Middleware Support for Global Access to Integrated Computational Collaboratories *

Vijay Mann and Manish Parashar
The Applied Software Systems Laboratory
Department of Electrical and Computer Engineering, Rutgers University
94 Brett Road, Piscataway, NJ 08854
{vijay,parashar}@caip.rutgers.edu

Abstract

The growth of the Internet and the advent of the computational "Grid" have made it possible to develop and deploy advanced computational collaboratories . These systems build on high-end computational resources and communication technologies underlying the Grid, and provide seamless and collaborative access to particular resources, services or applications. Integrating these "focused" collaboratories presents significant challenges. Key among these is the design and development of robust middleware support that addresses scalability, service discovery, security and access control, and interaction and collaboration management for consistent access. In this paper we first investigate the architecture of such a middleware that enables global (web-based) access to collaboratories. We then present the design and implementation of a middleware substrate that enables a peer-to-peer integration of and global (collaborative) access to geographically distributed instances of the DISCOVER computational collaboratory for interaction and steering.

1 INTRODUCTION

A Collaboratory is defined as a place where scientists and researchers work together to solve complex interdisciplinary problems, despite geographic and organizational boundaries [1]. Computational collaboratories provide uniform (collaborative) access to computational resources, services and/or applications. These systems expand the resources available to researchers, enable multidisciplinary collaborations and problem solving, increase the efficiency of research, and accelerate the dissemination of knowledge.

The growth of the Internet and the advent of the computational "Grid" [2] have made it possible to develop and deploy advanced computational collaboratories [3][4]. Recent efforts include the Upper Atmospheric Research Collaboratory (UARC) [5][6], Diesel Combustion Collaboratory (DCC) [7][8], Access Grid [9], RCSB [10], EMSL [11], Cactus [12], Astrophysics Simulation Collaboratory [13] and DISCOVER [14][15]. Each of these systems provides a high-level problem-solving environment (PSE) that builds on the underlying Grid technologies to provide seamless access to domain specific resources, services and applications. Together these systems have the potential for enabling truly global scientific investigation through the creation of meta-laboratories spanning many research groups, universities and countries, and transforming computational applications and simulations into global modalities for research and instruction.

While combining these systems can lead to truly collaborative, multi-disciplinary and multi-institutional problem solving, integrating these

* This paper is published in the proceedings of the 10th IEEE International Symposium on High Performance Distributed Computing, IEEE Computer Society Press, San Francisco, CA, August 2001. The research presented in this paper is supported by the National Science Foundation via grants number ACI 9984357 (CAREERS) awarded to Manish Parashar.

"focused" collaboratories presents significant challenges. This is because each of these systems has a unique architecture and implementation, and builds on different enabling technologies. Key among these challenges is the design and development of robust middleware support that addresses scalability, service discovery, security and access control, and interaction and collaboration management for consistent access. Such a middleware should define a minimal set of interfaces and protocols to enable collaboratories to share resources, services and applications on the Grid while being able to maintain their architectures and implementations of choice.

In this paper we first investigate the architecture of such a middleware that enables integration of and global access to computational collaboratories. We then present the design and implementation of a middleware substrate that enables a peer-to-peer integration of and global collaborative web-based access to multiple, distributed instances of the DISCOVER computational collaboratory. DISCOVER provides collaborative access to high-performance parallel and distributed applications for interaction and steering using web-based portals [14][15]. The key design challenge is enabling scalable, secure, consistent and controlled access to remote, highly dynamic distributed applications for real-time monitoring, interaction and steering by geographically distributed scientists and engineers in a collaborative environment. The middleware substrate enables DISCOVER interaction and steering servers to dynamically discover and connect to one another to form a peer-to-peer network. This allows clients connected to their local servers to have global access to all applications and services across all the servers in the network based on their credentials, capabilities and privileges. The design and implementation of the DISCOVER middleware substrate builds on existing web servers and leverages commodity technologies and protocols such as CORBA [16] and HTTP [17]. Its goal is to enable rapid deployment, ubiquitous and pervasive access, and easy integration with 3rd party services, while evaluating the viability of these technologies for advanced Grid applications.

The overall aim of Grid computing is to enable collaborative and coordinated problem solving in dynamic, multi-institutional virtual organizations and it focuses on large-scale resource sharing, innovative applications, and high performance computing [18]. The middleware substrate presented in the paper addresses one aspect of this general problem by providing global collaborative access to grid applications and services.

This paper is organized as follows: Section 2 discusses related work. Section 3 outlines a middleware design for integrating computational collaboratories. Section 4 introduces the DISCOVER web based computational collaboratory for interaction and steering, and describes the design, implementation, and operation of its interaction and collaboration server. This section also presents the design of the middleware substrate for peer-to-peer integration of a network of DISCOVER servers to provide global collaborative access to remote applications. Section 5 describes the implementation and operation of the DISCOVER middleware substrate. Section 6 presents a retrospective evaluation of the design and discusses its advantages and disadvantages. This section also presents an evaluation of commodity distributed technologies and protocols and their ability to support Grid applications, and briefly discusses open issues and challenges. Section 7 presents some conclusions and outlines current and future work.

2 RELATED WORK

Related research can be broadly categorized into current efforts in developing and deploying Computational Collaboratories and Problem Solving Environments, and efforts in Peer-to-Peer and Internet Computing. These categories are briefly described below.

