A Data Handling Architecture for a Prototype Federal Application

Chaitanya K. Baru, Reagan W. Moore, Arcot Rajasekar, Wayne Schroeder, Michael Wan
San Diego Supercomputer Center, 10100 Hopkins Drive, La Jolla, CA 92093
(baru,moore,sekar,schroede,mwan@sdsc.edu)

Richard L. Klobuchar, David M. Wade
Science Applications International Corp.(SAIC)

Randall K. Sharpe, Jeff Terstriep
National Center for Supercomputing Applications, Urbana-Champaign, Illinois
(rsharpe,jefft@ncsa.uiuc.edu)

Abstract: The Distributed Object Computation Testbed (DOCT) Project is a collaboration led by the San Diego Supercomputer Center (SDSC) and funded by the Defense Advanced Research Projects Agency (DARPA) and the US Patent and Trademark Office (USPTO). The DOCT project was initiated with the objective of creating a testbed system for handling complex documents on geographically distributed data archives and computing platforms. The project focuses on technologies that apply to the information needs of federal agencies, such as the National Science Foundation, the National Institutes of Health, the Nuclear Regulatory Commission, the Environmental Protection Agency (EPA), the Department of Energy, and the Department of Defense. In particular, the patent filing and amendment processing application of the USPTO is used as the prototype application for this testbed. This paper describes the DOCT system architecture and the USPTO application.

1. Introduction

The Distributed Object Computation Testbed (DOCT) enables the study of distributed processing issues in a geographically dispersed, heterogeneous computing environment. The DOCT project has been studying issues in implementing robust information processing systems in such environments. The testbed includes distributed computational and storage resources interconnected via high-speed networks, including OC-3 to OC-12 class networks. The computer systems which comprise the testbed include a 23-node IBM SP-2 at SDSC and an SGI PowerChallenge at the National Computational Science Alliance (NCSA) in Illinois; the High Performance Storage System (HPSS) archival system at SDSC[1] and at the California Institute of Technology (Caltech); database management systems at the offices of the Science Applications International Corporation (SAIC) in Virginia, SDSC, and NCSA; and, the vBNS and AAI national networks through NSF, ARPA, and the Naval Command, Control and Ocean Surveillance Center (NCCOSC) in San Diego. Software systems that have been prototyped as part of this project include, a Java-based framework for developing and deploying distributed software agents[2]; a Storage Resource Broker (SRB) middleware, which provides uniform access to distributed, heterogeneous storage systems[3]; the Network Queueing Environment (NQE) scheduling system[4] and the Network Weather Service (NWS) network monitoring system[5]; and authentication, encryption, and intrusion detection systems[6]. The testbed also includes a number of commercial off-the-shelf software systems such as the Texcel document management system[7], the OpenText text indexing/search system, and Oracle[8], DB2[9], and ObjectStore[10] database

management systems. A prototype version of DB2 is also used, which integrates DB2 with HPSS[12].

The USPTO application requires support for electronic filing, searching, and archiving of complex, multi-mode documents in a geographically distributed system. In addition, there is a need to provide a workflow system to automate office operation; maintain an archival audit trail of all office actions; provide Web-based access to the published patent database; and, convert legacy patent data. These application requirements are similar to those found in a number of other federal agencies. To support these requirements, the DOCT infrastructure provides mechanisms for handling very large, complex databases in a robust, fault-tolerant, metacomputing environment. In addition, we have also developed prototype software specifically to support secure electronic filing of documents; document validation; document classification; and, data mining.

The next section discusses the DOCT system in terms of the document flow for the USPTO application, and also highlights the DOCT data handling systems. Section 3 discusses the security infrastructure of the DOCT testbed. Section 4 describes the agent framework, which allows easy development of client applications in this environment. Finally, Section 5 provides a summary.

2. Document Flow

Figure 1 shows the various DOCT components involved in supporting the USPTO application. The following subsections describe the document flow in detail, beginning with the arrival of the patent application at the *Electronic Mailroom*, and ending with the publication of the final patent in the Patent Database.

