstrapping service is more flexible and manageable; it is included here in recognition that this information must be provided in some form or other. The bootstrapping service will sit directly on the raw materials layer and will have contact with all the services described here.

This completes the description of the services as identified by this group in the wholesale layer. Although this section suggests which services have interfaces to the retail and raw materials layers, each of these topics will need to be described separately as well, to clarify the functionality expected by each layer of the layer below.

3. Interface to retail layer

The interface to the retail layer is the embodiment of the object model and attendant services. Thus the interface provides the application environment with a collection of objects having identifiers for distinguishing them within the wholesale layer and support for a typing or abstract functionality model. It provides for the ability to create or import objects into this object world by the publication paradigm, and allows objects to evolve to support new or evolving functionality through the translation paradigm. Access to the objects is provided by object storage, enhanced with caching and replication services and mediated by the attributes managed by attribute management and accounting or content metering. Discovery of resources (figuring out which identifier to be chasing) is provided by resource discovery services. Types are registered and hence available both as definitions and perhaps in the form of implementations from a definition service. Lastly, there is a vertical model of providing the two-way services of adaptive glue for quality of service negotiation and for security constraints and requirements, with access and services at all three layers.

4. Interface to the raw materials layer

The raw materials layer falls into networking and operating systems. Hence it provides all those services currently available from current networking and operating systems. Wholesale services such as object management will be dependent on local operating system support such as a file system, as well as perhaps transport protocols. In fact, all instances of any of the above services will be dependent on local

storage, process management, local access control and other security mechanisms, as well as general transport protocols for communications both often among services of the same sort and among services dependent on each other that may not be collocated. In addition the group identified a set of issues that appear important for the networking components of the raw materials layer to provide to the wholesale layer in addition to the basic best effort transmission services that are commonly available. These take the form of a wish list with the recognition that they are not all equally easy or possible.

* Connectivity: It is useful and important for the operation of applications and the wholesale services to understand what connectivity is currently available. The group identified four categories of connectivity that it would be useful to know about represented by four questions:

1) Is there a wire out of the back of my machine?

2) Am I connected to a router?

3) Am I connected to the global internet? (Can I get beyond my own domain?)

4) Am I connected to a specific host?

These are probably in increasing difficulty of knowing.

* Connectivity forecast: Although this is recognized as either extremely difficult or impossible to do, some form of connectivity forecast would be very useful to the upper layers

* Bandwidth availability and reservation: It is useful for the application to know both what bandwidth might be available to it and, better yet, for it to be able to make some form of reservation.

* Latency availability and reservation: It is useful for the application to know both what latency the network is experiencing and, better yet, be able to set limits on it by means of a reservation.

* Reliability availability and reservation: Again, reliability constraints are important for many applications, although they may have differing reliability constraints and may be able to adapt differently to different circumstances. But, if the application could make a statement (reservation) about what level of unreliability it can tolerate, it might be able to make tradeoffs.

* Burstiness support: Although it is unlikely that the network can make predictions about the burstiness of its services, if the application can predict to the network its burstiness behavior, the network might be able to take advantage of that knowledge.

* Service envelope: It is possible that, as an alternative to the above four issues, the raw materials layer could negotiate a whole service envelope with the layers it is supporting.

* Security availability: In many cases, it will be important for the upper layers to be able to know what sorts and levels of security are available from the raw materials layer. This is true of both any operating system support as well as transmission.

* Cost: If there is to be usage charging at other than fixed flat rates, it will be important for applications and users to understand what those costs or at least estimates of them will be.

* Policy routing: If it will be important for transport services to support policy routing, it will be important for users of the transport services to identify into which policy classes they might fall.

4.5. Recommendations

This group has two categories of recommendations. One is those services in the wholesale layer that will both be especially useful and readily achieved because work is soon to be or already underway. The other set of recommendations was a three item rank ordering of services that are most important for the lower layer to provide to the wholesale layer.

Within the wholesale layer, the first services that should be provided are:

* Object retrieval,

* Name resolution,

* Caching and replication.

