Network Working Group
Request for Comments: 2768
Category: Informational

B. Aiken
J. Strassner
Cisco Systems
B. Carpenter
IBM
I. Foster
Argonne National Laboratory
C. Lynch
Coalition for Networked Information
J. Mambretti
ICAIR
R. Moore
UCSD
B. Teitelbaum
Advanced Networks & Services, Inc.
February 2000

Network Policy and Services: A Report of a Workshop on Middleware

Status of this Memo

Copyright Notice

Abstract

An ad hoc middleware workshop was held at the International Center for Advanced Internet Research in December 1998. The Workshop was organized and sponsored by Cisco, Northwestern University's International Center for Advanced Internet Research (iCAIR), IBM, and the National Science Foundation (NSF). The goal of the workshop was to identify existing middleware services that could be leveraged for new capabilities as well as identifying additional middleware services requiring research and development. The workshop participants discussed the definition of middleware in general, examined the applications perspective, detailed underlying network transport capabilities relevant to middleware services, and then covered various specific examples of middleware components. These included APIs, authentication, authorization, and accounting (AAA) issues, policy framework, directories, resource management, networked information discovery and retrieval services, quality of service,

security, and operational tools. The need for a more organized framework for middleware R&D was recognized, and a list of specific topics needing further work was identified.

Table of Contents

Introduction

This document describes the term "middleware" as well as its requirements and scope. Its purpose is to facilitate communication between developers of both collaboration based and high-performance distributed computing applications and developers of the network infrastructure. Generally, in advanced networks, middleware consists of services and other resources located between both the applications and the underlying packet forwarding and routing infrastructure, although no consensus currently exists on the precise lines of demarcation that would define those domains. This document is being developed within the context of existing standards efforts. Consequently, this document defines middleware core components within the framework of the current status of middleware-related standards activities, especially within the IETF and the Desktop Management Task Force (DMTF). The envisioned role of the IETF is to lead the work in defining the underlying protocols that could be used to support a middleware infrastructure. In this context, we will leverage the information modeling work, as well as the advanced XML

and CIM/DEN-LDAP mapping work, being done in the DMTF. (The recently constituted Grid Forum is also pursuing relevant activities.)

This document also addresses the impact of middleware on Internet protocol development. As part of its approach to describing middleware, this document has initially focused on the intersections among middleware components and application areas that already have well defined activities underway.

This document is a product of an ad hoc Middleware Workshop held on December 4-5 1998. The Workshop was organized and sponsored by Cisco, Northwestern University's International Center for Advanced Internet Research (iCAIR), IBM, and the National Science Foundation (NSF). The goal of the workshop was to define the term middleware and its requirements on advanced network infrastructures as well as on distributed applications. These definitions will enable a set of core middleware components to subsequently be specified, both for supporting advanced application environments as well as for providing a basis for other middleware services.

Although this document is focused on a greater set of issues than just Internet protocols, the concepts and issues put forth here are extremely relevant to the way networks and protocols need to evolve as we move into the implementation stage of "the network is the computer". Therefore, this document is offered to the IETF, DMTF, Internet2, Next Generation Internet (NGI), NSF Partnerships for Advanced Computational Infrastructure (PACI), the interagency Information Technology for the 21st Century (IT2) program, the Grid Forum, the Worldwide Web Consortium, and other communities for their consideration.

This document is organized as follows: Section 1 provides a contextual framework. Section 2 defines middleware. Section 3 discusses application requirements. Subsequent sections discuss requirements and capabilities for middleware as defined by applications and middleware practitioners. These sections will also discuss the required underlying transport infrastructure, administrative policy and management, exemplary core middleware components, provisioning issues, network environment and implementation issues, and research areas.

1.0 Contextual Framework

Middleware can be defined to encompass a large set of services. For example, we chose to focus initially on the services needed to support a common set of applications based on a distributed network environment. A consensus of the Workshop was that there was really no core set of middleware services in the sense that all applications

required them. This consensus does not diminish the importance of application domain-specific middleware, or the flexibility needed in determining customized approaches. Many communities (e.g., Internet2, NGI, and other advanced Internet constituencies) may decide on their own set of common middleware services and tools; however, they should strive for interoperability whenever possible. The topics in this workshop were chosen to encourage discussion about the nature and scope of middleware per se as distinct from specific types of applications; therefore, many relevant middleware topics were not discussed.

Another consensus of the Workshop that helped provide focus was that, although middleware could be conceptualized as hierarchical, or layered, such an approach was not helpful, and indeed had been problematic and unproductive in earlier efforts.

The better approach would be to consider middleware as an unstructured, often orthogonal, collection of components (such as resources and services) that could be utilized either individually or in various subsets. This working assumption avoided extensive theological modeling discussions, and enables work to proceed on various middleware issues independently.

An important goal of the Workshop was to identify any middleware or network-related research or development that would be required to advance the state of the art to support advanced application environments, such as those being developed and pursued by NGI and Internet2. Consequently, discussion focused on those areas that had the maximum opportunity for such advances.

2.0 What is Middleware?

The Workshop participants agreed on the existence of middleware, but quickly made it clear that the definition of middleware was dependent on the subjective perspective of those trying to define it. Perhaps it was even dependent on when the question was asked, since the middleware of yesterday (e.g., Domain Name Service, Public Key Infrastructure, and Event Services) may become the fundamental network infrastructure of tomorrow. Application environment users and programmers see everything below the API as middleware. Networking gurus see anything above IP as middleware. Those working on applications, tools, and mechanisms between these two extremes see it as somewhere between TCP and the API, with some even further classifying middleware into application-specific upper middleware, generic middle middleware, and resource-specific lower middleware. The point was made repeatedly that middleware often extends beyond the "network" into the compute, storage, and other resources that the network connects. For example, a video serving application will want

to access resource discovery and allocation services not just for networks but also for the archives and computers required to serve and process the video stream. Through the application of general set theory and rough consensus, we roughly characterize middleware as those services found above the transport (i.e., over TCP/IP) layer set of services but below the application environment (i.e., below application-level APIs).