2.1 Computational Collaboratories and PSEs

Computational collaboratories and PSEs such as UARC [5][6], Astrophysics Simulation Collaboratory (ASC) [12][13], Punch [20][21], WebFlow [22][23] and HotPage/GridPort [24][25][26] address different aspects of the overall Grid computing problem [18][19]. For example, UARC and ASC implement applications specific PSE's, WebFlow provides support for composing, configuring and deploying scientific applications on the Grid, and systems such as GridPort provide support for acquiring and managing Grid resources. The systems are briefly discussed below.

PUNCH (The Purdue University Network Computing Hubs) [20][21] presents the user with an illusion of a wide area computer by providing functionality similar to an operating system. It provides a computing portal for allocating resources, and deploying and running applications in a wide-area networked environment.

The NPACI HotPage [24][25] is a user portal that attempts to simplify access to HPC resources

distributed across member organizations, and allows them to be viewed either as an integrated Grid system or as individual machines. The Grid Port toolkit [26] generalizes the HotPage infrastructure and develops a reusable portal toolkit. The overall goal of HotPage and GridPort is to provide secure and customized access to grid services through web portals.

The Astrophysics Simulation Collaboratory (ASC Portal) [12][13] provides an application specific PSE for composing, configuring and deploying astrophysical simulations on the Grid. ASC uses a N-tier application model and builds on commodity technologies such as HTTPS, servlets, and RDBMS. The ASC server architecture leverages ongoing efforts aimed at providing high-level access to grid services such as the Java CoG [27], and the Grid Portal Development Toolkit (GPDK). The Java CoG kit is part of the Commodity Grid (CoG) project, which is working to overcome the difficulties of accessing advanced grid services, such as authentication, remote access to computers, resource management, and directory services by defining mappings and interfaces between the Grid and commodity frameworks (e.g. Java, CORBA, Python) that are familiar to problem solving environment developers.

WebFlow [22][23] provides a framework for publishing and reusing computational modules on the web, so that end users can, using a web browser, visually compose distributed applications using these modules. The overall WebFlow architecture is very similar to DISCOVER – it however addresses a different aspect of the grid-computing problem. The WebFlow middle tier also uses a network of Java enhanced web servers (although it does not exploit the peer-to-peer nature of the servers). Furthermore, WebFlow, like DISCOVER, uses high level distributed technologies like servlets and CORBA, and provides similar advantages such as portability and extensibility.

Note that the efforts described above do not address collaboration, shared workspaces or application interaction and steering. These systems however address related aspects of the overall grid computing problems and are, in fact, complementary to DISCOVER and the research presented in this paper. The overall goal of the middleware substrate presented here (and the related CORBA CoG effort [43]) is to define protocols and mechanisms for integrating these services.

The Salamander middleware substrate [28][29][30] that is used in the UARC [5][6] collaboratory and the IPMA (Internet Performance Measurement and Analysis) project [31] is a wide area network data dissemination substrate. The Salamander substrate provides support for both web casting and groupware applications by providing virtual distribution channels through its channel subscription service. The channel subscription service provides an abstraction for the distribution of data from publishers to subscribers with both anonymous and negotiated push techniques. Salamander supports a limited interaction and steering capability through its negotiated push technology. The DISCOVER computational collaboratory provides a richer control interface allowing users to collaboratively monitor and control overall application execution, access, interact with and steer individual computation objects, manage object dynamics and distribution, and schedule automated periodic interactions.

2.2 Peer-to-Peer Computing and Internet Computing

Peer-to-peer computing, as implemented in Internet communication and file sharing tools like ICQ, Napster [32], Gnutella [33], and Freenet [34], and Internet computing as implemented by systems such as SETI@home, Parabon, and Entropia [19], are examples of the more general sharing modalities and computational structures beyond traditional client server systems and characterize virtual organizations where information and resource sharing can take place among any subset of participants. These technologies and systems present a radical paradigm shift from client-server systems. These systems have so far focused entirely on vertically integrated solutions, rather than seeking to define common protocols that would allow for a shared infrastructure and interoperability. Enterprise computing technologies such as Universal Description Discovery and Integration (UDDI) [35] and Microsoft's .NET [36] are related efforts aimed at supporting discovery of web services and interactions between them. These technologies define protocols for publishing and discovering information about Web Services. A related proposal from Intel [37] presents an approach for peer-to-peer computing for enterprise systems, where jobs are "split" into bite-sized tasks for individual PCs.

The DISCOVER design philosophy and architecture presents a hybrid approach. While the DISCOVER middle tier server architecture is peer-to-peer, it continues to support a client-server view from the users point of view. As a result, clients can access the "closest" server and have access to applications and services provided by all the servers. This approach reduces the performance requirements of the server, and allows for more secure and better-

managed peers as compared to a system comprising only of peers. This is because, security and manageability are still open issues in true peer-to-peer systems, whereas one of the reasons for success of client server systems was the security and manageability associated with having centralized servers. The DISCOVER hybrid approach of having peer-to-peer servers drastically reduces the number of peers in the system and restricts the security and manageability concerns to the middle tier of servers.

3 A MIDDLEWARE ARCHITECURE FOR INTEGRATING WEB-BASED COMPUTATIONAL COLLABORATORIES

A schematic overview of the middleware architecture for integrating web-based computational collaboratories is presented in Figure 1. Web-based computational collaboratories typically have a 3-tier architecture consisting of collaborative client portals at the front end, the computational resource, services or applications at the back-end, and the server(s) in the middle. In order to enable ubiquitous web-based access, clients are kept as simple as possible. In this model, the middle-tier has the responsibility for providing controlled access to the back end, interacting with peer servers, providing a "repository of services" view to the client, and collectively managing and coordinating collaboration. A client can connect to its "local" server and have access to all (local and remote) backend services based on its privileges and capabilities. Backend services include resource access and management toolkits, high-performance applications, and network-monitoring tools. The backend services (resource, service or application) may be specific to a server or may form a pool of services that can be accessed by any server using standard protocols such as CORBA, RMI/RMI-IIOP, DCOM, etc. In the former case, direct access to the service is restricted to the local server, typically due to security, scalability or compatibility constraints. This is typically true of most scientific resources, services and applications. In this case, the local server advertises the service and its interface and clients and peer servers can discover and access the service through the local server. In either case, the servers are connected using a ubiquitous and pervasive protocol such as CORBA/IIOP.