In addition, the group rank ordered three areas in which there would be quick payoff if the raw materials layer could provide them. They are:

1. Connectivity

2. Bandwidth, latency, and reliability or service envelope

3. Security constraints on communication and transactions

5. Group 2B Report: Components of an Internet Information Architecture

Cecilia Preston, Chris Weider, Christian Huitema, Cliff Lynch, John Romkey, Joyce Reynolds, Larry Masinter, Mitra, Jill Foster

Group 2B discussed various aspects of problems in the Internet Information Infrastructure, thinking about recommendations to the IESG to focus on particular areas, and also paying attention to some of the philosophical and economic backgrounds to some of the problems. Economics can dictate some points of architecture: one can see economically why a publisher might bear the burden of the costs of publishing, or a consumer might bear the burden of costs associated with consumption, but not how some free-floating third party would necessarily bear the costs of providing services (such as third-party translators).

The group discussed the following topics:

access(URL)

gateways

URN resolution

definitions

updates

service location

cache & replication

security & authentication

payments, charging

presentation

search & index

metainformation

boot service

general computation

5.1 URNs

There are several issues in the use of Uniform Resource Names and Uniform Resource Locators. URN resolution is a database lookup that returns the URLs associated with a URN. The architecture must take into account not only how the lookup is performed, but how the database is maintained. Both the lookup problem and the update problem must be solved at the same time to allow deployment of URNs.

There are at least two problems in human interaction with unique names. First, the notion of a unique name is a fallacy. Unique naming cannot be enforced. Names may be forged or may simply be duplicated due to human error. The architecture must accept this observation and still operate in the face of it. Designing for global uniqueness, but not requiring it, was adequate. Errors based on names not being unique are likely to be insignificant compared to other errors.

Also, people frequently make assertions and assumptions about names rather than the documents that are being named. Making assertions about names is working at the wrong level of indirection. Making assumptions about names, such as determining the contents of the named object from the syntax of the name, can lead to nasty surprises.

Having a single, unified naming system is vital. While it is healthy to have multiple competing forms of other aspects of the information architecture, the naming system is what ties it all together. There must be only one naming system. If there is more than one, it may not be possible to compare names or to lookup locations based on names, and we will continue (to our detriment) to use locators rather than names.

5.2 Global Service Location

The IANA has become the central switch point for service identification. and recommended that numbers that are formally defined and kept in documents for use in distributed information systems (for instance, Assigned Numbers) should also be distributed online in some kind of database for use by applications. This distribution requires both an access method (perhaps multiple access methods) and an update method.

5.3 Security

Issues involving security arose over and over again. Security includes things like validation of authority, confidentiality, integrity of data, integrity of services, access control. The group agreed that, although often overlooked, confidentiality is important,

and, more strongly: anonymity is important. It should be possible to access documents or objects without the architecture requiring you to leave digital fingerprints all over the place.

Security must occur on an end-to-end basis. Documents or objects used on the Internet may not only traverse the Internet. Relying on security mechanisms in the underlying protocol suite does not necessarily provide end-to-end authentication or confidentiality.

Currently lower layer security is ill-defined and widely unimplemented. Designers building information applications atop the Internet currently receive little guidance in how to design security features into their applications, leading to weak ad hoc or nonexistent security in new applications. Designers are also unclear as to how to deal with the "security considerations" section that is mandatory in RFCs, and often fill them with boilerplate text.

Furthermore, retrofitting security into existing architectures does not work well. The best systems are built considering security from the very beginning. Some systems are being designed that, for instance, have no place for a digital signature to authenticate the data they pass. These issues apply to data management as well.

The group makes the following recommendations to the IESG regarding security:

A. Develop and communicate a security model usable by designers of information applications - current models are not considered usable.

B. RFC authors should be given advice on what security considerations need to be outlined and how to write them. The IESG security area should prepare guidelines for writing security considerations.

C. Proposed Standards should not be accepted by the IESG unless they really consider security. This will require that recommendations A and B have been implemented and that the guidelines have received enough visibility to reasonably expect authors to know of their existence.

D. Develop security modules usable by the implementors of information clients and servers - reusable across many different, heterogeneous applications and platforms.

