Grid Information Services for Distributed Resource Sharing

Karl Czajkowski[*] Steven Fitzgerald[†] Ian Foster[‡§] Carl Kesselman[*]

[*] Information Sciences Institute
University of Southern California
Marina del Rey, CA 90292

[†] Department of Computer Science
California State University, Northridge
Northridge, CA 91330

[‡] Department of Computer Science
The University of Chicago
Chicago, IL 60657

[§] Mathematics and Computer Science Division
Argonne National Laboratory
Argonne, IL 60439

Abstract

Grid technologies enable large-scale sharing of resources within formal or informal consortia of individuals and/or institutions: what are sometimes called virtual organizations. In these settings, the discovery, characterization, and monitoring of resources, services, and computations are challenging problems due to the considerable diversity, large numbers, dynamic behavior, and geographical distribution of the entities in which a user might be interested. Consequently, information services *are a vital part of any Grid software infrastructure, providing fundamental mechanisms for discovery and monitoring, and hence for planning and adapting application behavior.*

We present here an information services architecture that addresses performance, security, scalability, and robustness requirements. Our architecture defines simple low-level enquiry *and* registration *protocols that make it easy to incorporate individual entities into various information structures, such as* aggregate directories *that support a variety of different query languages and discovery strategies. These protocols can also be combined with other Grid protocols to construct additional higher-level services and capabilities such as brokering, monitoring, fault detection, and troubleshooting. Our architecture has been implemented as MDS-2, which forms part of the Globus Grid toolkit and has been widely deployed and applied.*

1. Introduction

Grid computing technologies enable wide-spread sharing and coordinated use of networked resources [15]. Sharing relationships may be static and long-lived—e.g., among the major resource centers of a company or university—or highly dynamic: e.g., among the evolving membership of a scientific collaboration [17]. In either case, the fact that users typically have little or no knowledge of the resources contributed by participants in the "virtual organization" (VO) poses a significant obstacle to their use. For this reason, information services designed to support the initial discovery and ongoing monitoring of the existence and characteristics of resources, services, computations, and other entities are a vital part of a Grid system [13].

Such information services find uses in a variety of Grid scenarios. The following examples illustrate but do not exhaust the range of applications that rely on information services, and the variety of information sources and information access and management methods that are associated with these applications [13, 30, 36, 11, 34].

A *service discovery service* operated by a community records the identity and essential characteristics of the "services" that are available to community members. Such a discovery service might enable a physicist to determine that a new university that has just joined his consortium has 100 new CPUs available for approved use. Here, information sources are relatively static and the information itself relates primarily to availability.

A *superscheduler* routes computational requests to the "best" available computer in a Grid containing multiple high-end computers, where "best" can encompass issues of architecture, installed software, performance, availability, and policy. Here, information sources are computers, and information can include both relatively static information such as system configuration (architecture, OS version, access policy) and more dynamic information such as instantaneous load and predictions of future availability [40, 10].

A *replica selection service* within a data grid responds to requests for the "best" copy of files that are replicated on multiple storage systems. Here, information sources can once again include system configuration, instantaneous performance, and predictions, but for storage systems and networks rather than computers.

An *application adaptation agent* monitors both a running application and external resource availability and modifies application behavior (e.g., reduces accuracy, changes algorithms) and/or its resource consumption (e.g., migrates to other resources) if, due to changes in resource status or application behavior, these changes are thought likely to improve performance. Information sources include various components of both the application and the underlying execution environment.

A *troubleshooting* service monitors Grid resources, looking for anomalous behaviors such as excessive load or extended failure of critical services. Here, the information sources can be arbitrary; the information that is of interest is determined by troubleshooter heuristics and can be highly dynamic.

A *performance diagnosis* tool, invoked by a user when anomalous behavior is detected, discovers what information sources are associated with an application and its resources (e.g., application sensors, network sensors, historical information sources) and accesses these information sources as it seeks to diagnose the poor performance.

These examples differ greatly in terms of the nature of the information sources (physical characteristics, location, dynamicity, sensitivity), the demands placed on those information sources (e.g., access rates, query vs. subscribe, accuracy demands), and the ways in which information is employed (e.g., discovery, brokering, monitoring, diagnosis, adaptation). Nevertheless, in each case we see a similar structure: one or more consumers (users or programs) wish to obtain information from one or more producers. Mechanisms are required that allow consumers and producers to discover each other and that subsequently allow information to flow between producers and consumers. *We show in the following that it is feasible to treat these different cases within a single, consistent framework; we argue that there are significant advantages to so doing, due to the inevitable need to combine aspects of these and other scenarios in various ways.*

The design of information services that address these requirements is made challenging by the diversity of resources involved, the range of queries required, and the dynamic nature of VO membership and resource status. Existing technologies do not address these requirements. Directory services such as X.500 [1], LDAP [21], and UDDI [2] do not address explicitly the dynamic addition and deletion of information sources; in the case of X.500 and LDAP, there is also an assumption of a well-defined hierarchical organization, and current implementations tend not to support dynamic data well. Metadirectory services permit coupling multiple information sources, but otherwise inherit the quality of their database engine. Service discovery services [9, 39, 29, 31] rely on multicast to scope requests, which is inappropriate when virtual and physical organizational structures do not correspond. Monitoring and event systems [36, 30, 11, 6] support information delivery, using disparate protocols, but not the scalable discovery of information sources. Our architecture supports extensions for special purpose (e.g., high data rate) transfer protocols, and so can serve as an integrating framework for monitoring systems, as envisioned in the Grid Monitoring Architecture [34].

We present here an information service architecture that addresses the unique requirements of Grid environments. Our architecture consists of two basic elements:

- A large, distributed collection of generic *information providers* provide access to information about individual entities, via local operations or gateways to other information sources (e.g., SNMP queries). Information is structured in term of a standard data model, taken from LDAP: an entity is described by a set of "objects" comprised of typed attribute-value pairs.

- Higher-level services, collect, manage, index, and/or respond to information provided by one or more information providers. We distinguish in particular *aggregate directory services*, which facilitate resource discovery and monitoring for VOs by implementing both generic and specialized views and search methods for a collection of resources. Other higher-level services can use this information and/or information obtained directly from providers for the purposes of brokering, monitoring, troubleshooting, etc.

Interactions between higher-level services (or users) and providers are defined in terms of two basic protocols: a soft-state *registration protocol* for identifying entities participating in the information service, and an *enquiry protocol* for retrieval of information about those entities, whether via query or subscription. In brief, a provider uses the registration protocol to notify higher-level services of its existence; a higher-level service uses the enquiry protocol to obtain information about the entities known to a provider, which it merges into its aggregate view. Integration with the Grid Security Infrastructure (GSI) [16] provides for *authentication* and *access control* to information.

Our architecture has a number of desirable features. The separation of concerns between information retrieval, on the one hand, and discovery and monitoring, on the other, means that a wide variety of discovery and monitoring strategies can be supported—implementing different tradeoffs between query language expressiveness, information timeliness, and cost—without modifying the various resources and services that comprise the Grid. (Those components need simply to speak the enquiry and registration protocols, or be in communication with a service that does so on their behalf.) The distributed architecture and use of soft-state registration makes the system highly fault tolerant: a component failure or network partition effects only those components that fail or are separated from the observer.