The middleware architecture defines a simple protocol requiring two levels of interfaces and interactions for each server. The first level interfaces provide a means for peer servers to authenticate with the server and query it for active services, applications and users. The second level interfaces define interactions with the active services and/or applications at the server, and enable a peer server or client to authenticate, interact with and invoke the service. For example, in the case of an interactive application it would provide the methods for monitoring application state, requesting/releasing locks for steering accesses, querying and/or changing its parameters, etc. If a server only provides a single instance of an application or a service, e.g. a server providing access to grid services using Java/CORBA CoG and GPDSK, only the second level interfaces would be required.

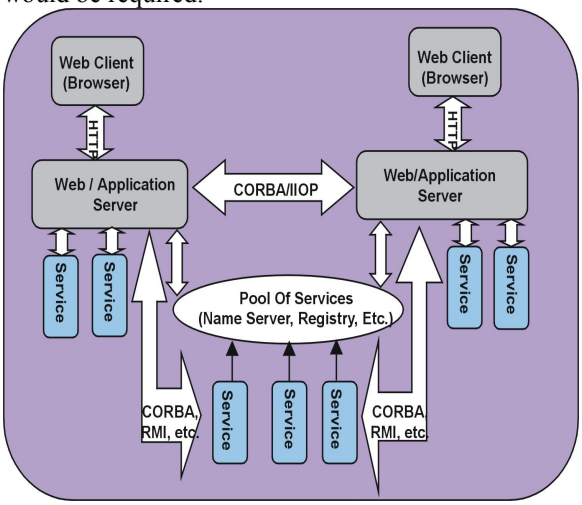

Figure 1. Middleware Architecture for Integrating Web-based Computational Collaboratories

The DISCOVER middleware substrate presented in the following sections is a prototype implementation of these interfaces to enable global discovery of, and provide access to, distributed applications for interaction and steering.

4 DISCOVER: A WEB-BASED COMPUTATIONAL COLLABORATORY FOR INTERACTION AND STEERING

This section presents the design and implementation of a middleware substrate that enables a peer-to-peer integration of and global collaborative web-based access to multiple, distributed instances of the DISCOVER computational collaboratory. DISCOVER is a virtual, interactive computational collaboratory that enables geographically distributed scientists and engineers to collaboratively monitor, and control (new and existing) high performance parallel/distributed applications. Its primary goal is to bring large

(remote) distributed simulations to the scientists'/engineers' desktop by providing collaborative web-based portals for interrogation, interaction and steering. DISCOVER has a 3-tier architecture (see Figure 2) composed of detachable client portals at the front-end, a network of interaction servers in the middle, and a control network of sensors, actuators, and interaction agents superimposed on the application at the back-end. Clients can connect to a server at any time using a browser to receive information about active applications. Furthermore, they can form or join collaboration groups and can (collaboratively) interact with one or more applications based on their capabilities. A network of interaction and collaboration servers forms the middle tier. These servers extend commodity web-servers with interaction and collaboration capabilities. The back-end consists of control network composed of sensors, actuators and interaction agents. Session management and concurrency control is based on capabilities granted by the server (middle) tier. A locking protocol is used to ensure that the applications remain in a consistent state during collaborative interaction and steering. Security and authentication services are provided using customizable access control lists built on the SSL-based secure server. DISCOVER is currently operational[1] and is being used to provide interaction capabilities to a number of scientific and engineering applications, including oil reservoir simulations, computational fluid dynamics, seismic modeling, and numerical relativity. Details about the design and implementation of DISCOVER can be found in [15].

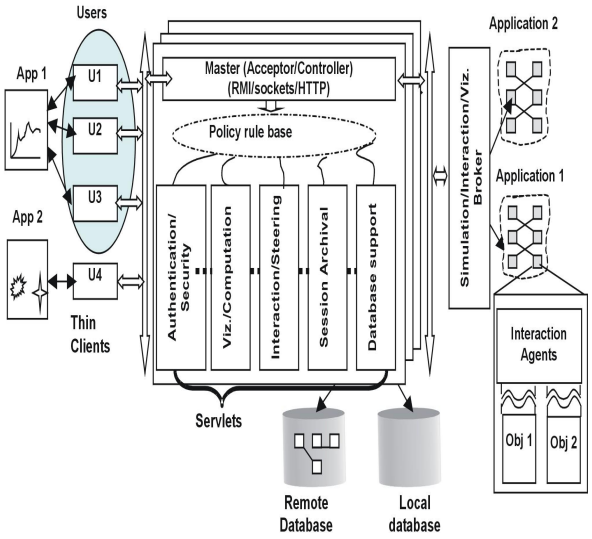

Figure 2. Architectural schematic of the ~~DISCOVER Computational Co~~llaboratory

[1] See http://www.discoverportal.org

4.1 DISCOVER Interaction and Collaboration Servers

The DISCOVER interaction/collaboration server builds on a commodity web server, and extends its functionality using Java servlets [38][39], to provide specialized services for real-time application interaction and steering and collaboration between client groups. Clients connect to the server using standard HTTP communication using a series of HTTP *GET* and *POST* requests. At the other end, application-to-server communication is achieved either using standard distributed object protocols such as CORBA [16] and Java RMI [40], or a more optimized, custom protocol using TCP sockets.