E. Make clear what security services you can expect from the lower layers.

F. Make sure that the key distribution infrastructure is reviewed for usability by information applications.

5.4 Search and Index

Searching is looking through directories that point to information. Indexing is scanning information to create directories. A "unified directory" is the result of combining several indices.

Indexing is currently done on the Internet via many mechanisms. Given the current ad hoc nature of the indexing, information is frequently indexed multiple times. This is wasteful, but due to the current economics of the Internet, it tends not to cost more money. If the Internet (or parts of thereof) transitions to usage based charging, it may cost the information provider too much to allow the information to be indexed. In general, the provider should have control over how the information they control is indexed.

Above all, the architecture should not encourage a situation where information is normally not indexed. It should encourage the collection of indexing data only a single time. Having a local computation of a summary which is sent to a search/index server is vastly preferable to having that server "walk the net" to discover information to index.

Indexing and search techniques are quite varied. It is quite likely that index and search are too close to general computation to try to standardize on a single protocol for either. Instead, it is important that the architecture allow multiple search techniques. There are currently certain types of indices that can only be generated by humans because of their level of semantic content. There are large differences in the quality and usability of indices that are machine-generated vs. human generated.

Unified directories tend to combine indexing results from quite different techniques. The architecture should constrain indexing so that it remains possible to merge the results of two searches done by different protocols or indexing systems. Returning information in standard formats such as URNs can help this problem.

Vocabulary issues in search and index are very difficult. The library and information services communities do not necessarily use vocabulary that is consistent with the IETF community, which can lead to difficult misunderstandings.

"Searching the Internet" is an inappropriate attempt to categorize the information you’re attempting to search. Instead, we search certain public spaces on the Internet. The concept of public space vs. private space on the Internet deserves further investigation.

Indexing can run afoul of access control considerations. Access control must be done at the object, but access control information should be propagated through indices as well. The index should be able to say "you're not allowed to ask that" rather than the user attempting to retrieve the object and being denied.

An architectural point was raised that an index query should return the same result independent of who is asking. This is an important notion in the Domain Name System. This is inconsistent with some real-world indexing (for instance, corporate record management systems) which doesn't want to admit that some documents exist if you're not allowed to read them.

5.5 Miscellaneous

Electronic mail, netnews, FTP and the web are frequently used to access information on the net today. Each protocol seems to provide a consistent view of the information on the Internet. In addition, the recent popularity of multi-protocol clients such as Mosaic seem to imply that the information content of the Internet is uniformly retrievable and manageable. This perception is misleading because most protocols are used for other applications than they were originally designed for. In addition, Telnet, which has no concept of information retrieval and management, is often used to access information as well, for example in DIALOG and card file accesses. Since each protocol has different access and management capabilities, the inconsistencies show up in erratic search and retrieval results, puzzling error messages, and a basic lack of standard techniques for dealing with information. A consistent underlying information architecture will go a long way towards alleviating these problems.

As the information architecture develops we should reconsider the electronic mail and netnews architecture in terms of the new architecture.

The group noted that there have been difficulties in scheduling joint working group meetings and recommends that there be a clearly defined process inside the IETF to facilitate scheduling such meetings.

6. Conclusions and Recommendations

The workshop provided an opportunity for ongoing conversations about the architecture to continue and also provided space for focused examination of some issues and for some new voices and experience from other areas of Internet growth to participate in the architectural process.

Part of the conclusion of the workshop is a set of recommendations to the IESG and IETF community.

Recommendations on research/implementation directions:

1. Caching and replication are important and overlooked pieces of Internet middleware. We should do something about it as soon as possible, perhaps by defining an architecture and service model for common implementation.

2. Within the 'wholesale' layer, i.e. within the layer which provides a consistent view of the information resources available on the Internet, the first services that should be provided are:

* Object retrieval,

* Name resolution,

* Caching and replication.

3. There would be quick payoff if the raw materials layer, i.e. the layer in which information resources are physically transmitted to computers, could provide the following services:

* Connectivity

* Bandwidth, latency, and reliability or a service envelope

* Security constraints on communication and transactions

4. Develop security modules usable by the implementors of information clients and servers - reusable across many different, heterogeneous applications and platforms

Recommendations to the IESG, IETF, and IANA

1. Numbers that are formally defined and kept in documents in distributed information systems (for instance, Assigned Numbers) should be available in some kind of database for use by applications.