Some of the earliest conceptualizations of middleware originated with the distributed operating research of the late 1970s and early 1980s, and was further advanced by the I-WAY project at SC'95. The I-WAY linked high performance computers nation-wide over high performance networks such that the resulting environment functioned as a single high performance environment. As a consequence of that experiment, the researchers involved re-emphasized the fact that effective high performance distributed computing required distributed common computing and networking resources, including libraries and utilities for resource discovery, scheduling and monitoring, process creation, communication and data transport.

Subsequent research and development through the Globus project of such middleware resources demonstrated that their capabilities for optimizing advanced application performance in distributed domains.

In May 1997, a Next Generation Internet (NGI) workshop on NGI research areas resulted in a publication, "Research Challenges for the Next Generation Internet", which yields the following description of middleware. "Middleware can be viewed as a reusable, expandable set of services and functions that are commonly needed by many applications to function well in a networked environment". This definition could further be refined to include persistent services, such as those found within an operating system, distributed operating environments (e.g., JAVA/JINI), the network infrastructure (e.g., DNS), and transient capabilities (e.g., run time support and libraries) required to support client software on systems and hosts.

In summary, there are many views of what is middleware. The consensus of many at the workshop was that given the dynamic morphing nature of middleware, it was more important to identify some core middleware services and start working on them than it was to come to a consensus on a dictionary-like definition of the term.

Systems involving strong middleware components to support networked information discovery have also been active research areas since at least the late 1980s. For example, consider Archie or the Harvest project, to cite two examples. One could easily argue that the site logs used by Archie or the broker system and harvest agents were an important middleware tool, and additional work in this area is

urgently needed in order to improve the efficiency and scope of web-based indexing services.

"As long ago" as 1994, the Internet Architecture Board held a workshop on "Information Infrastructure for the Internet" reported in RFC 1862, which in many ways covered similar issues. Although its recommendations were summarized as follows:

- 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."

it is clear that this work is far from done.

Finally, this workshop noted that there is a close linkage between middleware as a set of standards and protocols and the infrastructure needed to make the middleware meaningful. For example, the DNS protocol would be of limited significance without the system of DNS servers, and indeed the administrative infrastructure of name registry; NTP, in order to be useful, requires the existence of time servers; newer middleware services such as naming, public key registries and certificate authorities, will require even more extensive server and administrative infrastructure in order to become both useful and usable services.

3.0 Application Perspective

From an applications perspective, the network is just another type of resource that it needs to use and manage. The set of middleware services and requirements necessary to support advanced applications are defined by a vision that includes and combines applications in areas such as: distributed computing, distributed data bases, advanced video services, teleimmersion (i.e., a capability for providing a compelling real-life experience in a virtual environment based for example on CAVE technologies), extensions with haptics, electronic commerce, distance education, interactive collaborative research, high-rate instrumentation (60 MByte/s and above sustained), including use of online scientific facilities (e.g. microscopes, telescopes, etc.), effectively managing large amounts of data, computation and information Grids, adaptable and morphing network infrastructure, proxies and agents, and electronic persistent presence (EPP). Many of these applications are "bleeding edge" with respect to currently deployed applications on the commodity Internet and hence have unique requirements. Just as the Web was an advanced application in the early 1990s, many of the application areas defined above will not become commonplace in the immediate future. However, they all possess the capability to change the way the network is used

as well as our definition of infrastructure, much as the Web and Mosaic changed it in the early 90s. A notable recent trend in networks is the increasing amount of HTTP, voice, and video traffic, and it was noted that voice and video particularly need some form of QoS and associated middleware to manage it.

A quick review of the requirements for teleimmersion highlight the requirement for multiple concurrent logical network channels, each with its own latency, jitter, burst, and bandwidth QoS; yet all being coordinated through a single middleware interface to the application. For security and efficiency those using online instruments require the ability to steer the devices and change parameters as a direct result of real-time analysis performed on the data as it is received from the instruments. Therefore, network requirements encompass high bandwidth, low latency, and security, which must all be coordinated through middleware. Large databases, archives, and digital libraries are becoming a mainstay for researchers and industry. The requirements they will place on the network and on middleware will be extensive, including support of authentication, authorization, access management, quality of service, networked information discovery and retrieval tools, naming and service location, to name only a few. They also require middleware to support collection building and self-describing data. Distributed computing environments (e.g., Globus, Condor, Legion, etc.) are quickly evolving into the computing and information Grids of the future. These Grids not only require adaptive and manageable network services but also require a sophisticated set of secure middleware capabilities to provide easy-to-use APIs to the application.

Many application practitioners were adamant that they also required the capability for "pass through" services. This refers to the ability to bypass the middleware and directly access the underlying infrastructure such as the operating system or network), even though they were eager to make use of middleware services and see more of it developed to support their own applications. In addition, authentication and access control, as well as security, are required for all of the applications mentioned above, albeit at different levels.

4.0 Exemplary Components

In an attempt to describe middleware and discuss pertinent issues relating to its development and deployment, an exemplary set of services were selected for discussion. These services were chosen to stimulate discussion and not as an attempt to define an exclusive set of middleware services. Also, it is the intent of this effort not to duplicate existing IETF efforts or those of other standards bodies (e.g., the DMTF), but rather to leverage those efforts, and indeed to

highlight areas where work was already advanced to a stage that might be approaching deployment.

5.0 Application Programming Interfaces and Signaling

Applications require the ability to explicitly request resources based on their immediate usage needs. These requests have associated network management controls and network resource implications; however, fulfillment of these requests may require multiple intermediate steps. Given the preliminary state of middleware definition, there currently is no common framework, much less a method, for an application to signal its need for a set of desired network services, including quality and priority of service as well as attendant resource requirements. However, given the utility of middleware, especially with regard to optimization for advanced applications, preliminary models for both quality and priority of service and resource management exist and continue to evolve. however, without an agreed-to framework for standards in this area, there is the risk of multiple competing standards that may further delay the deployment of a middleware-rich infrastructure. This framework should probably include signaling methods, access/admission controls, and a series of defined services and resources. In addition, it should include service levels, priority considerations, scheduling, a Service-Level-Agreement (SLA) function, and a feedback mechanism for notifying applications or systems when performance is below the SLA specification or when an application violates the SLA. Any such mechanism implies capabilities for: 1) an interaction with some type of policy implementation and enforcement, 2) dynamic assessment of available network resources, 3) policy monitoring, 4) service guarantees, 5) conflict resolution, and 6) restitution for lack of performance.