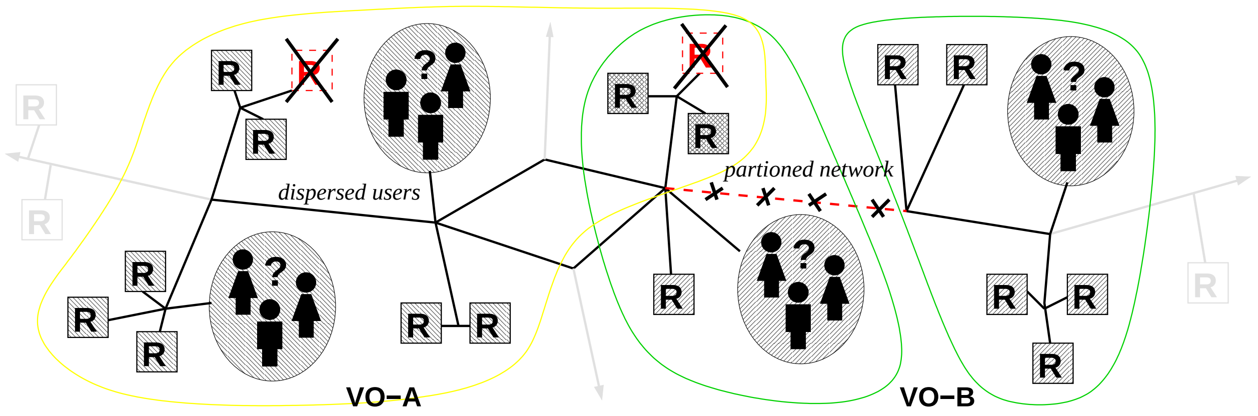

Figure 1. Distributed virtual organizations. Users in VO-A and VO-B have access to partially overlapping resources. While VO-B is split by network failure, it should operate as two disjoint fragments.

Our implementation of this architecture, MDS-2, forms part of the Globus Toolkit of Grid services [14, 17]. MDS-2 supersedes MDS-1 [13], which pioneered the use of Grid information service concepts but did not address all requirements identified here. MDS-2 provides a configurable information provider component called a Grid Resource Information Service (GRIS) and a configurable aggregate directory component called a Grid Index Information Service (GIIS). MDS-2 has been deployed since June 2000 and we now have significant positive experience in operational settings in large, multi-institutional Grids.

In the rest of this paper, we first review our requirements, then describe the various elements of our architecture. Next, we discuss security, our implementation, and approaches to system configuration. We conclude with a discussion of related work and future directions.

2. Grid Information Service Requirements

The requirements of any Grid based information system are driven by basic properties of the Grid environment. Information sources are necessarily distributed and individual sources are subject to failure. The total of number of information providers can be large, and both the types of information sources and the ways in which information is used can be highly varied. We examine the impact of each of these properties on information service requirements.

2.1 Distribution of Information Providers

One consequence of distribution is that we cannot in general provide users with accurate information: any information delivered to a user will necessarily be "old." Since all information to which a Grid information service provides access is, at some timescale, dynamic, the state of the system component on which information is being provided may have changed, potentially rendering the information invalid. Because of the local policy aspect of Grid environments, it can be expensive if not impossible to delay changes in distributed system state until the information has been delivered and processed by remote requestors. Thus, we require that information producers should explicitly model the currency and confidence of their information, for example via timestamps and time-to-live metadata. This approach allows users and delivery components to manage data in a manner that is appropriate for its degree of dynamism. We also require that an information service transport information as rapidly and efficiently as possible from producer to consumer.

The above discussion illustrates that the distribution of information sources places restrictions on the types of information that are both meaningful and feasible to collect. A further ramification results in an important requirement on the types of behavior an information service should *not* have. We argue that we should not in general provide users with a consistent view of global state [8]. Even when accurate information could be guaranteed by the use of transaction mechanisms [22] and distributed snapshots, we believe that it is inappropriate to provide such mechanisms as information service primitives. Such mechanisms do not scale well to large numbers of providers. In summary, we require that a Grid information service should focus only on *efficient delivery of state information from a single source.* If applications require accurate local state or consistent global state, this functionality can be achieved via other control functions that provide necessary atomic operations at a higher cost.

2.2. Managing Failure

In distributed environments, both individual entities and the networks that provide access to those entities may fail. We hence require that information services behave robustly

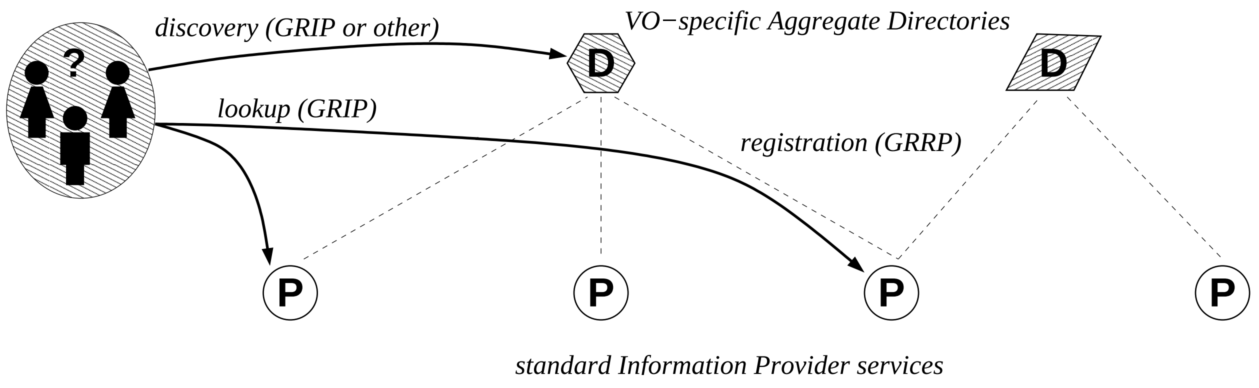
discovery (GRIP or other)
VO−specific Aggregate Directories
D
D
?
lookup (GRIP)
registration (GRRP)
P
P
P
P
standard Information Provider services

Figure 2. Architecture overview. Using the GRid Information protocol (GRIP), users can query aggregate directory services to discover relevant entities, and/or query information providers to obtain information about individual entities. Description services are normally hosted by a Grid entity directly, or by a front-end gateway serving the entity.

in the face of failure of any of the components on which the service is built. We define robust behavior to mean that the failure of any one component should not prevent obtaining information about other components of the system. Users should have as much partial or even inconsistent information as is available (see partitioned “VO-B” in Figure 1).

This requirement has two ramifications. First, an information service should be as distributed and decentralized as possible, with providers located near (e.g., on) the entities they describe. This strategy increases the likelihood of obtaining information about available resources. Second, it means that information system components should be constructed under the assumption that failure is the rule, not the exception. This means not only making sure that unavailable or unreachable services and resources do not interfere with other functions, but also providing a means to gain a timely awareness of when failures have occured. It is this second aspect of system behavior that motivates our use of soft-state registration protocols described in Section 4.3.

2.3. Diversity in Information Service Components

A new VO may involve many entities and have unique requirements for discovery and monitoring. Yet it is typically impractical to require that each resource, service, etc., be modified or reconfigured for VO operation. Instead, we would like to be able to define once, ahead of time, the discovery and enquiry mechanisms that must be supported by any Grid entity. These mechanisms should then be sufficient to support a rich set of discovery and monitoring strategies—including hierarchical resource groupings; alternative resource naming mechanisms; and search procedures—while allowing users, system operators, and resource providers to control overheads, by for example trading off cost and timeliness.