The DISCOVER middleware creates 3 communication channels between a server and an application: (1) a *MainChannel* for application registration and periodic updates, (2) a *CommandChannel* for forwarding client interaction requests to the local or remote application, and (3) a *ResponseChannel* for communicating application responses to the interaction requests. Clients differentiate between the different messages (i.e. Response, Error or Update) using Java's reflection mechanism, by querying the received object for its class name. Messages are processed differently at the client based on their type.

An *ApplicationProxy* object is created at the server for each active application, and is given a unique identifier. This object encapsulates the entire context for the application.

The core service handlers provided by each server include the Master Handler, Collaboration Handler, Command Handler, Security/Authentication Handler and the Daemon Servlet that listens for application connections. In addition to these core handlers, there can be a number of handlers providing auxiliary services such as session archival, database handling, visualization, request redirection, and remote application proxy invocations (using CORBA). These services are optional and need not be provided by every server. We briefly discuss some of the core handlers below.

The *master (accepter/controller) handler* servlet is the client's gateway to the server. The master servlet creates a session object for each connecting client and uses it to maintain information about client-server-application sessions. It provides each client with a unique client-id. The client-id along with an application-id (corresponding to the application to which the client is connected) is used to identify each session.

The *command handler* servlet manages all client view/command requests. On receiving these requests from the master handler, this handler looks up the appropriate application proxy, and redirects them to this proxy. The collaboration handler described below handles the responses to these requests. All requests and responses are Java objects and take advantage of Java's object serialization capability.

The *collaboration handler* enables multiple clients to collaboratively interact with and steer applications. All clients connected to a particular application form a collaboration group by default. Global updates (e.g. current application status) are automatically broadcast to this group. Clients can form or join (or leave) collaboration sub-groups within the application group. Clients can also disable all collaboration so that their requests/responses are not broadcast to the entire collaboration group. Individual views can still be explicitly shared in this mode. In addition to collaborative interaction/steering, the client portal is provided with chat and whiteboard tools to further assist collaboration.

The *Daemon servlet* forms the bridge between the server and the applications. Each application is authenticated at the server using a pre-assigned unique identifier. The daemon servlet creates an Application Proxy for each new application that connects to it, and maintains a handle to the proxy object. It also assigns the application a unique session identifier. It buffers all client requests and sends them to the application when the application is in the ``interaction'' phase. This ensures that requests are not lost while the application is busy computing.

Section 5 discusses the server architecture again in the context of peer-to-peer network of servers.

4.2 A Middleware Substrate for Peer-to-Peer Integration of DISCOVER Servers

The primary objective of the middleware substrate is to enable integration of the DISCOVER computational collaboratories so that a client can access and interact with all the applications for which it has access privileges, regardless of whether they are local or remote. Having all applications connect to a single DISCOVER server or having a centralized repository of servers are not scalable options. Furthermore, security constraints often prevent applications from connecting to remote servers outside their domain. This is true for applications executing on most high-end resources. Finally, applications typically do not provide standard access interfaces for interaction and steering, and need to be coupled to their server using a proprietary protocol. The proposed peer-to-peer architecture with coupled server/application(s) sets is a more appropriate architecture for such integration.

The DISCOVER peer-to-peer architecture consists of multiple independent collaboratory domains, each consisting of one or more DISCOVER servers, and applications connected the server(s). A discovery mechanism is provided to allow a server to locate remote servers and to access applications connected to those remote servers. As a result, a client can connect to its "closest" collaboratory (using HTTP) and have secure and authorized access to all applications. This is significant as no assumptions are made on the nature of the client-server connection. Furthermore, the client can collaboratively interact with and steer the application in a controlled manner. The access control mechanisms, such as locks, are extended by the middleware to manage local and remote accesses. Finally, clients spanning multiple collaboratories can form virtual communities and collaborate seamlessly.

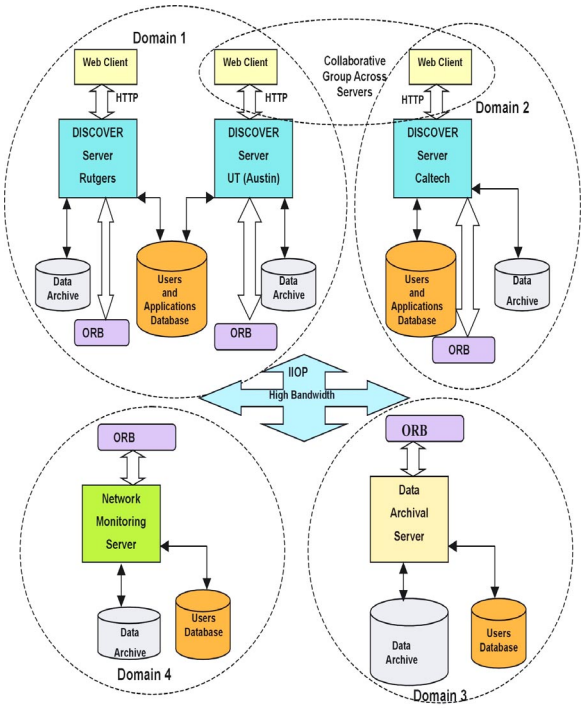

Figure 3. A typical network of servers providing a repository of services

An overview of the DISCOVER network of peer-to-peer servers is presented in Figure 3. As shown in this figure, the middleware can be extended to include other servers and services using the "pool of services" model described earlier. For example, a

middleware can provide access to a monitoring service, an archival service or grid services using Java/CORBA CoG Kits. Note that the availability of these servers is not guaranteed and must be determined at runtime. The middleware substrate builds on CORBA/IIOP, which provides peer-to-peer connectivity between DISCOVER servers within and across domains. Furthermore, server/service discovery mechanisms are built using the CORBA *Trader Servic*e [41]. Although CORBA does introduce some overheads, it enables scalability and high availability and provides the services required to implement the middleware substrate. Moreover, we believe that the servers will be typically connected through reasonable bandwidth links (~100 MB). As no assumptions can be made about client-server connections, having the client connect to the "nearest server", and using CORBA to connect that server and the desired application may actually reduce client latencies.