Application programmers are concerned with minimizing the interfaces that they must learn to access middleware services. Thus the unification of common services behind a single API is of great interest to middleware users. Examples of common APIs that may be achievable are:

* Environmental discovery interface, whether for discovering hardware resources, network status and capabilities, data sets, applications, remote services, or user information.
* Remote execution interface, whether for distributed metacomputing applications, or for access to a digital library presentation service, or a Java analysis service.
* Data management interface, whether for manipulating data within distributed caches, or replication of data between file systems, or archival storage of data.

* Process management interface, whether for composing data movement with remote execution, or for linking together multiple processing steps.

6.0 IETF AAA

The IETF AAA (authentication, authorization, and accounting) effort is but one of many IETF security initiatives. It depends heavily on a Public key infrastructure, which is intended to provide a framework which will support a range of trust/hierarchy environments and a range of usage environments (RFC1422 is an example of one such model).

The IETF AAA working group has recently been formed. IETF AAA working group efforts are focused on many issues pertaining to middleware, including defining processes for access/admission control and identification (process for determining a unique entity), authentication (process for validating that identity), authorization (process for determining an eligibility for resource requests/utilization) and accounting (at least to the degree that resource utilization is recorded). To some degree, AAA provides for addressing certain levels of security, but only at a preliminary level. Currently, AAA protocols exist, although not as an integrated model or standard. One consideration for AAA is to provide for various levels of granularity. Even if we don't yet have an integrated model, it is currently possible to provide for basic AAA mechanisms that can be used as a basis to support SLAs. Any type of AAA implementation requires a policy management framework, to which it must be linked. Currently, a well-formulated linking mechanism has not been defined.

Middleware AAA requirements are also driven by the distributed interoperation that can occur between middleware services. The distribution of application support across multiple autonomous systems will require self-consistent third-party mechanisms for authentication as well as data movement. Conceptually, an application may need access to data that is under control of a remote collection, to support the execution of a procedure at a third site.

The data flow needs to be directly from the collection to the execution platform for efficiency. At the same time, the procedure will need access permission to the data set while it is acting on behalf of the requestor. How the authentication is done between the remote procedure and the remote data collection entities raises significant issues related to transitivity of trust, and will require establishment of a trust policy for third-party mechanisms. This is exacerbated when a collection of entities, such as is required for visualization applications, is involved.

7.0 Policy

The IETF Policy Framework working group is addressing a policy framework definition language, a policy architecture model, policy terminology and, specifically, a policy model that can be used for signaled as well as provisioned QoS. The policy meta-model links high-level business requirements, such as those that can be specified in an SLA, to low-level device implementation mechanisms, ranging from specific access control and management of services, objects and other resources to configuration of mechanisms necessary to provide a given service.

Polices are an integral component of all middleware services, and will be found within most middleware services in one form or another. Policies are often represented as an "if condition then action" tuple. Policies can be both complex and numerous; therefore, policy management services must be able to identify and resolve policy conflicts. They also need to support both static (i.e. loaded at boot time via a configuration file) and dynamic (i.e. the configuration of a policy enforcing device may change based on an event) modes.

A generalized policy management architecture (as suggested by the IETF policy architecture draft) includes a policy management service, a dedicated policy repository, at least one policy decision point (PDP), and at least one policy enforcement point (PEP). The policy management service supports the specification, editing, and administration of policy, through a graphical user interface as well as programmatically. The policy repository provides storage and retrieval of policies as well as policy components. These policy components contain definitional information, and may be used to build more complex policies, or may be used as part of the policy decision and/or enforcement process. The PDP (e.g. resource manager, such as a bandwidth broker or an intra-domain policy server) is responsible for handling events and making decisions based on those events (e.g., at time x do y) and updating the PEP configuration appropriately. In addition, it may be responsible for providing the initial configuration of the PEP. The PEP (e.g., router, firewall or host) enforces policy based on the "if condition then action" rule sets it has received from the PDP.

Policy information may be communicated from the PDP to the PEP through a variety of protocols, such as COPS or DIAMETER. A proxy may be used to translate information contained in these protocols to forms that devices can consume (e.g., command line interface commands or SNMP sets). Additional information, contained in Policy Information Bases (PIBs), may also be used to translate from an intermediate specification to specific functions and capabilities of

a device. For example, a policy may specify "if source IP address is 198.10.20.132, then remark traffic with a DSCP of 5". The PIB would be used to translate the device-specific meaning of the conditioning specified by the DiffServ code point of 5 (e.g., a specific set of queue and threshold settings).

Policy requires AAA functions, not only for access control, but also to establish the trust relationships that will enable distributed policy interactions. PDPs may require the requesting end systems and applications to be authenticated before the PDP will honor any requests. The PDP and PEP must be authenticated to each other to reduce the probability of spoofing. This will be true whichever protocol is utilized for supporting communications between these entities. Audit trails are essential for all of these transactions. In addition, trust management policies will need to be developed as well as the supporting middleware mechanisms to enable inter-domain policy negotiation.

Ultimately, many policy processes link entities to resources, And therefore require interactions with entity identification mechanisms, resource identification mechanisms, and allocation mechanisms. The distributed computing community has already started efforts developing policy definition languages and systems. Globus uses its Resource Services Language (RSL) to define the resources and policies associated with them. Condor uses a matchmaking bidding technique to match those providing and those acquiring services. Similarly, the IETF has several policy definition languages in varying stages of development, including RPSL, RPCL, SPSL, PFDL, PAX, and Keynote. Ultimately, these efforts should be merged into a single specification (or at least a smaller group of specifications) to enable distributed computing applications to be able to effectively communicate and utilize network resources and services.

Directories play a crucial role in policy systems. Directories are ideally suited for storing and retrieving policy information, due to their exceptionally high read rates, ability to intelligently replicate all or part of their information, per-attribute access control, and use of containment. To this end, the IETF Policy Framework working group (in conjunction with the DMTF) is developing a core information model and LDAP schema that can be used to represent policy information that applications can use. This core model is used to provide common representation and structure of policy information. Applications can then subclass all or part of the classes in this core schema to meet their own specific needs, while retaining the ability to communicate and interoperate with each other.