Another aspect of diversity concerns the policies under which information providers are operated and the ways in which information can be used. Components of a Grid information service may span multiple administrative domains. Because of this, information is often provided with restrictions: specific policies must be followed in its use and dissemination. Specifically:

- Information producers may wish to restrict who has access to specific pieces of information based on the requestor’s identify, affiliation, type of information requested, or other factors. Thus we must have robust authentication and authorization mechanisms that information owners will trust.
- Scalability concerns will drive the aggregation of individual information providers into collections, where each collection may correspond to a different VO. However, the administrators of these collections will want to control membership, defining a policy under which information providers can contribute to a VO. Conversely, information providers may wish to assert policy over which VOs they are prepared to join.

Fortunately, policy problems are simplified by the VO paradigm. Because a VO typically groups together users and resources who share common goals and approaches, important policies are often uniform within the community. This uniformity simplifies aggregate policy expression and also allows the construction of shared discovery mechanisms that safely and efficiently provide specialized or sensitive data only to VO members.

3. Architecture Overview

Our Grid information service architecture (Figure 2) comprises two fundamental entities: highly distributed *information providers* and specialized *aggregate directory*

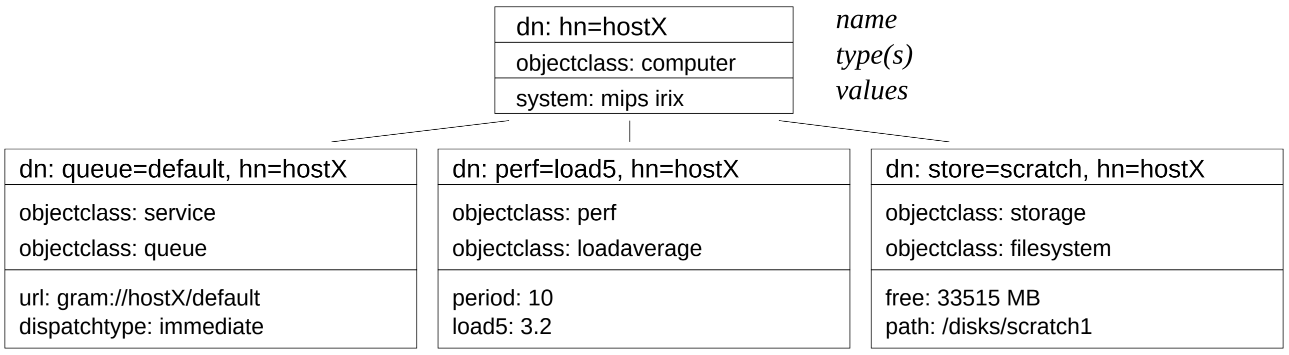

Figure 3. The LDAP data model represents information as a set of *objects* organized in a hierarchical namespace. Each object is tagged with one or more named types. Each object contains bindings of values to named attributes according to the object type(s).

services. Information providers form a common, VO-neutral infrastructure providing access to detailed, dynamic information about Grid entities. For example, a provider for a compute resource might provide information about the number of nodes, amount of memory, operating system version number and load average; a provider for a running application might provide information about its configuration and current status. Aggregate directories provide often specialized, VO-specific views of federated resources, services, and so on. For example, a directory intended for use by a broker might list the computers available to a VO organized by operating system type; another directory, intended to support application monitoring, might keep track of running applications.

An information provider is defined as a service that speaks two basic protocols. The GRid Information Protocol (GRIP) is used to access information about entities, while the GRid Registration Protocol (GRRP) is used to notify aggregate directory services of the availability of this information. *These two protocols are the fundamental building blocks on which our architecture is based*. For an entity to be known to VO participants, it must either speak these protocols directly (hence being its own information provider) or interact with some other entity that acts as an information provider on its behalf.

We define an aggregate directory as a service that uses GRRP and GRIP to obtain information (from a set of information providers) about a set of entities, and then replies to queries concerning those entities. As we explain below, an aggregate directory can itself adopt GRIP as the protocol by which others query it (and, for that matter, GRRP as the protocol that it uses to notify others of its existence), but it is not obliged to do so: in fact, such directories can support arbitrary data models, query languages, and protocols.

The definition of GRIP and GRRP enables a clean separation of concerns between enquiry and discovery. A wide variety of discovery strategies can be implemented simply by constructing aggregate directory services that use GRIP and GRRP in different ways to obtain information that is then used to construct various precomputed *indices*. We discuss this issue below, but here are two examples:

- A name-serving aggregate directory simply records the name of each entity for which a GRRP registration was recorded, and supports only name-resolution queries.
- A relational aggregate directory follows up each registration of a new entity with a GRIP query to determine its properties, which it records in a relational database against which relational queries can be performed.

Notice that there will be inevitably be tradeoffs between the power of an index, the cost associated with maintaining it, and its freshness. Metadata such as timestamps and confidence estimates can be used to guide how often information maintained in indices is updated, and both push and pull models can be used to move information from providers to directories. Of course, following discovery, a client can always refresh interesting information by directly consulting the authoritative source.

Aggregate directories make an important contribution to the scalability of the information services architecture. Each aggregate directory provides a single point of contact to which queries for a specific VO may be directed. More importantly, each aggregate directory defines a scope within which search operations take place, allowing users and other services within a VO to perform efficient discovery without resorting to searches that do not scale well to large numbers of distributed information providers. This scoping allows many independent VOs to co-exist in a grid without adversely affecting their individual discovery performance.

We observe that this architecture embodies many of the same structural principles as the World-Wide Web, arguably the largest federated information system. GRIP corresponds to HTTP, and aggregate directories to search engines.

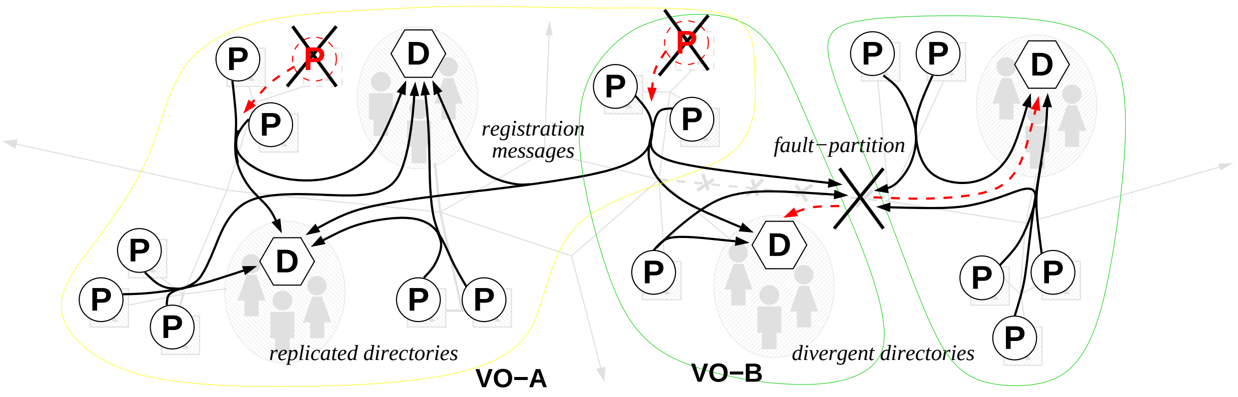

Figure 4. Fault-tolerant registration. Information providers *register* with aggregate directories to provide user communities with listings of available resources. The redundant VO-A directories converge, while the VO-B directories cannot due to network partition.

4. Base Protocols

We now describe GRIP and GRRP, focusing on defining the protocols and indicating the roles that they play in a Grid information system. We also note a few roles that are best addressed by other protocols. In Section 10, we discuss how GRIP and GRRP have been implemented in MDS-2.