5 Implementation and Operation of the DISCOVER Middleware Substrate

This section presents the implementation and operation of a CORBA-based prototype of the DISCOVER middleware substrate.

5.1 Middleware Implementation

The middleware substrate builds on the DISCOVER interaction/collaboration server architecture described in Section 4. It implements the 2 interface levels described in Section 3 – the *DiscoverCorbaServer* interface is the level one interface and represents a server in the system while the *CorbaProxy* interface represents an application at a server. These interfaces expose the servers' and applications' functionality to other servers and collaboratories.

The DISCOVER middleware uses the same 3 communication channels between two servers which it uses between a server and an application i.e. a *Main Channel*, a *Command Channel* and a *Response Channel* (see Section 4.1). For interaction between two servers, an additional *Control Channel* is used to forward error messages and system events. The *Control Channel* serves as a notification service similar to the one used in Salamander substrate [28][29][30].

The *ApplicationProxy* object created at the server for each active application encapsulates the entire context for the application including its *CorbaProxy* interfaces.

5.1.1 The *DiscoverCorbaServer* Interface

The *DiscoverCorbaServer* interface is implemented by each server, and specifies methods for interacting with the server including authenticating with the server, getting a list of all active services on the server, and getting a list of users logged on to the server. The *DiscoverCorbaServer* object is the server's gateway for all other DISCOVER servers. It is maintained by the Daemon Servlet (see Section 4). *DiscoverCorbaServer* publishes its availability using the CORBA trader service. It also maintains a table of references to the *CorbaProxy* objects (i.e. *CorbaProxyInterface*) for remote applications. Using this reference, the DISCOVER Daemon Servlet can provide transparent access to remote applications and support local clients' interactions with those applications. Thus all requests for remote applications from locally connected clients go through the *DiscoverCorbaServer*, which then forwards them to the appropriate *CorbaProxyInterface* reference.

5.1.2 The CorbaProxy Interface

The *CorbaProxy* interface represents a service or an application that is active at a particular server. This interface specifies all the methods that are required for accessing, interacting with and steering the application. This includes methods for querying application status, querying and changing application parameters, requesting steering controls (locks) and issuing commands. The *CorbaProxy* interface is therefore an application's gateway for all other servers. All servers that have clients interacting with remote applications maintain a reference to the *CorbaProxy* objects for those applications (as mentioned above, the DiscoverCorbaServer object maintains this table of references). *CorbaProxy* also binds itself to the CORBA naming service using the application's unique identifier as the name. This allows the application to be remotely accessed from any server.

A *CorbaProxy* object is contained within each DISCOVER *ApplicationProxy* object. The *ApplicationProxy* object manages all communication with the application required during application registration, or during interaction and steering with the application. In the case of local applications, *ApplicationProxy* directly communicates with the applications using the appropriate protocol. In the case of remote applications however, this

communication is done with the remote *CorbaProxy* object using its local reference (i.e. *CorbaProxyInterface*).

5.2 Middleware Operation

The overall interaction between DISCOVER servers is summarized in Figure 4. Key operations are described below.

5.2.1 Servers and Applications Discovery

DISCOVER servers locate each other using the CORBA trader services. The CORBA trader service maintains all the server references as *service-offer* pairs. In our prototype we have implemented a minimalist trader service on top of the CORBA naming service. All DISCOVER servers are identified by the service-id 'DISCOVER'. The service offer is a CORBA *CosTrading* module (CORBA Trader service specification), which encapsulates the CORBA object reference and a list of properties defined as name-value pairs. Thus an object can be identified based on the service it provides or its properties list.

Applications are located using their globally unique identifiers, which are dynamically assigned by the *DaemonServlet*. The application identifier is chosen to be a combination of the server's IP address and a local count of the applications on each server. This ensures that even if the same application is connected to multiple servers or multiple instances of an application are connected to the same server, each instance will have a unique identifier. Moreover, the server's IP address can be extracted from this application identifier, making it very easy to determine if the application is a local application or a remote application.

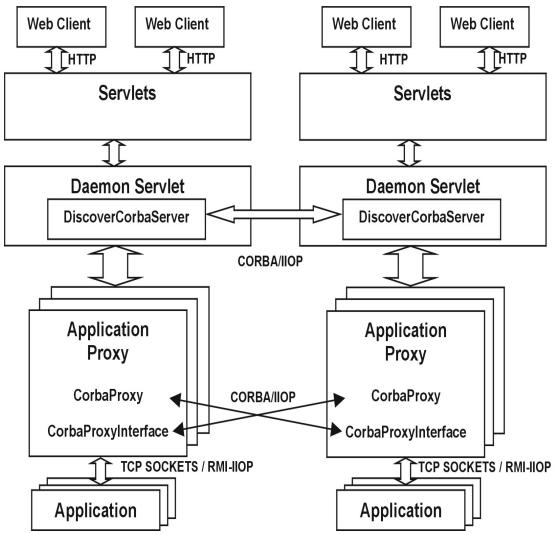

Figure 4. Interaction Model between DISCOVER Servers.