8.0 Directories

Directories are critical resource components that provide support to many other elements in the middleware environment, especially policy. As network-based environment evolves, it will no longer be viable to encode policy information directly into each individual application. The prevailing model in use today is for each application to store its view of a device's data (e.g., configuration) in its own private data store.These data include relevant information concerning network resources and services as well as clients wanting to use those resources (e.g., people, processes, and applications). The same resource (or aspects of that resource, such as its physical vs. logical characteristics) may be represented in several data stores. Even if the device is modeled the same way in each data store, each application only has access to its own data. This leads to duplication of data and data synchronization problems.

The promise of technologies like CIM and DEN is to enable each application to store data describing the resources that they manage in a single directory using a common format and access protocol. This results in the data describing the resource being represented only once. Defining a logically centralized common repository, where resources and services are represented in a common way, enables applications of different types to utilize and share information about resources and services that they use.

Not only does this solve the data duplication and synchronization problems, it also provides inherent extensibility in describing the characteristics of an object - a single entity can be represented by multiple directory objects, each representing a different aspect of the entity. Different applications can be responsible for managing the different objects that together make up a higher-level object, even if the applications themselves can not communicate with each other. This enables these applications to effectively share and reuse data. This provides significant benefits for users and applications. In the short term, users and applications will benefit from having all of the data in one place. In the long term, users and applications will be able to take advantage of data managed by other applications.

Directories are key to supporting advanced network-based application environments. Directory purists say that the directory is not middleware; rather, it is a dumb storage device that is made into an intelligent repository by encapsulating it within middleware. Although a directory associates attributes with objects, what makes it different from a database are four key things:

- directory objects are essentially independent of each other, whereas database objects are related to each other (sometimes in very complex ways)
- directories organize their information using the notion of containment, which is not naturally implemented in databases
- directory objects can have specific access controls assigned to an object and even attributes of an object
- directories, unlike databases, are optimized to perform a high number of reads vs. writes.

Directories use a common core schema, supporting a common set of syntaxes and matching rules, that defines the characteristics of their data. This enables a common access protocol to be used to store and retrieve data.

Containment can be used for many purposes, including associating roles with objects. This is critical in order to support a real world environment, where people and elements may assume different roles based on time or other context.Containment may also be used to provide different naming scopes for a given set of data.

Directories use attribute inheritance - subclasses inherit the attributes of their superclasses. This enables one to define generalized access control at a container (e.g., a group) and then refine the access control on an individual basis for objects that are inside that container (e.g., different objects have different access privileges).

Currently, directories are used mostly to represent people, servers, printers, and other similar objects. CIM, DEN, and other similar efforts have encouraged directories to be used to contain common objects in a managed environment. For networked applications, this enables clients of the network (e.g., users and applications) to be bound to services available in the network in a transparent manner. The "Grid" community is making extensive use of directory services for this purpose, using them to maintain information about the structure and state of not only networks but also computers, storage systems, software, and people. The DMTF is using directories to contain CIM and DEN information, which enables a common information model to be applied to objects in a managed environment. The IETF is using directories for many different purposes, not the least of which is to contain common policy information for users and applications of an environment, as well as services and configuration information of network devices.

CIM and DEN are conceptual information models for describing the management of entities ranging from network elements to protocols to hosts and services. CIM and DEN are platform- and technology-independent. DEN is an extension of CIM that, among other things, describes how to map CIM data into a form usable by LDAP.

The CIM Specification describes the meta schema, information model, language, naming, and mapping techniques to other management models, such as SNMP MIBs and DMTF MIFs. DEN provides a good start on a model that addresses the management of the network and its elements; DEN is an extension of CIM to include the management of networks as a whole and not just the individual elements. DEN addresses the requirements for abstracting a complex entity, such as a router, into multiple components that can be used to manage individual aspects of that complex entity. The DEN information model, like CIM, incorporates both static and dynamic information. DEN provides a mapping to directories for the storage and retrieval of data. DEN will also rely heavily on the use of AAA services in order to maintain the integrity of the directory and its policies as well as to manage the distribution of policies among the policy repositories, PDPs and PEPs. Resource managers and applications will also rely heavily on directories for the storage of policy and security information necessary for the management and allocation of resources.

Since much of the information associated with a person, agent or element is stored in a directory, and access to that information will be controlled with appropriate security mechanisms, many voiced the need for a single user/process sign on.

Future advanced applications (e.g., NGI, Internet2, PACI, Grids) may require a variety of PDPs to manage a variety of resource types (i.e., QOS, security, etc.). In this case, a general model would have to be developed that defines the protocols and mechanisms used by cooperating resource managers (i.e., PDPs) of different domains and different genres of resource (i.e., network, security, storage, proxy agents, online facility, etc.). For policies to be implemented in a coherent fashion, it is necessary to have a mechanism that discovers and tracks resources and utilization.

There is an architectural issue of central importance, which has most recently surfaced in the directory area. Many applications, and many middleware components, need what is essentially a highly scalable, distributed database service. In other words, people want to take the best of what directories and databases have to offer. This would result in a distributed, replicated database that can use containment to effectively organize and scope its information. It would be able to have exceptional read response time, and also offer transactional and relational integrity. It would support simple and complex

queries. Such a service has never been defined as a middleware component; the complexities involved in specifying and implementing such a service are certainly formidable. However, in the absence of such a general service, many middleware components have attempted to use the closest service available, which is deployed - historically first using DNS, and more recently, directory services.

It will be important to clarify the limitations of the appropriate use of directory services, and to consider whether a more general data storage and retrieval service may be required, or whether directory services can be seamlessly integrated (from the point-of-view of the applications using them) with other forms of storage and retrieval (such as relational databases) in order to provide an integrated directory service with these capabilities.

9.0 Resource Management

Policy implementation processes need to be linked to Resource Managers in a more sophisticated way than those that currently exist. Such processes must be dynamic, and able to reflect changes in their environment (e.g., adjust the quality of service provided to an application based on environmental changes, such as congestion or new users with higher priorities logging onto the system). We need to determine how different types of resource managers learn about one another and locate each other - as well as deal with associated cross-domain security issues. Another aspect of this problem is developing a resource definition language that can describe the individual elements of the resource being utilized, whether that is a network, processor, agent, memory or storage. This will require developing an appropriate metadata representation and underlying meta schema that can be applied to multiple resource types.