4.1. Grid Information Protocol

A user, or more frequently an aggregate directory or other program, uses GRIP to obtain information from an information provider about the entity(s) on which the provider possesses information.

Because an information provider can possess information on more than one entity, GRIP supports both discovery and enquiry. Discovery is supported via a *search* capability. For example, consider an information provider that maintains information on a set of workstations. A broker might then perform a search on that provider to obtain a set of results that roughly match a given criteria. From the set of discovered resources, enquiry can be used to refine the set of resources upon which a broker may schedule. Enquiry corresponds to a direct *lookup* of information: the enquiry supplies the resource name and the provider returns the resource description. Subscription (i.e., a request that results in the subsequent delivery of a sequence of updates) can be an important enquiry mode, and should be supported.

We adopt the standard Lightweight Directory Access Protocol (LDAP) [21] as the protocol for GRIP. LDAP defines a data model, query language, and wire protocol. Aspects of the *data model* are illustrated in Figure 3; notice in particular the use of *object classes* with named types to characterize resources, and the hierarchical *distinguished names* used to name resources within the information provider. The *query language* supports search, lookup, and (via recently proposed extensions) subscription operations. A filter can be used in all cases to specify a set of criteria to be matched. A subset of attributes associated with an entity can be retrieved—reducing the amount of information that must be transmitted. We have used this query language within MDS-1 [13] with great success in a Grid context. Finally, the LDAP *protocol* supports a query-reply exchange, with the query specifying a search, lookup, or (via recent proposed "persistent search" extensions [32]) subscription, and the reply comprising the specified fields of any matching object(s).

We emphasize that we adopt LDAP as a data model, query language, and protocol, not an implementation vehicle. Hence, while it is certainly feasible to use existing LDAP server technology to construct information providers (and/or aggregate directories), this strategy may be inappropriate if the LDAP server implementation is optimized for read access to static data.

One consequence of the above is that there is no requirement that an information provider explicitly store information about its entity(s): it can, for example, generate dynamic information only as it is requested. It follows that the number of entities do not necessarily have to be enumerable: a provider can represent an infinite parametric name space, generating elements of this space lazily in response to direct queries. For example, we have constructed in collaboration with Rich Wolski and Martin Swany an information provider that allows users to request bandwidth information for entities corresponding to network links connecting specified endpoints. In practice, such requests do not access a database maintained within the information provider, but are handed off to the Network Weather Service (NWS) [40] network performance characterization system, which may variously access cached data or perform an experiment. Information providers that support queries on nonenumerable namespaces might signal an error and/or re-

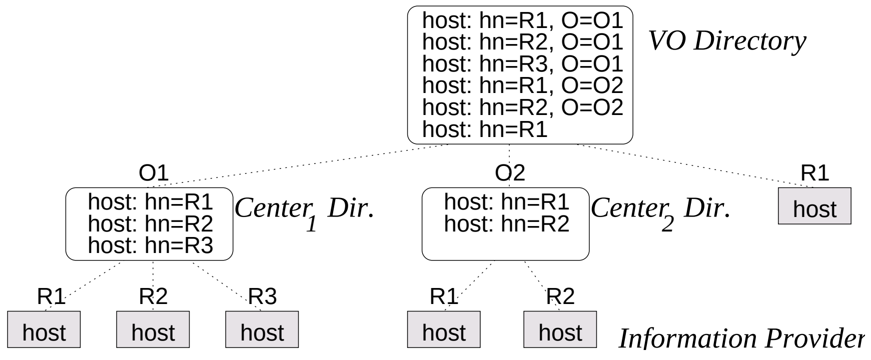

Figure 5. Hierarchical discovery. Two resource centers and one individual are contributing resources (bottom, squared boxes) to a VO. The three aggregate directories (upper rounded boxes) that form the associated hierarchical discovery service are organized in a way that matches this logical structure. Notice how resource names can be used to scope searches to particular organizations, if this is desired; alternatively, searches can be directed to the root directory without concern for scope.

turn partial results for searches that use too wide a scope.

We also observe that LDAP naming conventions (i.e., hierarchical, globally unique, distinguished names) need only be observed within the scope of a specific provider or directory. While common naming conventions can simplify some administrative tasks, globally unique names are defined by combination of name of information within the scope of the provider and the name of the provider (i.e., an LDAP URL that includes the host name, port number and distinguished name of the information in question).

4.2 Limitations of Information Protocol

The LDAP query language also has its limitations. In particular, it cannot specify relational "joins," i.e., operations that allow data from several different typed entities to be combined to yield a new composite entity. We believe that join should not be provided as part of the basic GRIP query language. The utility of generalized joins is limited within the Grid due to our inability to view a consistent global state. A join operation can be supported when needed via an optimized discovery service, using techniques such as those found in current LDAP-based metadirectory servers (Section 5.2).

4.3. The Grid Registration Protocol

GRRP complements GRIP by defining a notification mechanism that one service component can use to "push" simple information about its existence to another element of the information services architecture. For example, GRRP is used by an information provider to notify a aggregate directory of its availability for indexing, or by an aggregate directory to invite an information provider to join a VO.

GRRP is defined in detail elsewhere [18]. In brief, it is a soft-state protocol [5], meaning in our context that state established at a remote location by a notification (e.g., an index entry for an information provider) may eventually be discarded unless refreshed by a stream of subsequent notifications. Such protocols have the advantages of being both resilient to failure (a single lost message does not cause irretrievable harm) and simple (no reliable "de-notify" protocol message is required).

Each GRRP message contains the name of the service that is being described (i.e., a URL to which GRIP messages can be directed), the type of notification message, and timestamps that determine the interval over which the notification should be considered to hold. The GRRP definition does not specify the underlying transport: it is designed to run over an unreliable transport, but a reliable transport can also be used. We discuss in Section 10 below the transport adopted in our MDS-2 implementation of GRRP.

The use of GRRP in constructing resilient aggregate directories is depicted in Figure 4. Under the direction of local and VO-specific policies, an information provider determines the directory(s) with which it will register. The provider then sustains a stream of registration messages to each directory. After some time without a refresh, the directory can assume the provider has become unavailable, and purge knowledge of it from its directory. As long as a directory has fresh knowledge of a provider, it might include that provider in results for relevant discovery queries. Conversely, when its knowledge of a provider becomes stale, the discoverer might omit that provider from results. The question of how a resource provider determines to which directories to register is discussed in Section 9.

GRRP provides a discoverer with an *unreliable failure detector* [7]. A discoverer can decide at a certain point (e.g., after a certain amount of time has passed without a registration message being received from a producer) that the producer has failed or become inaccessible. (Discriminating between resource and network failure is fundamentally difficult for any remote monitoring approach.) Any such decision can be erroneous, as the missing registration messages may have been sent but discarded by a lossy network connection. There is thus a tradeoff to be made, when choosing the criteria used to decide that a producer has failed, between likelihood of an erroneous decision and timeliness of failure detection. An experimental study of a failure detection service provided in earlier versions of the Globus Toolkit, the Globus Heartbeat Monitor (now superseded by GRRP) [33], showed that in typical wide area networks, failure detectors can operate effectively despite often high packet loss rates [3, 25, 4, 28].