2.1 The Electronic Mailroom

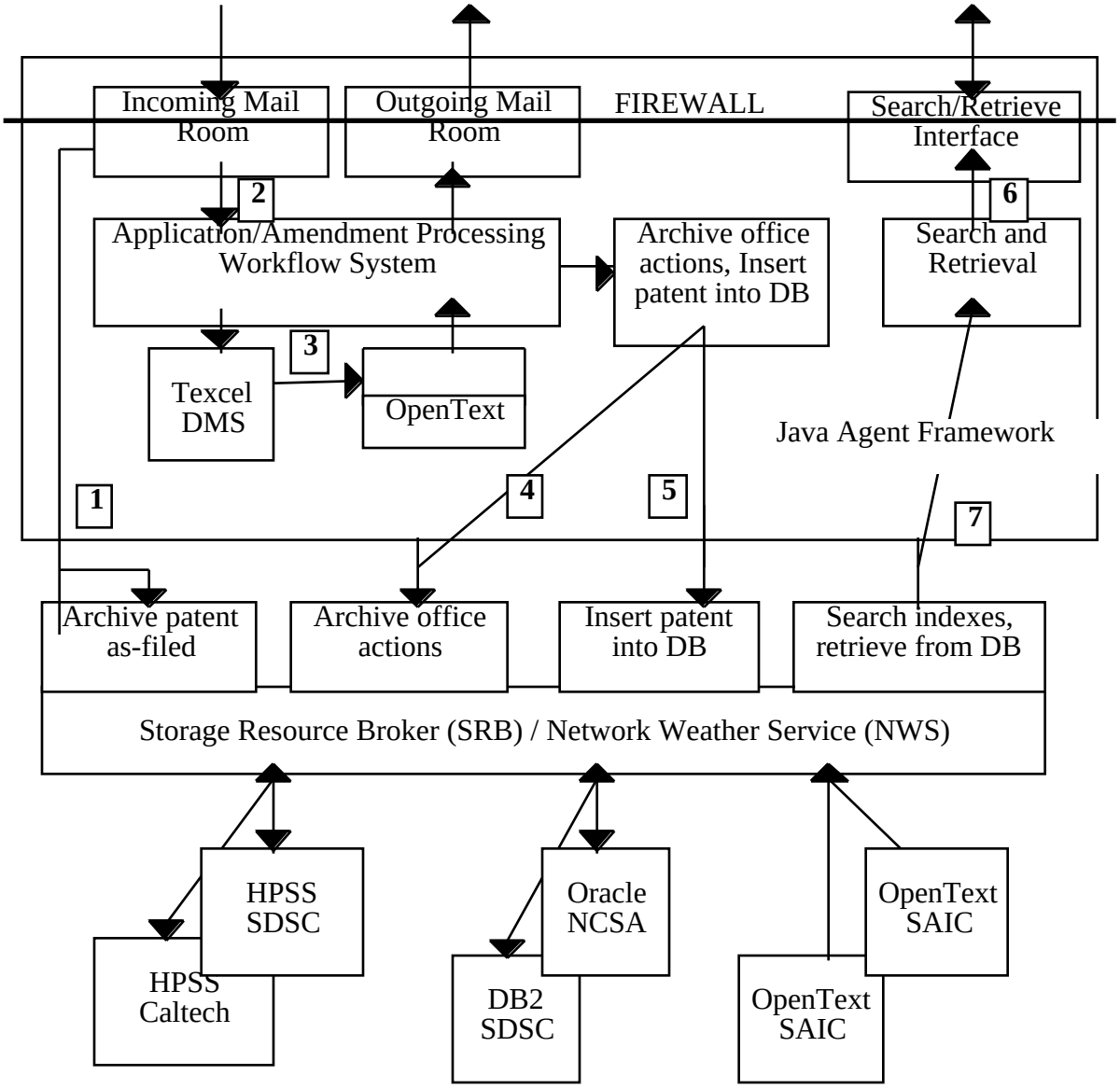
The key requirement of the USPTO application is the ability to support electronic filing of patent applications and amendments via the Internet. Applications and amendments arrive at the *incoming Electronic Mailroom*. Text files are in SGML form while images are in tif format. File and document validation is performed on all received files. Validated files are sent across the security firewall to the internal workflow systems (see Section 2.2). The security aspects associated with these communications are described in Section 3.

During application processing, there may be several "office actions" initiated by patent examiners. These include requests for further information from the applicants and amendments to the patent application, such as deletion of claims. Each of these office actions results in communication with the patent applicant. Communications originating from USPTO to the applicant are routed via the *outgoing Electronic Mailroom.* Each office action, including type of action and date/timestamp, is recorded and archived along with the application, in order to maintain an audit trail of office actions.