2. Develop and communicate a security model usable by designers of information applications - current models are not considered usable or are not widely accepted on the Internet.

3. RFC authors should be given advice on how security considerations need to be written. The IESG security area should prepare guidelines for writing security considerations.

4. Proposed Standards should not be accepted by the IESG unless they really consider security. This will require recommendations 2 and 3 to be implemented first.

5. Make clear what security services you can expect from the lower layers.

6. Make sure that the key distribution infrastructure is reviewed for usability by information applications.

7. There needs to be a process inside the IETF for scheduling a joint meeting between two working groups - for example, so that the key distribution WG can meet jointly with IIIR.

APPENDIX A - Workshop Organization

The workshop was held at MCI's facility in Tyson Corners, Virginia. The workshop organizers and attendees wish to thank MCI for the use of their facilities to host the workshop.

All attendees met in joint session for the first half of October 12. They then split into three groups. The first group considered the "distributed database" problem which has arisen over and over again in the design of parts of the Internet. The two other groups met to consider a list of issues pertaining to the information infrastructure. The groups ran independently until the morning of October 14, when they met again in joint session.

The following people attended the workshop:

Abel Weinrib	abel@bellcore.com
Barry Leiner	BLeiner@ARPA.MIL
Cecilia Preston	cpreston@info.berkeley.edu
Chris Weider	clw@bunyip.com
Christian Huitema	Christian.Huitema@SOPHIA.INRIA.FR
Cliff Lynch	calur@uccmvsa.ucop.edu
Clifford Neuman	bcn@isi.edu
Dan LaLiberte	liberte@ncsa.uiuc.edu
Dave Sincoskie	sincos@THUMPER.BELLCORE.COM
Elise Gerich	epg@MERIT.EDU
Erik Huizer	Erik.Huizer@SURFnet.nl
Jill Foster	Jill.Foster@newcastle.ac.uk
John Curran	jcurran@near.net
John Klensin	klensin@infoods.mit.edu
John Romkey	romkey@asylum.sf.ca.us
Joyce Reynolds	jkrey@isi.edu

Karen Sollins	sollins@lcs.mit.edu
Larry Masinter	masinter@parc.xerox.com
Lixia Zhang	LIXIA@PARC.XEROX.COM
Mark McCahill	mpm@boombox.micro.umn.edu
Michael Mealling	Michael.Mealling@oit.gatech.edu
Mitchell Charity	mcharity@lcs.mit.edu
Mike Schwartz	schwartz@cs.colorado.edu
Mike St. Johns	stjohns@DARPA.MIL
Mitra	mitra@pandora.sf.ca.us
Paul Mockapetris	pvm@zephyr.isi.edu
Steve Crocker	Crocker@TIS.COM
Tim Berners-Lee	tbl@info.cern.ch
Ton Verschuren	Ton.Verschuren@surfnet.nl
Yakov Rekhter	yakov@WATSON.IBM.COM

Security Considerations

This memo discusses certain aspects of security and the information infrastructure. It contains general recommendations about security enhancements required by information applications on the Internet.

Authors' Addresses

Mark McCahill
University of Minnesota
room 190 Shepherd Labs
100 Union Street SE
Minneapolis, MN 55455
EMail: mpm@boombox.micro.umn.edu

John Romkey [Editor]
1770 Massachusetts Ave. #331
Cambridge, MA 02140
EMail: romkey@apocalypse.org

Michael F. Schwartz
Department of Computer Science
University of Colorado
Boulder, CO 80309-0430
EMail: schwartz@cs.colorado.edu

Karen Sollins
MIT Laboratory for Computer Science
545 Technology Square
Cambridge, MA 02139-1986
EMail: sollins@lcs.mit.edu

Ton Verschuren
SURFNet
P.O. Box 19035
3501 DA Utrecht
The Netherlands
EMail: Ton.Verschuren@surfnet.nl

Chris Weider
Bunyip Information Systems
310 St. Catherine St. West
Suite 300
Montreal, PQ H2A 2X1
CANADA
EMail: clw@bunyip.com