5. Aggregate Directory Services

Having defined GRIP and GRRP, we turn to the question of how to create aggregate directory services. As we indicated above, one can define any directory that one wants—there are no technical limitations on what sorts of indices or naming, search, and monitoring strategies directories may maintain, or on what data models, query languages, and protocols they may support. Nevertheless, there are significant benefits to adopting standard data models, query languages, and protocols within aggregate directories, as otherwise users and programs have to write different code to query each directory variant that might be produced. We anticipate that in practice a Grid information system will see a small number of standard aggregate directory structures. In the following, we first explain how GRIP and GRRP can be used to construct a standard hierarchical discovery service and then briefly discuss some alternatives.

5.1. GRIP and GRRP for Aggregate Directories

Figure 5 shows how a hierarchical discovery service can be structured as a network of aggregate directories. Each directory uses the GRIP data model, query language, and protocol, and acts as an information provider that contains information about all of the resources beneath it in the hierarchy. Directories use GRRP to register with higher-level directories to construct the hierarchy. Such aggregate directories could also use lossy aggregation techniques, as in the Service Discovery Service [9], which hashes descriptions and summarizes hashes via Bloom filtering.

This hierarchical discovery structure conveniently mirrors the typical decomposition of VO administration, with multiple site administrators coordinating with the VO service administrator(s). Each site administrator can maintain their local aggregate directory and register it with directories maintained by the VOs in which the site participates. We note that such hierarchical directories can easily be created using GRRP. Local aggregate directories can use GRRP to register with VO directories just as information providors use GRRP to register with local aggregate directories, as described in the preceding section.

5.2. Specialized Services

We envision a hierarchical discovery service serving primarily as a name-based location service, allowing users to discover what resources are available within a VO but not supporting sophisticated queries on those resources.

Given such a service, we can easily construct more specialized aggregate directories that use the basic hierarchical discovery mechanism to discover VO members and then use GRIP queries to gather more detailed information about the member resources. The specialized directory servers can use this detailed resource information to construct search indices that can be used to answer various types of qualitative queries. In effect, the result is that the specialized directories have defined an alternative organization or namespace for the information, creating a view that is optimized towards specific usage patterns.

A specialized aggregate directory may choose to adopt specialized update strategies to improve its ability to perform its specialized functions. For example, a directory designed to locate "idle multicomputers" might maintain an index of only these resources, and then keep careful track of changing patterns of multicomputer load so as to maximize accuracy while minimizing query traffic. A directory designed to monitor network resources for unacceptable load conditions might use historical information to direct its queries (whether using push or pull models) to expected trouble spots. In constructing such specialized services, one can of course use any appropriate database technology to maintain the necessary indices.

5.3. Alternative Directory Protocols

We pointed out above the benefits of using the GRIP data model, query language, and protocol when constructing aggregate directories. However, one can certainly define alternatives. For example, it has been argued [12] that relational data models and query languages can be useful in Grid settings, due to their ability to support join operations. (E.g., "find me an idle computer that is connected to an idle network.") Directories that maintain relational representations of associated resources and that support SQL or some other relational query language can of course be constructed in this framework. Or, we can construct directories that employ the Condor matchmaking algorithm as a query evaluation mechanism [23] (e.g., see [38]).

6. Monitoring and Other Applications

In the preceding section we described how GRIP and GRRP can be used to construct a variety of information service components called, collectively, aggregate directories.

These same protocols, and many of the same strategies, can be used to construct a variety of other services and applications, concerned, as we noted in the introduction, with such things as brokering, monitoring, application adaptation, troubleshooting, and performance diagnosis.

Many such applications are concerned primarily with monitoring rather than discovery of Grid entities. Monitoring and discovery are closely related activities that involve, in many cases, the same types of information—but that tend to differ in terms of their preferred delivery mechanisms. In the case of discovery, we are primarily concerned with the characteristics of an entity at a specific point in time, and so typically want to request information explicitly. In the case of monitoring, we are more often interested in how characteristics vary over time, and so may prefer that the information is delivered asynchronously if and when specified conditions are met: for example, when an information value changes by a specified amount.

GRIP is designed to support both delivery models and hence to support both discovery and monitoring. In *pull* mode, a query-response exchange supports on-demand access to information; in *push* mode, an initial subscription request [32] requests subsequent asynchronous delivery.

However, even given this flexibility, there are information delivery roles for which GRIP is ill-suited. The retrieval of archival information can require the support of more powerful database query interfaces, to reduce search costs over a continuously growing mountain of data. Similarly, various flavors of event delivery system [6] can provide specialized synchronization and reliability properties (e.g., source-ordered delivery to multiple recipients, or exactly-once delivery). Nevertheless, we believe that GRIP supports the discovery and characterization needs of a large proportion of remote observers—and we emphasize that the adoption of this standard approach has tremendous advantages in terms of interoperability.

For those exceptional situations where GRIP is not adequate, our architecture allows for extensions in two ways:

- *GRIP extension.* Resources may offer additional information delivery capabilities beyond those provided by GRIP. For example, an information provider that interfaces to a large archive might implement protocol extensions to support richer relational queries.

- *Service publication.* GRRP and GRIP are designed to permit discovery of other Grid resources and services. For example, a high data rate network monitor [36] may deliver information via a specialized, binary-encoded push protocol. The information provider for this monitor can indicate that this protocol is supported, and provide the information needed to subscribe to it.

We emphasize that such capabilities should be viewed as extensions, not replacements, for GRIP and GRRP, which must be universally deployed to ensure interoperability. Extensions permit more specialized behaviors in specialized situations, and their use should be circumscribed. For example, the network monitor might push its information to a network profiler that publishes summary information for general distribution using standard GRIP and GRRP.

7. Security

Physical and virtual organizations typically define policies controlling who can access information about their resources. Any Grid information service must hence incorporate security mechanisms so that it can comply with these policies. Security issues arise with both GRIP and GRRP.

Our security approach is intended to support a wide range of access control policies. We assume that an information provider may specify, for each piece of information that it maintains, the credentials that must be presented to access that information. These credentials may be identity credentials, in which case the access control policy is essentially an access control list, or a capability issued by some authority, in the case of policies based, for example, on group membership [27]. GSI public-key security mechanisms are used to verify credentials and to achieve mutual authentication between information consumers and information providers.

Aggregate directories pose interesting security issues, as these services make available to others information obtained from information provider(s). This distribution of information must be performed in a fashion consistent with the policy of the underlying provider(s). Fortunately, our architecture can support a number of alternatives:

- *The information provider(s) and aggregate directory have the same data access policy and the provider(s) trusts the directory*. Here, the provider can respond to any authenticated query from the directory, which it trusts to apply its policy on its behalf. We note that this case may be common, as VOs often link together institutions and people with common policy concerns.

- *The information provider(s) limits the information that is available to an aggregate directory*. For example, provider policy may make operating system type known to a directory, but demand that load averages can only be given to specific users. A query for machines running RedHat Linux 6.2 with a load of less than 1.0 would thus require a first query to the directory to identify Linux platforms and a second set of queries (perhaps subscriptions) to each machine enquiring as to their load. The second query would require re-authentication and then application of the local access control based on the requestor's identity.

- *The information provider makes no information known other than its existence*. In this situation, the directory can only enumerate the known resources, with no attribute-based indexing possible.

- *The information provider places no restriction on the information provided*. In this case, authenticated queries are not required.

Security issues also arise with respect to the registration protocol. We need to (1) ensure that registration messages are authentic, and (2) control which registration events are accepted and which are denied. Authenticity of GRRP messages can be determined by two alternative means. GRRP messages can be delivered over a secure channel, such as that provided by *globus_io* using GSI. Alternatively, we can cryptographically sign each GRRP message with the credentials of the registering entity. A variety of access control mechanisms can be applied at the message destination prior to accepting the message, as we know the identity of the service generating the registration message.