Some models of resource managers are currently being used to provide for the management of distributed computing and Grid environments (e.g., Condor, Globus, and Legion). These resource managers provide languages, clients, and servers to support accessing various types of distributed computing resources (e.g. processors, memory, storage and network access). There is a broad interest in the distributed and parallel computing communities in developing an automated access control architecture, using policies, to support the evolving IETF differentiated services architecture. However, this work has not yet been incorporated into any IETF working group charter. The term "bandwidth broker" has been used to refer to the agents that will implement this functionality through network resource management, policy control, and automated edge device configuration. The IETF Policy Framework working group is currently working on a policy architecture framework, information model, and policy definition language that is targeted initially at policy management within a

single domain. However, this work is fundamental in defining inter-domain policy management issues, such as those that are required in implementing a network resource manager / bandwidth broker. Many resource managers being deployed today rely on directory services for storing policy information as well as X.509 for certificate-based authentication and authorization to these resources. Middleware will be required to translate the needs of distributed and parallel computing applications within and across different policy domains. It is crucial that a standard means for representing and using resource management be developed.

Advance reservation of resources, as well as dynamic requests for resources, is a crucial aspect of any resource management system. Advance reservations are more of a policy issue than a provisioning issue; however, the mechanisms for exchanging and propagating such requests between resource managers located within different administrative domains is a currently unsolved problem that needs to be addressed. In addition, it is important to address the issue of possible deadlock and/or the inefficient use of resources (i.e., the time period between a request, or set of requests, being initiated and honored and resources being allocated). There is also a need for rendezvous management in resource allocation services, where an application must gather resource reservations involving multiple sites and services.

A mesh of cooperating resource managers, which interact with each other using standards based protocols (e.g. COPS), could be the model for a resource management infrastructure. Each of these may manage different sets of resources. For example, one may be a bandwidth broker that only manages network bandwidth, while another may be a general-purpose resource manager that manages security, IP address allocation, storage, processors, agents, and other network resources. There are already plans for middleware resource managers that not only allocate the resources but also manage the composition of a group of services that may include security services, billing services, shaping of multimedia composite images, etc.). Another form of resource manager may provide mapping between a set of related services (i.e., mapping an IP based RSVP request to an ATM SVC, as was demonstrated in a pilot project on the vBNS).

Resource managers depend on the use of locator services to find other resource managers as well as to locate the AAA server(s) for the requestor and the associated directories containing applicable policy information. They may also need to query the network to determine if a policy request for bandwidth can be satisfied. It is essential that these (and other) different uses of resource management be integrated to provide an end-to-end service for applications and users alike.

10.0 Networked Information Discovery and Retrieval Services

There are a wide range of middleware services broadly related to the discovery and retrieval of networked information. Because such a broad range of applications (and not just high-performance, distributed, or parallel applications) requires these services, this area is under very active development and new requirements are constantly emerging.

Perhaps the most basic service in this area is persistent naming and location services (and infrastructure) that can resolve names to locations (i.e., URLs). The IETF has done considerable work in defining a syntax for Uniform Resource Identifiers (URIs), which are intended to be persistent name spaces administered by a wide range of agencies. URIs are resolved to URLs using resolver services; there are a number of different proposals for such resolver services, and some implementations exist such as the CNRI Handler Service. Many organizations are beginning to establish and manage URI namespaces, notably the publishing community with their Digital Object Identifier (DOI). however, there are many unresolved questions, such as how to most effectively deal with the situation where the resource named by a URI exists in multiple places on the network (e.g., find the "closest" mirror in terms of network connectivity and resource availability). There is a need for an extensive set of infrastructure around resolvers, including how resources are registered and identifiers are assigned, the ongoing management of data about the current location of resources that are identified by a specific URI, and the operation of sets of resolvers for various name spaces. Finally, given a URI, one needs to locate the resolver services that are connected with that namespace; the IETF has done initial work on resolution service location for URI namespaces.

URIs are intended to be processed primarily by machines; they are not intended to necessarily be easy to remember, though they are intended to be robust under transcription (not sensitive to whitespace, for example). More recently, the IETF has begun work on defining requirements for human friendly identifier systems that might be used to register and resolve mnemonic names.

Another set of issues revolves around various types of metadata - descriptive, ratings, provenance, rights management, and the like, that may be associated with objects on the network. The Resource Description Framework (RDF) from the Worldwide Web Consortium (W3C) provides a syntax for attaching such descriptions to network objects and for encoding the descriptions; additional middleware work is needed to locate metadata associated with objects that may be stored in repositories, and to retrieve such metadata. Validation of metadata is a key issue, and both IETF and W3C are working on XML

canonicalization algorithms that can be used in conjunction with public key infrastructure to sign metadata assertions. However, such an approach implies a complex set of trust relationships and hierarchies that will need to be managed, and policies that will need to be specified for the use of these trust relationships in retrieval.

There is specific work going on in defining various types of metadata for applications such as rights management; ultimately this will imply the development of middleware services. It will also impact the use of directory, database, and similar services in the storage, access, and retrieval of this information. Similarly, there will be a need for services to connect descriptive metadata and identifiers (URNs).

(See also the NSF/ERCIM report on metadata research issues at
http://www.ercim.org/publication/ws-proceedings/EU-NSF/metadata.html
http://www.ercim.org/publication/ws-proceedings/EU-NSF/metadata.ps
http://www.ercim.org/publication/ws-proceedings/EU-NSF/metadata.pdf

Finally, there is a need for a set of middleware services which build upon the research work already integrated into services such as Archie and Harvest. These services permit the efficient extraction of metadata about the contents of network information objects and services without necessarily retrieving and inspecting those services. This includes the ability to dispatch "indexing agents" or "knowbots" that can run at a site to compute such indexing, under appropriate security and authentication constraints. In addition, a set of "push-based" broker services which aggregate, filter and collect metadata from multiple sites and provide them to interested applications are also required. Such services can provide a massive performance, quality, comprehensiveness and timeliness improvement for today’s webcrawler-based indexing services.