2.2 Application Processing Workflow

Validated applications are transferred from the Electronic Mailroom to the internal systems, which operate behind a security "firewall". Here, the patent application is first classified based upon its subject matter. This determines the area/group within USPTO that will be responsible for examining this application. Each area within the USPTO is associated with a separate workflow. The application is inserted into the corresponding workflow (Step 2 in fig.) and, simultaneously, it is indexed by a text indexing system to

enable efficient text-based searching of active applications and to support *interference searching* (Step 3 in fig). An interference search is used to determine if a newly submitted patent application has overlapping claims with any of the other currently active patent applications.



2.2.1 Document Classification

We undertook a study at NCSA to evaluate the potential of employing neural network-based techniques for automatic classification of patent applications. The project investigated the possibility of using *Kohonen Self-Organizing Maps (SOM)*[13] in the "reclassification" problem and the use of *Learning Vector Quantization (LVQ)* technique[14] in the "preemptive classification" problem.

The SOM technique is based upon *feed-forward* neural networks. It lends itself to "unsupervised" clustering of existing data, from which classification labels may be obtained. Using this approach, it is possible to collect together documents with similar content, while separating them from other documents with dissimilar content. LVQ is based upon the *feed-forward back-propagation* neural network and lends itself to "supervised" targeting of new data into pre-existing classification schemes. It is good at evaluating document content based on its context, and is able to use this evaluation to quantify document similarity. We used the public-domain implementations of the SOM and LVQ implementations, viz. SOM-pak and LVQ-pak which are available from the Neural Network Research Center[15]. Further discussion and descriptions of this software, including source code, can be found at this location.

For patent documents, the *Claims* and *Abstract* sections provide the most relevant content for the purposes of classification. Therefore, the concatenation of text from these two sections is used for classification. For SOM, the entire collection of patent documents (from 1976-1997) is used as the input for *feature vector* generation. For LVQ, a random sample of 10 documents from each USPTO-defined patent class was chosen as the input data. We employed the "histogram" method for document encoding, based on successful results from other studies conducted at NCSA[16].

Except for the difference in the choice of document corpus, the method used for creation of document feature vectors for SOM and LVQ was the same. For each document, each input word is checked against a "stop word" list, and words in this list were ignored. If the word is not a stop word, it is then *stemmed* using the stemming procedure provided in FreeWAIS[17]. This stem is then put into two count lists. The *global count list*, which keeps track of the number of documents in which a stem occurs, and the *document count list*, which keeps track of the number of occurrences of each stem within the given document. After a document is completely processed, its document count list and associated document label are pushed onto a stack. This process continues for all of the input documents. When finished, the result is a global count list that contains every distinct stem that occurred in any document, and the number of documents in which that stem occurs. Also, for each document, there is a document count list which contains every distinct stem in that document as well as the number of occurrences of that stem in the document.

Once this global information has been gathered, the *normalization* process begins. This process yields the actual feature vectors. First, the global count list is sorted in descending order of the counts. The first M items are removed from this list, where M is a predetermined number. Then the first N stem/count pairs are taken, where N is again predetermined. These provide the foundation for the normalization/projection of the document count lists. The combination of the word stemming and the "first N offset M" stem choices is used as the "blurring" agent. Each document count list is then parsed for occurrences of the stems in this reduced global count list, in the order that they occur in the global count list. If the *i-th* stem in the global count list occurs in the document count list for a given document, then the count associated with that stem is the *i-th* component of the raw feature vector for that document. Otherwise, a zero is placed in the *i-th*

component. After all of the N stems of the global word count have been checked, the raw vector is normalized to unit length and then paired with the associated document label. This process continues for each of the document count lists. Each of the resulting vectors in pushed onto a stack and this stack is returned as the collection of feature vectors and associated labels for the input document corpus. These features and their associated labels are then written out to disk to be used subsequently in the SOM or LVQ processes.

The SOM approach was applied to the collection of patent applications discussed above, as an attempt to aid in the resolution of the classification problem faced by the USPTO. The idea is to use the output of SOM classification as a decision support tool in the effort to modify the classification scheme. Over the course of several years, there is the possibility that, as a result of changes in science, technology, and engineering, there is *content drift* within a classification, as well as "over stacking" of a classification. Sub-fields of specialization may arise within existing fields of invention and new fields of invention may be created. This may require splitting of some existing classes and subclasses, and may cause further reclassification. In this project, we specifically studied the clustering of documents within two existing classes, to examine the possibility of creating further subclasses. Each of the sets of documents for each class was treated as a separate corpus and each document in each class was labeled by its current subclass. The results from this classification were made available as a VRML-based visualization, which shows proximity between classes, as derived by the neural network-based system. The results of this study are currently under evaluation by the USPTO.