8. Generating Unique Names

We touch briefly upon the tangential, but related and important, topic of naming. It can be desirable to be able to assign entities names that are guaranteed to be unique within some scope [26]. (Globally unique names are a special case of this requirement.) We may want unique names so that we can refer to the same entity repeatedly, or so we can determine whether or not the results of two searches refer to the same entity.

Obtaining unique names is not straightforward. Some resources have DNS names, but certainly not all, so some other mechanism is also required. To address this problem, we describe two possible approaches, one based on naming authorities and one on probabilistic techniques.

In the first approach, we introduce *naming services* responsible solely for generating names guaranteed to be unique within the scope that the naming service operates. Such a service could be operated by individual VOs for their own private purposes; alternatively, an international standards body could be chartered with operating the service so as to provide globally unique names. (The Domain Name Registry that manages Internet domain names is an example of such a service.) Particularly in the latter case, a hierarchical organization of this service will be important, for scalability. This use of a distinct naming service has the disadvantage of introducing a new class of infrastructure, which must be created, maintained, etc.

In the hierarchical organization described in Section 5.1, each aggregate directory effectively serves as a local naming authority. Note that in this case, administrative overheads are low but names are only relatively unique: different entries can have the same name, and a single entry multiple different names, within different hierarchies.

In the second approach, we assign names at random from a large name space, hence obtaining a name that is highly likely to be unique. A Globally Unique Identifier (GUID) is an example of such a system. Of course, such names do not contain any structural information: they cannot be used to scope searches, for example. But we can use other techniques, such as the hierarchies of Section 5.1, for that purpose. This may be the preferred approach.

It can also be desirable to be able to enforce standard formats for entity descriptions, so that entities that share major characteristics have comparable descriptions. This implies a need for both a convenient and extensible mechanism for defining information types, and a mechanism for assigning (and discovering) type names. If required, this capability can be provided via type authorities.

It is important to note that neither naming nor typing is a universal requirement: one can construct useful discovery and monitoring systems that provide neither unique names nor consistent naming of types. (For example, the Condor Matchmaker [23] does not enforce a type system, relying instead on informal procedures for achieving reasonably consistent descriptions.) Hence, we argue that a Grid information service should support naming and typing but not force systems that do not require these capabilities to pay for them.

9. Configuring Information Services

Our previous discussion on constructing aggregate directories was predicated on the assumption that the information providers knew in which aggregate directories they wished to be registered. In this section, we examine three possible techniques by which this association can be made. Others approaches are also possible.

Manual configuration. Users or system administrators can configure information providers with the addresses of directories, or directories with the names of providers. This approach has obvious scalability problems, but is practical in the case of small and/or long-lived VOs. Note that in hierarchical organizations (Section 5.1), an entire organization's resources can be added to a VO by registering the appropriate directory, thus overcoming scalability issues.

Automated discovery based on a hierarchical discovery service. New discovery services can potentially be configured via searches of a hierarchical discovery service (Section 5.1), if one exists.

Automated discovery based on other information services. We can use services such as SLP [29], DNS [24], or UDDI [2] to assist with configuration. For example, clients can use SLP to locate a default local directory from which to initiate VO resource discovery. Multicast techniques may sometimes be appropriate (see Section 11.2).

10. Implementation

We have constructed a Grid information service, MDS-2, that implements the architecture described above. MDS-2.0 was released in the Globus 1.1.3 distribution in June 2000 and has since been widely deployed. We describe here the MDS-2.1 version of MDS-2, which offers improved performance and additional features over MDS-2.0, and which we are currently preparing for release. *MDS-2.1 implements all functionality described in this paper, with the exception of push operations.*

10.1. MDS-2.1 Protocol Engine

Recall that our architecture defines two base protocols, GRIP and GRRP, with GRIP defined to be LDAP and GRRP defined in [18] but without specifying transport. In MDS-2.1, we adopt LDAP as our GRRP transport, with GRRP messages mapped onto LDAP `add` operations and then carried via the normal LDAP protocol. This choice of LDAP was made for pragmatic reasons (it simplified development); alternative GRRP transport protocols are certainly possible and in the future we may use SOAP for this purpose, if the use of SOAP in other Grid services becomes prevalent.

The use of a standard protocol allows us to exploit a large body of existing code in our MDS implementation, including, in the case of MDS-2.1, the OpenLDAP client and server implementation of the LDAP V3.0 protocol. (MDS-2.0 implemented LDAP V2.0.) The OpenLDAP server defines an extensible server framework in which specialized *backends* can by plugged into a standard protocol interpreter. The interpreter handles all authentication, data formatting, query interpretation, results filtering, network connection management, and dispatch to the appropriate backend. This flexible design allows us to use the OpenLDAP server without modification.

10.2. Security

MDS-2.1 offers complete integration with GSI single sign-on authentication and access control mechanisms [16], and both authentication and access control can be supported if required by information provider and/or information consumer policy. This integration of GSI into OpenLDAP did not require any modification to the normal code base, due to OpenLDAP 2.0 including optional security bindings using the Simple Authentication and Security Layer (SASL) library, which in turn includes bindings for GSS-API, a standard authentication API of which GSI is an implementation. Thus we can load GSI support dynamically into a standard OpenLDAP server.

MDS-2.1 access control list functionality is consistent with defined LDAP access control mechanisms and supports the specification and enforcement of policies that restrict access to specific pieces of information. Richer access control mechanisms are currently being developed for LDAP and we will exploit these once they become standardized. OpenLDAP also passes authentication information to the information service-specific backends, a feature that enables information providers to enforce arbitrary policy constraints on the information that they provide. Additional policy mechanisms will be introduced into upcoming revisions to the MDS-2 server code. Support for capabilities will be added when the Globus Community Authorization Service is complete.

10.3. Information Providers (GRIS)

The MDS-2 release includes a standard, configurable information provider framework called a Grid Resource Information Service (GRIS). This framework is implemented as an OpenLDAP server backend that can be customized by plugging in specific information sources. To date, we have implement information sources for static host information (operating system version, CPU type, number of processors, etc.), dynamic host information (load average, queue entries, etc.), storage system information (available disk space, total disk space, etc.), and network information via the Network Weather Service (network bandwidth and latency, both measured and predicted) [40]. Others, including highly dynamic information sources that will push frequent updates, are planned.

GRIS authenticates and parses each incoming GRIP request and then dispatches those requests to one or more “local” information providers, depending the type of information named in the request. Results are then merged back to the client. To efficiently prune search processing, a specific provider’s results are only considered if the provider’s namespace intersects the query scope.

The GRIS communicates with an information provider via a well-defined API. We have implemented two variants of this API. The simpler version is implemented via a set of scripts (typically Unix shell scripts) that are called by the back end. This more sophisticated version is implemented via loadable modules, thus allowing provider implementations to execute within the server. Such modules allow low-latency providers to execute without the overhead of server-side process creation, and also allow the construction of efficient providers requiring RAM-based persistent state. Thus a GRIS is configured by specifying the type of information to be produced by a provider and the provider-defined set of routines that implement the GRIS API. Configuration can be done either dynamically or statically via configuration files.

To control the intrusiveness of GRIS operation, improve response time, and maximize deployment flexibility, each provider’s results may be cached for a configurable period of time to reduce the number of provider invocations; this cache time-to-live (TTL) is specified per-provider as part of the local configuration, and the appropriate value depends greatly on both the dynamism of the modeled resource and the cost of the provider mechanism. By providing caching as part of the GRIS, we simplify the implementation of information providers.