5.2.2 Security/Authentication across Servers

As described in Section 4.1, each DISCOVER server supports a two-level client authentication; the first level authorizes access to the server and the second level permits access to a particular application. To control access, all applications are required to be registered with the server and to provide a list of users and their access privileges (e.g. read-only, read-write). This information is used to create access control lists (ACL) for each user-application pair. For access to remote applications, the security handler uses the *DiscoverCorbaServer* to authenticate the client with each server in the network, and in return gets the list of active applications connected to all the servers to which the user has some access privileges. Once the client selects a remote application, the second level authentication is performed to get a customized interaction/steering interface for the application based on the client's access privileges. As a result each client can access only those applications that it is authorized to, and only it can interact in ways defined by its privileges and capabilities. Note that a client has access only to those servers where he is a registered user – i.e. he is on the authorized user list for at least one of the applications registered with the server. Thus, in the current system, a client's user-Id for a particular application is assumed to be consistent across all servers.

5.2.3 Collaboration across Servers

DISCOVER enables multiple clients to collaboratively interact with and steer applications. As described in section 4.1, the dedicated

collaboration handler servlet handles all collaboration on the server side, while a dedicated thread is used on the client side. All clients connected to an application form a collaboration group by default. These collaboration groups can span multiple servers. In this case, the *CorbaProxy* objects poll each other for updates and responses. The peer-to-peer architecture offers two significant advantages for collaboration. First, it reduces the network traffic generated by reducing the large number of broadcast messages that would be typically sent by a server to all the participants of the collaboration session. This is because, now, instead of sending individual collaboration messages to all the clients connected through a remote server, only one message is sent to that remote server, which then updates it locally connected clients. Since clients always interact through the server closest to them and the broadcast messages for collaborative updates are generated at this server, these messages don't have to travel large distances across the network. This reduces overall network traffic as well as client latencies when the servers are geographically far away. It also leads to better scalability in terms of the number of clients that can be supported within a collaboration session without overloading a server as the collaboration load now spans across multiple servers.

5.2.4 Distributed Locking

Session management and concurrency control is based on capabilities granted by the server. A simple locking mechanism is used to ensure that the application remains in a consistent state during collaborative interactions. This ensures that only one client "drives" (issues commands) the application at any time. In a distributed server framework, locking information is only maintained at the application's host server i.e. the server to which the application connects directly. Servers providing remote access to this application only relay lock requests to the host server and receive locking information from the host server.

5.2.5 Distributed Logging

The session archival handler maintains two types of logs. The first one logs all interactions between a client(s) and an application. This log enables clients to replay their interactions with the applications. It also enables latecomers to a collaboration group to get up to speed. The second log maintains all requests, responses, and status messages for each application. This log allows clients to have direct access to the entire history of the application. For remote applications, the client logs are maintained at the server where the clients are connected, where as the application logs are maintained at its application's host server.

6 DESIGN EVALUATION

6.1 Current Status and Performance

DISCOVER is currently operational and the DISCOVER server network includes a deployment at CSM, University of Texas at Austin, and is being expanded to include a deployment at CACR, California Institute of Technology. We have conducted some preliminary experiments to test the performance of the middleware under high load conditions. The initial tests were conducted within the local area network at Rutgers University. The current middleware can support more than 40 simultaneous applications on a single server. With the peer-to-peer server network in place, the number of simultaneous applications that can be supported should further increase. In another set of experiments, the middleware was able to support 20 simultaneous clients. As we increased the number of simultaneous clients beyond 20, we noticed degradation in performance.

These set of experiments show an interesting trend, which we are further exploring. The server and application interaction used TCP sockets and a custom protocol, while the client and server interaction used servlets over HTTP. The fact that the system is able to support more simultaneous applications than simultaneous clients, illustrates the design trade off between high performance and wide spread deployment when using commodity technologies. We comment more on this in the next subsection.

6.2 Use of Commodity Technologies

The primary goal of the solutions presented in the paper is to support wide deployment and global access; as a result we build on widely used commodity distributed technologies. For example, access to DISCOVER is provided using thin web browsers and the ubiquitous HTTP protocol. Our implementation builds on existing HTTP servers and adds new services, rather than building customized servers from scratch. The choice of CORBA as the middleware substrate is motivated by its inherent support for peer-to-peer interactions. But the use of these commodity technologies is not without

disadvantages and limitations. While the use of HTTP for client-server interactions provides ubiquitous pervasive access, it necessitates a poll and pull mechanism for fetching the data from the server instead of a push mechanism, as HTTP is a request-response protocol. The poll and pull mechanism makes it necessary to maintain FIFO buffers at the server for each client to support slow clients. Such a poll and pull mechanism may be unsuitable for large virtual reality collaborative environments where 3D data is involved, as it presents both memory and performance overheads.

Similarly the use of CORBA as the middleware technology supports peer-to-peer interactions and enables seamless integration with 3rd party custom servers. CORBA, however, causes the middleware to give up control over its transport and communication policies and reduces performance when compared to a lower level socket based system. Furthermore, in our experience, the current CORBA ORBs leave much to be desired, especially in the areas of performance and interoperability.

6.3 Challenges and Open Issues in a Peer-to-Peer Computational Environment

The peer-to-peer integration of DISCOVER servers in Grid environments consisting of multiple institutions and different administrative domains has presented many challenges. We briefly discuss a few of them.