11.0 Network QoS

As noted earlier, applications may need to explicitly request resources available in the network to meet their requirements for certain types of communication, or in order to provide service with an appropriate guarantee of one or metrics, such as bandwidth, jitter, latency, and loss. One type of request that has been the focus of much effort recently is for services beyond best effort, particularly with respect to services running over IP. This is particularly important for the advanced applications noted previously (e.g., visualization and teleimmersion) as well as the emerging importance of voice and video, especially voice and video operating with lower bandwidth or voice and video co-mingled with data. One perspective on this issue is to consider the effect of multiple drops

in a single RTT, which is catastrophic for TCP applications but may be of no special significance for real-time traffic. Providing for improved services can be accomplished through a variety of quality of service (QoS) and class of service (CoS) mechanisms. The first IETF model was the Integrated Services (IntServ) model, which used RSVP as the signaling mechanism. Since this model requires state in every router for every session and to manage the traffic flows, it is generally recognized to have scaling limits. However, it is very appropriate for certain situations.

Differentiated Services, or DiffServ, grew out of a reaction against the perceived scalability problems with the IETF IntServ model. DiffServ is an architecture for implementing scalable service differentiation in the Internet. Scalability is achieved by aggregating traffic through the use of IP-layer packet marking. Packets are classified and marked to receive a particular per-hop forwarding behavior on nodes along their path. Sophisticated classification, marking, policing, and shaping operations need only be implemented at network boundaries or hosts. Network resources are allocated to traffic streams by service provisioning policies which govern how traffic is marked and conditioned upon entry to a differentiated services-capable network, and how that traffic is forwarded within that network. These simple PHBs are combined with a much larger number of policing policies enforced at the network edge to provide a broad and flexible range of services, without requiring state or complex forwarding decisions to be performed in the core and distribution layers.

Recently, the idea of "tunneling" RSVP over a DiffServ-capable network has generated significant interest. This attempts to combine the best features of both IntServ and DiffServ while mitigating the disadvantages of each. This in turn has led the IETF to study ways to ensure that Differv and Inteserv can not only coexist, but are also interoperable.

The practical realization of either or both architectures depends on many middleware components, some of which are described in this document. The workshop discussion mainly focused on DiffServ mechanisms and on what effect such mechanisms would have on middleware and its ability to monitor and manage the network infrastructure for the benefit of the applications. Both IntServ and DiffServ only fully make sense if linked to a policy mechanism. This mechanism must be able to make policy decisions, detect and resolve conflicts in policies, and enforce and monitor policies.

Workshop participants almost unanimously agreed that they also required a scalable inter-domain resource manager (e.g., a bandwidth broker). Currently, if an RSVP session is run, each router along a

path becomes involved, with flow policing at each hop. Bandwidth Broker models include the bandwidth broker, a policy decision point (which makes admission control and policy decisions) and the policy enforcement points (i.e., edge routers) which provide for policing at the first hop and for remarking aggregate flows so that subsequent routers need only deal with the aggregate flows.

IETF protocols that could be used to implement a Bandwidth Broker model (e.g., COPS, Diameter, and others) were also discussed. The Diameter protocol is interesting in this context, because it provides set up mechanisms for basic network resource allocations and reallocations, as well as optional allocations.- All of these can be used for various types of bandwidth broker implementations, including those directed at QoS, using RSVP type information. Diameter currently does not provide path information, but instead relies on network pathway information established at ingress and egress nodes. However, the status of Diameter is still open in the IETF.

COPS was initially developed as a mechanism for establishing RSVP policy within a domain and remains intra-domain centric. It is a useful intra-domain mechanism for allocating bandwidth resources within a policy context. Work is now being conducted to use COPS for establishing policy associated with a DiffServ-capable network. COPS is designed to facilitate communication between the PDP and the PEP, carrying policy decisions and other information.

To implement any type of Bandwidth Broker model, it is necessary to establish a mechanism for policy exchanges. The Internet2's Qbone working group is currently working to define a prototype inter-domain bandwidth broker signaling protocol. This work is being coordinated with IETF efforts.

Another mechanism is required for traffic shaping and SLA policing and enforcement. One mechanism is fair queuing in its various forms, which has been described as TDM emulation without the time and space components. Techniques have been used for several years for fair queuing for low speed lines. For DS-3 with 40 byte packets and OC-3c speeds with 200-byte packets, weighted fair queuing uses a deficit round-robin algorithm that allows it to scale. It is capable of flow discrimination based on stochastically hashing the flows. An additional expansion of this technique is to preface this technique with class indicators. Currently, classification techniques are based on IP precedence. However, classification will soon be achieved in many routers using Diffserv code points (DSCPs) to specify the type of conditioning to be applied. The complete requirements of policing for DiffServ implementations, e.g., via bandwidth brokers, have not yet been fully explored or defined.

Network monitoring capabilities (i.e., querying the network for state information on a micro and macro level) that support middleware and application services were identified as a core requirement. In fact, a network instrumentation and measurement infrastructure, upon which a set of intelligent network management middleware services can be built, is absolutely critical.

Current mechanisms (e.g. ICMP, SNMP) were not deemed robust enough for middleware and applications developers to determine the state of the network, or to verify that they were receiving the specific type of treatment they had requested. This was judged especially true of a network providing QoS or CoS. Indeed, it is not at all clear that SNMP, for example, is even the right architectural model for middleware to use to enable applications to determine the state of the network. Other capabilities, such as OcxMon, RTFM, new MIBs, and active measurement techniques (e.g., IPPM one-way delay metrics) need to be made available to middleware services and applications.

The provisioning of differentiated services takes the Internet one step away from its "dumb" best effort status. As the complexity of the network increases (e.g. VPNs, QoS, CoS, VoIP, etc.), more attention must be paid to providing the end-user/customer or network administrator with the tools they require to securely and dynamically manage an adaptable network infrastructure. Differentiated services means that theoretically some traffic gets better service than other traffic; subsequently, one can expect to pay for better service, which means that accounting and billing services will be one of the important middleware core components that others will rely upon. The model and protocols necessary to accomplish this are not developed yet.