Results returned by an information provider are filtered by the GRIS to eliminate any objects that do not match the client's search space and/or filter constraints, prior to sending them to the destination. This is not a performance optimization, but a necessary step to ensure that the protocol's search semantics are implemented correctly. Filtering is implemented by the GRIS and not the information providers so that (1) simple providers need not duplicate this implementation effort, and (2) cached providers can maximize their performance by returning a superset of results that are then processed out of the cache for each client request.

10.4. Aggregate Directory (GIIS)

The MDS-2.1 code base also provides a framework for constructing aggregate directories called Grid Index Information Services (GIIS), plus a simple instantiation of this framework that implements a simple aggregate directory that provides the hierarchical structure discussed in Section 5.1. The simple directory accepts GRRP messages from "child" GRIS or GIIS instances and merges these information sources into a unified information space. Client searches may obtain information from any or all of the GIIS' children, as illustrated in Figure 5.

The GIIS framework comprises three major components: generic GRRP handling, pluggable index construction, and pluggable search handling. As with GRIS, GIIS functionality is implemented as a special purpose backend for an OpenLDAP server, and in fact index construction and search handling both use the same APIs used to interface a GRIS to an information source.

As mentioned above, GRRP messages are delivered using LDAP as the transport protocol. The OpenLDAP frontend decodes GRRP messages and delivers them to the backend to perform any actions necessary to construct and maintain the GIIS' indices. In the case of our simple aggregate directory, these actions comprise little more than management of a *list* of active providers. In a more complex directory, index maintenance might include the gathering and transformation of additional state information from the remote providers using GRIP.

Our GIIS (and GRIS) implementations are configured so that GRRP can be used for both registration and invitation. In registration, a GRIS explicitly registers with an aggregate directory: in effect, it joins a VO. In the case of invitation, a GRIS is asked to join by the aggregate directory service—or perhaps a third party. If a GRIS agrees to join, it turns around and uses GRRP to register itself with the specified aggregate directory in a fault-tolerant manner.

Pluggable search handling allows for the implementation of customized data access and search mechanisms. In our simple aggregate directory service, we implement *chaining*: GRIP requests directed to the GIIS are simply forwarded on to the appropriate information provider for response. The list of active providers constructed in response to GRRP messages is used to map between the requested entity name and its location, or to support searches across multiple entities.

Performance concerns make caching data within the GIIS desirable, and this capability is provided as part of the basic GIIS framework. As discussed in Section 7, access control issues complicate caching. Clearly, the GIIS can fetch and cache any data available to anonymous clients. Because the GIIS can also bind using a trusted server credential, each GRIS may export some data that it trusts the GIIS to handle properly. In the absence of delegation, GIIS is unable to transfer data from GRIS to client if the data is restricted by the GRIS to only be visible to that client. In this situation, we can return the name of the information provider directly to the client in the form of a LDAP URL using the referral mechanisms defined as part of the standard LDAP protocol.

It should be clear from this discussion that our GRIS and GIIS implementations have much in common. Both rely on an LDAP front end for protocol processing, authentication, and result filtering. Both use a common API for customization, and can in fact co-exist within the same server. Using a common protocol for provider and directory interactions not only promotes interoperability, it also simplifies implementation.

11. Related Work

We have referred to a range of related work in the body of the paper. Here we make additional comments concerning the use of standard directory services, metadirectory services, and other work on service discovery and monitoring.

11.1. Directory Services

One approach to constructing a Grid information service is to push all information into a directory service. We employed this approach in early versions of MDS-1 [13]. While this system pioneered information services for the Grid, the strategy of collecting all information into a database inevitably limited scalability and reliability, even when (as in later versions of MDS-1) LDAP distribution and replication mechanisms were used. In addition, the well-defined hierarchical naming structure required that prior arrangements be established between participants on name space structure, which proved problematic in practice.

Metadirectory services (e.g., RadiantOne) focus on creating a uniform directory service from diverse information sources. These services could, presumably, be used to create specialized views associated with VOs. This can be accomplished by copying data and coordinate updates among different information sources. Metadirectory services, however, suffer from the same problem as standard directory services—they are not designed to handle highly dynamic data. Moreover, scalability is limited due to the coordination required between a multitude of directory services.

11.2. Service Discovery

The Domain Name Service [24] and Globe [37] systems are both designed to support scalable service discovery. These systems scale extremely well, to the extent that all resources on the Internet can be named. These systems are used to map these names into IP addresses and object brokers, respectively. Both systems, however, use a predefined hierarchical naming scheme and only support extremely simple queries, two assumptions that we cannot make in the VO environment. In addition, neither support security that controls access to information.

The Service Location Protocol (SLP) is a proposed protocol to simplify the advertisement and the discovery of network resources [29]. SLP resources, which are associated with a local intranet, are grouped into predefined administrative domains called "scopes" [29, 19]. Resource discovery can be achieved via static configuration, DHCP request, or advertisement via IP multicast. The use of static configuration or multicast makes SLP inappropriate to support VOs. VOs can be highly dynamic making static configuration impractical. Furthermore, VOs can span multiple intranet domains, making the use of broadcast protocols problematic and restricting scalability. However, as noted in Section 9, SLP and other similar services can be used to assist with configuring a Grid information service.

A number of other proposed service discovery services also rely on IP multicast to locate or to disseminate service descriptions: for example, the Service Discovery Service (SDS) [9], the JINI Lookup Service [39], and the wide area extension to SLP, WASRV [31]. Although these services differ in their description, matching, and security mechanisms, the reliance on IP multicast makes them inappropriate for our use.

11.3. Monitoring Services

Various systems support the monitoring of network and Grid resources, services, and applications: for example, NetLogger [36, 35], Autopilot [30], Remos [11], and Paradyn [20]. These systems incorporate a range of often sophisticated sensor interface, instrumentation, data collection, data filtering, and data summarization techniques that have proven invaluable in a range of application experiments. However, each systems uses a different data delivery protocol and while some (e.g., Autopilot and Remos) provide limited directory support, there are no standard mechanisms for discovering and characterizing information sources. The difficulties these heterogeneities cause for applications that require integrated, end-to-end monitoring and analysis emphasize the benefits of a uniform Grid information architecture such as that proposed here.

12. Conclusions and Future Work

We have described a Grid information service architecture that defines simple data models and registration and enquiry protocols for Grid entities, and supports the creation of a wide assortment of specialized information services as well as other high-level, information-intensive services.

Our implementation of this architecture, MDS-2, has been widely deployed in a number of different configurations as part of the Globus 1.1.3 software release. We are currently working to incorporate subscription-based *push* methods and more sophisticated access control methods. We also plan to explore the construction of different and more specialized types of aggregate directories, investigate update versus freshness tradeoffs in directory implementation, explore applications in different settings and domains, develop flexible configuration tools to enable lightweight VO formation, and extend our security models to incorporate capabilities and delegation to enable more sophisticated directory construction and caching of information provider values.

Acknowledgements

We are grateful to many colleagues for numerous discussions on the topics discussed here, in particular Ruth Aydt, Peter Dinda, Michael Helm, Bill Johnston, Miron Livny, Bruce Lowekamp, Warren Smith, Brian Tierney, Steve Tuecke, Gregor von Laszewski, and Rich Wolski. This work was supported in part by the Mathematical, Information, and Computational Sciences Division subprogram of the Office of Advanced Scientific Computing Research, U.S. Department of Energy, under Contract W-31-109-Eng-38; by DARPA; by the National Science Foundation; and by the NASA Information Power Grid program.