Authentication and Security across servers: Authentication of users and applications across servers presents a significant challenge. DISCOVER tries to minimize global knowledge by having the services or applications identify the users (or user-IDs) that have access and their access privileges. Thus when an application or a service registers with a server, it supplies the server with this information in the form of a list of authorized users-IDs and their privileges. Once a user-ID is supplied, a server will automatically authenticate that user-ID. Thus, even though the system assumes consistent user-IDs across all servers, this is hardly a problem, as the user-IDs do not belong to a server but to an application/service. In the current implementation, the user is authenticated at his home server before he can proceed. One way to get around this problem is to have a centralized directory service like the GIS that maintains user-IDs and other global information. All the servers in the system can now use this directory service.

Resource utilization: It is important to account for the resources used by any remote server. Currently, the system does not track the use of resources. It is, however, possible to add control mechanisms by creating access policies for each server, and then restricting each server's use of resources according to that policy. The access policies can be added to the current ACLs and can be defined in terms of metrics like number of requests per second, or the data bytes being transferred to each server per second.

Data Management and ownership across servers: In a peer-to-peer environment enabling interaction and steering of remote applications, the management of the generated data becomes important. The current implementation of DISCOVER avoids these issues by using Relational Databases to store all the data generated in the form of records. Access to these databases is provided through customized interfaces and is protected through passwords. The data produced by the application/service in response to clients' requests is handled by the local server (i.e. the server to which the clients are directly connected) and the local server creates the output files or the records under the ownership of the user who requested that data. The periodic data generated by the applications/services are handled by the home server of the application (i.e. the server to which the application is directly connected) and records are created under the ownership of the user-id who owns the application. All the clients, who have access privileges to this application, are also provided with read only access rights to these records. Thus, the peer-to-peer architecture doesn't allow creation of files/records on a remote server by the clients.

Some of these issues are still open and we are in the process of addressing them. Note that the current implementation of DISCOVER is a prototype aimed at evaluating the viability of the peer-to-peer architecture and current commodity technologies for addressing the requirements of current and emerging Grid applications.

7 Conclusion and Future Work

This paper presented the design, implementation, and operation of a middleware substrate that enables peer-to-peer integration of and global collaborative (web-based) access to multiple, geographically distributed instances of the DISCOVER computational collaboratory for interaction and steering. The substrate builds on the CORBA distributed object technology and enables dynamic application/service discovery on the Grid, remote authentication and access control, coordinated interactions for collaborative interaction and steering.

The DISCOVER middleware architecture is currently operational and provides collaborative

interaction and steering capabilities to remote distributed scientific and engineering simulations, including oil reservoir simulations, computational fluid dynamics and numerical relativity. The DISCOVER server network currently includes deployments at CSM, University of Texas at Austin, and is being expanded to include CACR, California Institute of Technology. In addition to addressing the issues discussed above, we are currently evaluating this framework to determine response latencies and throughput for remote applications as compared to multiple applications connected to the same server. We are also measuring the overheads incurred for application/service discovery and for remote authentication. Finally we are evaluating the scalability of the architecture and the limitations of the underlying technologies.

In a related research effort we are building a CORBA CoG kit to provide application developers with access to Grid services using CORBA [43][2]. The overall goal is to integrate Grid services provided by CORBA with the collaborative interaction and steering services provided by DISCOVER. For example a client can use Globus services provide by the CORBA CoG Kit to discover, allocate and stage a scientific simulation, and then use the DISCOVER web-portal to collaboratively monitor, interact with, and steer the application.

REFERENCES

[1]. R. T. Kouzes, J. D. Myers, and W. A. Wulf, "Collaboratories: Doing science on the Internet", IEEE Computer, Vol.29, No.8, August 1996.

[2]. I. Foster and C. Kesselman, "The Grid: Blueprint for a New Computing Infrastructure, Morgan Kaufmann", San Francisco, 1998.

[3]. The 1st Global Grid Forum, March 2001, Amsterdam, Netherland, http://www.ggf1.nl.

[4]. Grid Computing Environments Working Group, Global Grid Forum, http://www.computingportals.org.

[5]. S. Subramanian, G.R. Malan, H.S. Shim, J.H.Lee, P. Knoop, T. Weymouth, F. Jahanian, A. Prakash, and J. Hardin, "The UARC web-based collaboratory: Software architecture and experiences", IEEE Internet Computing, Vol.3, No.2, pp.46-54, 1999. See also: http://intel.si.umich.edu/sparc/.

[6]. J. H. Lee, A. Prakash, T. Jaeger, and G. Wu, "Supporting multi-user, multi-applet workspaces in CBE", Proc. of the ACM 1996 Conf. on Computer Supported Cooperative Work (CSCW'96), Cambridge, MA, pp.344-353, November 1996.

[7]. C. M. Pancerella, L. A. Rahn, and C. L.Yang, "The diesel combustion collaboratory: Combustion researchers collaborating over the Internet", Proc. of IEEE Conference on High Performance Computing and Networking, Portland, OR, November 1999.

[8]. R. A. Whiteside, E. J. Friedman-Hill, and R. J. Detry, "PRE: A framework for enterprise integration", Proc. of Design and Information Infrastructure Systems for Manufacturing (DIISM), Fort Worth, TX, May 1998.

[9]. Argonne National Laboratory. Access Grid. Online at: http://www-fp.mcs.anl.gov/fl/accessgrid/

[10]. Protein Data Bank Research Collaboratory for Structural Bioinformatics. http://www.rcsb.org/pdb/

[11]. The EMSL Collaboratory. http://www.emsl.pnl.gov:2080/docs/collab/.

[12]. Cactus Computational Collaboratory. http://www.cactuscode.org.