12.0 Authentication, Authorization, and Accounting

The IETF’s AAA working group is focusing on the requirements for supporting authentication, authorization, accounting, and auditing of access to and services provided by network resource managers (e.g., bandwidth brokers). These processes constitute an important security infrastructure that will be relied upon by middleware and applications. However, these components are only basic security components. A public key infrastructure (PKI) was identified as a crucial security service infrastructure component. For example, the PKI will be required to support the transitivity of authentication, authorization, and access control and, where appropriate, accounting and billing. It was noted that, except for issues dealing with group security and possibly more efficient and simple management, there are no real technical challenges preventing the wide scale deployment of a PKI support structure at this time. Instead, the main obstacles to overcome are mostly political and economic in nature. However,

additional middleware may be required to better facilitate a PKI. That being said, some people believe that we do have some large technical security challenges, revocation lists and security with respect to changing group memberships being two examples.

Middleware and security support is also required for newer applications (e.g., proxy agents that would act on a process or application's behalf and gather the necessary certificates for access and using resources). A particularly difficult example is remote collaboration. Accessing a particular resource may require a user and/or application to gather certificates from more than one policy-controlling agent. It is also true that an entity may have various identities that are dependent on the task they are performing (usage or role based) or the context of the application. In order for the PKI to become truly functional on a ubiquitous level, there needs to exist a set of independent signing authorities that can vouch for the top-level certificate authorities.

There are also higher-level middleware services which will build on public key infrastructure, notary services and provenance verification. As we move from a relatively dumb network (e.g. best effort IP) to an Internet with embedded intelligence (e.g., DiffServ, IntServ, bandwidth brokers, directory-enabled networks, etc.), the secure exchange of information will become even more important. In addition, as we start to provide differentiated services, accounting and statistics gathering will become much more important. We also need to provide for the integrity and security of collecting, analyzing, and transporting network management and monitoring information. And the issues of data privacy and integrity, along with addressing denial of service and non-repudiation, cannot be ignored.

13.0 Network Management, Performance, and Operations

Network management capabilities were identified as being paramount to the success of middleware deployment, and subsequently to the success of the application. Many of the issues addressed here are not part of standard NOC operations. In a more complex world of QoS, CoS, and micro prioritization, reactions to network failures must be handled differently than current procedures. Allocations are more dynamic, especially additions, deletions, and changes with additional sets of requirements, such as priorities and new types of inter-domain interactions. These will inevitably increase the complexity of network management.

There are many microscopic and macroscopic network management projects focusing on making both active and passive network statistics and information available to end-users. Current visual

debugging and analysis capabilities (e.g., those developed by NLANR/CAIDA) are crucial tools for network administrators and designers for understanding their networks. In addition, current network management techniques and mechanisms, which were designed for network designers and managers, need to be adapted to provide a dynamic and relevant set of information to the middleware or application service software. This will allow the programs to dynamically adapt to the changing state of the network infrastructure while ensuring the integrity and security of the network and other resources.

Another aspect of network management that has not received the necessary attention, is the need for modeling and analysis tools for network and middleware designers. CIM and DEN show great promise in providing a common framework for modeling the management of network elements and services as well as users, applications, and other resources of the network. Undoubtedly, middleware designers will place new requirements on CIM and DEN that will cause these approaches to evolve.

14.0 Middleware to support multicast applications

IP multicast - that is, the routing and forwarding of mutlicast packets in an IP-based network, is in the view of the workshop part of the basic network infrastructure. The Internet Group Multicast Protocol, which manages the joining and leaving of multicast groups, could also be considered a basic network service. However, there is a tremendous need for middleware services to make multicast useable for various applications, much like TCP played a key role in making IP applications useable. Specifically, one might reasonably want middleware services to provide authenticated control of multicast services. Examples of these services include the creation and joining of multicast groups, multicast address management, multicast channel directories (there has already been considerable work in this area), various forms of reliable multicast services (this has been an IRTF research area), and to secure multicast groups through various cryptographic strategies. In addition, because of the large impact that multicast can have on a network, multicast management middleware services, particularly in conjunction with QoS, will be needed, as will services to link together multicasting within various networks that do not directly interchange multicast routing information. It should be noted, however, that several security issues with multicast, especially groups with dynamic membership policies, still need to be resolved.

15.0 Java and Jini

Java was chosen as an example of a heterogeneous runtime support system for the sake of discussion as to whether it could be qualified as a development language particularly suitable for the development of middleware. The consensus was that the Java language and compilers are important in the current distributed model of the Internet and for the support of middleware (i.e., middleware written using Java). Also, a virtual Java machine located on a system can be considered middleware as much as any operating system or network operating systems would be considered middleware. Jini middleware technology not only defines a set of protocols for discovery, join, and lookup, but also a leasing and transaction mechanism to provide resilience in a dynamic networked environment. Java and Jini will be dependent on a functioning PKI, especially for signed applets. That being said, there are security concerns with both Java and Jini that need to be addressed, such as allowing the downloading of applets and servlets.

16.0 Security Considerations

This document is a report of a workshop in which security was a common theme, as can be seen by the references to security through out the document; but the workshop did not reach any specific recommendations for new security-related terminology.

17.0 Summary

Middleware may have components and services that only exist in the persistent infrastructure, but it will also have components that enable and support end-to-end (i.e. application to application or host to host) interaction across multiple autonomous administrative domains. A set of core persistent middleware services is required to support the development of a richer set of middleware services which can be aggregated or upon which applications will be based (e.g., an onion or layered model). This set of core middleware services will help applications leverage the services and capabilities of the underlying network infrastructure, along with enabling applications to adjust in changes to the network. The particular set of such services utilized by an application or process will be a function of the requirements of the application field or affinity group (e.g., network management or high energy physics applications) wishing to utilize the network or distributed data/computation infrastructure. This document discusses some of the basic and core middleware services, which include, but are not limited to: directories, name/address resolution services, security services (i.e., authentication, authorization, accounting, and access control), network management, network monitoring, time servers, and accounting. Network level capabilities, such as multicast and DiffServ, are not

classified as middleware; rather, they are enabling infrastructure services upon which middleware will be built or which middleware may use and manage. A second level of important middleware services, which builds upon these core set of services, may include accounting/billing, resource managers, single sign-on services, globally unique names, metadata servers, and locators.