References

[1] Recommendation X.500, Information technology - Open System Interconnection - The directory: Overview of concepts, models, and services. ITU-T, November 1995.

[2] Universal description discovery and integration (UDDI). http://www.uddi.org, 2001.

[3] J. Bolot. Characterizing end-to-end packet delay and loss in the Internet. *Journal of High-Speed Networks*, 2(3):305–323, 1993.

[4] M. S. Borella, D. Swider, S. Uludag, and G. Brewster. Analysis of end-to-end Internet packet loss: Dependency and asymmetry. Technical Report AT031798, 3Com Advanced Technologies, 1998.

[5] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin. Resource reservation protocol (RSVP) version 1 functional specification. Internet RFC 2205, September 1997.

[6] A. Carzaniga, D. S. Rosenblum, and A. L. Wolf. Design and evaluation of a wide-area event notification service. *ACM Transactions on Computer Systems*, to appear.

[7] T. D. Chandra and S. Toueg. Unreliable failure detectors for reliable distributed systems. *J. ACM*, 43(2), Mar. 1996.

[8] K. M. Chandy and L. Lamport. Distributed snapshots: Determining global states of distributed systems. *ACM Transactions on Computer Systems*, 3(1):63–75, 1985.

[9] S. E. Czerwinski, B. Y. Zhao, T. D. Hodes, A. D. Joseph, and R. H. Katz. An architecture for a secure service discovery service. In *Mobicom '99*. ACM Press, 1999.

[10] P. Dinda. Online prediction of the running time of tasks. In *Proc. 10th IEEE Symp. on High Performance Distributed Computing*, (to appear) 2001.

[11] P. Dinda, T. Gross, R. Karrer, B. Lowekamp, N. Miller, P. Steenkiste, and D. Sutherland. The architecture of the Remos system. In *Proc. 10th IEEE Symp. on High Performance Distributed Computing*, 2001. (to appear).

[12] P. Dinda and B. Plale. A unified relational approach to grid information services. Grid Forum Working Draft GWD-GIS-012-1, February 2001. http://www.gridforum.org.

[13] S. Fitzgerald, I. Foster, C. Kesselman, G. von Laszewski, W. Smith, and S. Tuecke. A directory service for configuring high-performance distributed computations. In *Proc. 6th IEEE Symp. on High Performance Distributed Computing*, pages 365–375, 1997.

[14] I. Foster and C. Kesselman. Globus: A metacomputing infrastructure toolkit. *Intl. Journal of Supercomputing Applications*, 11(2):115–128, 1997.

[15] I. Foster and C. Kesselman, editors. *The Grid: Blueprint for a Future Computing Infrastructure*. 1999.

[16] I. Foster, C. Kesselman, G. Tsudik, and S. Tuecke. A security architecture for computational grids. In *ACM Conference on Computers and Security*, pages 83–91. ACM Press, 1998.

[17] I. Foster, C. Kesselman, and S. Tuecke. The anatomy of the Grid: Enabling scalable virtual organizations. *Intl. Journal of Supercomputing Applications*, (to appear) 2001. http://www.globus.org/research/papers/anatomy.pdf.

[18] S. Gullapalli, K. Czajkowski, C. Kesselman, and S. Fitzgerald. The grid notification framework. Grid Forum Working Draft GWD-GIS-019, June 2001. http://www.gridforum.org.

[19] E. Guttman, C. Perkins, J. Veizades, and M. Day. Service location protocol, version 2. IETF RFC 2165, 1998.

[20] J. Hollingsworth and B. Miller. Instrumentation and measurement. In *The Grid: Blueprint for a Future Computing Infrastructure*, pages 339–365. 1999.

[21] T. A. Howes and M. Smith. A scalable, deployable directory service framework for the internet. Technical report, Center for Information Technology Integration, Univerity of Michigan, 1995.

[22] B. Lampson. *In Distributed Systems—Architecture and Implementation: An Advanced Course*, chapter Atomic Transactions, pages 246–255. LNCS. Springer-Verlag, 1981.

[23] M. Livny. Matchmaking: Distributed resource management for high throughput computing. In *Proc. 7th IEEE Symp. on High Performance Distributed Computing*, 1998.

[24] P. V. Mockapetris and K. Dunlap. Development of the domain name system. In *SIGCOMM'88*, 1988.

[25] A. Mukherjee. On the dynamics and significance of low-frequency components of network load. *Internetworking: Research and Experience*, 5:163–205, 1994.

[26] R. M. Needham. Names. In S. Mullender, editor, *Distributed Systems*, Frontier Series, chapter 5. Addison-Wesley Publishing Company, 1989.

[27] B. Neuman. Proxy-based authorization and accounting for distributed systems. In *Proceedings of the 13th International Conference on Distributed Computing Systems*, pages 283–291, 1993.

[28] V. Paxson. *Measurements and Analysis of End-to-End Internet Dynamics*. PhD thesis, U.C. Berkeley, 1997.

[29] C. Perkins. SLP white paper topic. http://playground.sun.-com/srvloc/slp_white_paper.html, 1997.

[30] R. L. Ribler, J. S. Vetter, H. Simitci, and D. A. Reed. Autopilot: Adaptive control of distributed applications. In *Proc. 7th IEEE Symp. on High Performance Distributed Computing*, 1998.

[31] J. Rosenberg, H. Schulzrinne, and B. Suter. Wide area network service location. IETF Internet Draft, 1997.

[32] M. Smith, G. Good, T. Howes, and R. Weltman. Persistent search: A simple LDAP change notification mechanism. Working document, November 2000. draft-ietf-ldapext-psearch-03.txt.

[33] P. Stelling, I. Foster, C. Kesselman, C. Lee, and G. von Laszewski. A fault detection service for wide area distributed computations. In *Proc. 7th IEEE Symp. on High Performance Distributed Computing*, pages 268–278, 1998.

[34] B. Tierney, R. Aydt, D. Gunter, W. Smith, V. Taylor, R. Wolski, and M. Swany. Grid monitoring service architecture. Grid Forum Working Draft GWD-GP-006-1, February 2001. http://www.gridforum.org.

[35] B. Tierney, B. Crowley, D. Gunter, M. Holding, J. Lee, and M. Thompson. A monitoring sensor management system for grid environments. In *Proc. 9th IEEE Symp. on High Performance Distributed Computing*, pages 97–104, 2000.

[36] B. Tierney, W. Johnston, B. Crowley, G. Hoo, C. Brooks, and D. Gunter. The NetLogger methodology for high performance distributed systems performance analysis. In *Proc. 7th IEEE Symp. on High Performance Distributed Computing*. 1998.

[37] M. van Steen, F. Hauck, P. Homburg, and A. Tanenbaum. Location objects in wide-area systems. *IEEE Communications Magazine*, pages 104–109, 1998.

[38] S. Vazhkudai, S. Tuecke, and I. Foster. Replica selection in the globus data grid. In *International Workshop on Data Models and Databases on Clusters and the Grid (DataGrid 2001)*. IEEE Computer Society Press, 2001.

[39] J. Waldo. The Jini architecture for network-centric computing. *Communications of the ACM*, 42(7):76–82, July 1999.

[40] R. Wolski. Forecasting network performance to support dynamic scheduling using the network weather service. In *Proc. 6th IEEE Symp. on High Performance Distributed Computing*, Portland, Oregon, 1997. IEEE Press.