[13]. M. Russell, G. Allen, G. Daues, I. Foster, T. Goodale, E. Seidel, J. Novotny, J. Shalf, W. Suen, and G. von Laszewski, "The Astrophysics Simulation Simulation Collaboratory: A Science Portal Enabling Community Software Development". Proceedings of Tenth IEEE International Symposium on High Performance Distributed Computing, August 2001 (submitted).

[14]. DISCOVER (Distributed Interactive Steering and Collaborative Visualization EnviRonment), http://www.discoverportal.org.

[15]. S. Kaur, V. Mann, V. Matossian, R. Muralidhar, M. Parashar, "Engineering a Distributed Computational Collaboratory", 34th Hawaii Conference on System Sciences, January 2001.

[16]. "CORBA: Common Object Request Broker Architecture", http://www.corba.org.

[17]. HyperText Transfer Protocol (HTTP), http://www.w3.org/Protocols/

[18]. I. Foster, C. Kesselman, S. Tuecke, "The Anatomy of the Grid: Enabling Scalable Virtual Organizations", Intl. J. Supercomputing Applications, 2001

[19]. I. Foster, "Internet Computing and the Emerging Grid", Nature Web Matters, (http://www.nature.com /nature/webmatters/grid/grid.html) 2000.

[20]. N. H. Kapadia and J. A. B. Fortes," PUNCH: An Architecture for Web-Enabled Wide-Area Network-Computing", Cluster Computing: The Journal of Networks, Software Tools and Applications; special issue on High Performance Distributed Computing. September 1999.

[21]. N. H. Kapadia, R. J. Figueiredo, and J. A. B. Fortes, "PUNCH: Web Portal for Running Tools", IEEE Micro, May-June 2000.

[22]. D. Bhatia, V. Burzevski, M. Camuseva, G. Fox, W. Furmanski, and G. Premchandran, "WebFlow - A Visual Programming Paradigm for Web/Java Based

[2] www.caip.rutgers.edu/TASSL/CorbaCoG/CORBACog.htm

Coarse Grain Distributed Computing", Presented at Workshop on Java for Computational Science and Engineering Workshop, Syracuse University, December 1996.

[23]. E. Akarsu, G. Fox, T. Haupt, A. Kalinichenko, K. Kim, P. Sheethaalnath, and C. H. Youn, "Using Gateway System to Provide a Desktop Access to High Performance Computational Resources", 8th IEEE International Symposium on High Performance Distributed Computing (HPDC-8), Redondo Beach, California, August, 1999.

[24]. HotPage User Portal- https://hotpage.npaci.edu/

[25]. M. Thomas, S. Mock, and J. Boisseau, "Development of Web Toolkits for Computational Science Portals: The NPACI HotPage", The 9th IEEE International Symposium on High Performance Distributed Computing (HPDC 2000), Pittsburgh, Aug. 1-4, 2000.

[26]. SDSC GridPort Toolkit - http://gridport.npaci.edu/

[27]. Gregor von Laszewski, Ian Foster, Jarek Gawor, Peter Lane, Nell Rehn, and Mike Russell, "Designing Grid-based Problem Solving Environments and Portals", Proceedings of the 34th Hawaii International Conference on System Sciences, January 2001.

[28]. G. R. Malan, F. Jahanian, and S. Subramanian, "Salamander: A Push-based Distribution Substrate for Internet Applications", Proceedings of the USENIX Symposium on Internet Technologies and Systems, December 1997, Monterey, CA.

[29]. G. R. Malan, F. Jahanian and P. Knoop, "Comparison of Two Middleware Data Dissemination Services in a Wide-Area Distributed System", Proceedings of the 17th IEEE International Conference on Distributed Computing Systems, May 1997, Baltimore, MD.

[30]. G. R. Malan, F. Jahanian, C. Rasmussen, and P. Knoop, "Performance of a Distributed Object-Based Internet Collaboratory", Technical Report CSE-TR-297-96, University of Michigan EECS Deptartment, July 1996.

[31]. Internet Performance Measurement and Analysis (IPMA) project homepage, http://nic.merit.edu/ipma/

[32]. Napster, http://www.napster.com/ (2000).

[33]. Gnutella, http://gnutella.wego.com/ (2000).

[34]. I.Clarke, O. Sandberg, B.Wiley, and T.W. Hong, "Freenet: A Distributed Anonymous Information Storage and Retrieval System", ICSI Workshop on Design Issues in Anonymity and Unobservability, 1999.

[35]. Universal Description Discovery and Integration (UDDI), Technical White Paper, http://www.uddi.org, September 6,2000.

[36]. Microsoft .NET, http://www.microsoft.com/net/

[37]. Intel Proposals on Peer-to-Peer Computing, http://www.intel.com/ebusiness/products/peertopeer/index.htm

[38]. J. Hunter, "Java Servlet Programming", 1st edition, O'Reilly, California (1998).

[39]. Java Servlet API Specification, http://java.sun.com/products/servlet/2.2/.

[40]. Java Remote Method Invocation, http://java.sun.com/products/jdk/rmi.

[41]. CORBA Trader Service Specification, ftp://ftp.omg.org/pub/docs/formal/97-07-26.pdf.

[42]. M. Baker, R. Buyya and D. Laforenza, "The Grid: A Survey on Global Efforts in Grid Computing", ACM Journal of Computing Surveys, 2000 (submitted).

[43]. S. Verma, J. Gawor, M. Parashar, and G. von Laszewski, "A CORBA Commodity Grid Kit", Submitted to the 2nd International Workshop on Grid Computing, November 2001.