A recognized goal is to provide a set of middleware services that enable access to and management of the underlying network infrastructure and support applications wishing to make use of that network-based infrastructure. It appears necessary to agree to a framework of services for the support, provisioning and operations, and management of the network. Today, we have piecemeal activities already being pursued in various standards organizations. These include efforts in the IETF and DMTF (e.g., AAA, Policy Framework, DiffServ, DEN, CIM, etc.), as well as in the advanced application environments (e.g., Grid Forum, the PACIs, NGI, Internet2, etc.). Both of these efforts require the integration and management of many infrastructure components, not just networks; however, we have no overall framework that pulls all of these together, or a mechanism to coordinate all of these activities. We are just embarking on the development of a rich plan of middleware services. Consequently, we have a lot of work yet to be done. For instance, as we move into an electronic persistent presence (EPP) environment where multiple instances of an identity or person (or even their proxy agents) are supported, we will require enhanced locator and brokering services. The directory (e.g., DNS or X.500) and locator services of today may not be appropriate for this task.

One goal of the workshop was to identify research and development areas in middleware that federal agencies and industry may choose to support. The workshop highlighted a few areas that may benefit from additional R&D support. These areas include, but are not limited to:

- inter-domain resource management architecture and protocols (e.g., inter-domain bandwidth brokers)
- resource languages that describe and enable the management of a wide variety of resources (e.g., networks, data bases, storage, online facilities, etc.
- avoiding deadlock and ensuring efficiency with resource managers
- network management tools and APIs that provide macroscopic and microscopic real-time infrastructure
- information to middleware services and applications (not just MIBs and SNMP access)
- domain and inter-domain accounting and billing
- monitoring and verification services of contracted infrastructure services
- enhanced locators that can locate resources and resource managers

- cross administrative policy negotiation and authentication
- middleware bypass (i.e. access to raw system or network resources metadata (i.e., data that is used to describe data found in directories or exchanged between services such as resource managers, PDPs, PEPs, directories, accounting and billing services, etc.)
- middleware support for mobile or nomadic use
- support for availability of resources (i.e. replication and load balancing

This workshop was just one small step in identifying relevant middleware topics, technologies and players. Even though this workshop did not arrive at a consensual definition of middleware, it did identify the need for additional work. Specifically, further work is needed to identify and qualify middleware services for specific affinity groups (e.g. Internet2, Education, the PACIs, Grids, etc.) as well as to define a macroscopic framework that incorporates the middleware work of the IETF, DMTF and other relevant organizations such as the Grid Forum.

18.0 Participants

Deb Agarwal <deba@george.lbl.gov>, Bob Aiken <raiken@cisco.com>, Guy Almes <almes@internet2.edu>, Chase Bailey <chase@cisco.com>, Fred Baker <fred@cisco.com>, Pete Beckman <beckman@lanl.gov>, Javad Boroumand <jborouma@nsf.gov>, Scott Bradner <sob@harvard.edu>, George Brett <ghbrett@mindspring.com>, Rich Carlson <racarlson@anl.gov>, Brian Carpenter <bcarpent@uk.ibm.com>, Charlie Catlett <catlett@ncsa.uiuc.edu>, Bill Cheng <wtcheng@us.ibm.com>, Kim Claffy <kc@caida.org>, Bill Decker <Wdecker@nsf.gov>, Christine Falsetti <cfalsetti@arc.nasa.gov>, Ian Foster <foster@mcs.anl.gov>, Andrew Grimshaw <grimshaw@cs.virginia.edu>, Ed Grossman <egrossma@ncsa.uiuc.edu>, Ted Hanss <ted@internet2.edu>, Ron Hutchins <ron@oit.gatech.edu>, Larry Jackson <jackson@ncsa.uiuc.edu>, Bill Johnston <Wejohnston@lbl.gov>, Juerg von Kaenel <jvk@us.ibm.com>, Miron Livny <miron@cs.wisc.edu>, Cliff Lynch <cliff@cni.org>, Joel Mambretti <j-mambretti@nwu.edu>, Reagan Moore <moore@sdsc.edu>, Klara Nahstedt <klara@cs.uiuc.edu>, Mike Nelson <mrn@us.ibm.com>, Bill Nitzberg <nitzberg@nas.nasa.gov>, Hilarie Orman <ho@darpa.mil>, John Schnizlein <jschnizl@cisco.com>, Rick Stevens <stevens@mcs.anl.gov>, John Strassner <johns@cisco.com>, Ben Teitelbaum <ben@advanced.org>, George Vanecek <g.vanecek@att.com>, Ken Klingenstein <Ken.Klingenstein@Colorado.EDU>, Arvind Krishna <akrishna@us.ibm.com>, Dilip Kandlur <kandlur@us.ibm.com

19.0 URLs/references

Please see http://www.mcs.anl.gov/middleware98 for copies of the slides presented at the workshop as well as a list of related URLs on applications, middleware and network services.

20.0 Authors' Addresses

Editor: Bob Aiken
EMail: raiken@cisco.com

Authors:

Bob Aiken
Cisco Systems, Inc.
6519 Debold Rd.
Sabillasville, Md. 21780 USA

Phone: +1 301 271 2919
EMail: raiken@cisco.com

John Strassner
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134

Phone: +1 408 527 1069
EMail: johns@cisco.com

Brian E. Carpenter
IBM United Kingdom Laboratories
MP 185, Hursley Park
Winchester, Hampshire SO21 2JN, UK

EMail: brian@hursley.ibm.com

Ian Foster
Argonne National Laboratory
The University of Chicago
Argonne, IL 60439 USA

Phone: +1 630 252 4619
EMail: foster@mcs.anl.gov

Clifford Lynch
Coalition for Networked Information
21 Dupont Circle
Washington, DC 20036

Phone: +1 202 296 5098
EMail: cliff@cni.org

Joe Mambretti
International Center for Advanced Internet Research
1890 Maple, Suite 150
Northwestern University, Evanston, Illinois 60201

Phone: +1 847 467 3911
EMail: j-mambretti@nwu.edu

Reagan Moore
University of California, San Diego
NPACI/SDSC, MC 0505
9500 Gilman Drive
La Jolla, CA 92093-0505 USA

EMail: moore@sdsc.edu

Benjamin Teitelbaum
Advanced Networks & Services, Inc.

EMail: ben@internet2.edu

21.0 Full Copyright Statement

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.