The LVQ approach has been incorporated in the DOCT testbed, as an integral part of the document workflow. A neural network was trained using the *LVQ-pak* with the LVQ corpus described earlier. Each node is associated with a class and each feature vector of the training set was labeled with its pre-established class. After the network was randomly initialized, the network was trained by first running the features through the OLVQ1 training algorithm and the network was further conditioned by tuning it with the same input vectors and a chain of training algorithms (from LVQ3 to LVQ1). The resulting neural network became the basis for the *presumptive classification* portion of the workflow.

In presumptive classification, a new document (i.e. a new application) would be submitted to a function which encodes the document using the same global document count used by the training. Next, the vector (without a label, since it is yet unclassified) is compared with all the nodes in the neural network. The closest *n* nodes (where the value of *n* can be specified as input) are found in order, and the associated labels are returned. These labels can then be used to classify the document. If the results from this classification are deemed incorrect, the patent examiner can resubmit the document with an explicitly specified label. In this case, the neural network retrains itself using the new input vector. The effectiveness of this mechanism is also under study.

In summary, automated classification can be used both as a means of creating new classifications in existing data, as well as an aid to classifying new data into pre-existing classifications.

2.3 Archiving Data

There are two points in the document dataflow where data is archived. First, the patent application is archived as soon as it is validated in the incoming Electronic Mailroom (Step 1 in fig.), to satisfy the legal requirement of archiving all patent applications "as filed". Second, after a patent application has gone through the entire workflow, the final application is archived, along with all related office actions (Step 4 in fig). In addition,

the final patent is also inserted into the Patent Database (Step 5 in fig). An office action is any action in the workflow related to the patent application, including electronic mail communication with an applicant. For each office action, the workflow system obtains a date/timestamp and a unique hash record from a digital notary service. This is stored as auxiliary information related to the patent application. By archiving all office actions, we can provide an audit trail for future use, either for legal discovery, or other investigations and studies.

The DOCT infrastructure incorporates the SDSC *Storage Resource Broker (SRB)* middleware[3], which is used for data storage and archiving. The SRB uses a metadata catalog, MCAT, to record information on where the data sets are located. Using this information, it is able to provide seamless access to heterogeneous, distributed storage systems. MCAT stores information about the contents of the database as well as the "system-level" information needed to access the database. It also implements the concept of *logical storage resource*, which is used by the SRB to support data replication across multiple *physical storage resources.* A logical storage resource may contain two or more physical resources. Any data that is written/stored into a logical resource is replicated across the corresponding physical resources. When reading data from a logical resource, the SRB picks any one of the component physical resources. The user also has the option of specifying a particular physical resource. The patent application "as filed" and the final application are archived in an SRB logical resource containing an HPSS system at SDSC and another HPSS system at Caltech. The SDSC HPSS installation includes a 23-node IBM RS/6000 SP, 1TB disk cache, and two tape robots with a total capacity of 120TB. The Caltech HPSS system runs on an IBM RS/6000 workstation.

2.4 The Published Patent Database

Published patents are stored in an SRB logical resource containing two databases, one of which is an Oracle database at NCSA and the other is an IBM DB2 database at SDSC. Both databases employ the same database schema to represent patent documents. Patent information fields such as, Title, Inventor, Organization, Legal Representative, are all stored as tables in a relational database. In addition, the SGML text of the entire patent application, and all related tif images, are stored as *binary large objects* within the databases. This scheme allows full *ad hoc* query capability on various information fields within a patent, as well as the ability to easily extract patent images, or the complete patent in SGML form. To provide full-text search capability, the Patent Database is also indexed using a text indexing engine.

While the schemas are the same for the Oracle and DB2 databases, the implementations are different. The data in the Oracle database at NCSA is stored entirely on disk. In the DB2 system at SDSC, the structured columns (i.e. integer, character columns) in the tables are stored on disk, while the text and image objects are stored in HPSS. This is done using a prototype version of DB2, which provides transparent access to HPSS by allowing HPSS files to be defined as DB2 *tablespace containers*. This allows users to create database tables in which data for some columns are stored on disk, while data for other columns are stored in HPSS.

2.4.1 Database Interface

We have provided a Web-based search interface to the Patent Database (Step 6 in fig). The interface allows users to specify search conditions for retrieving patents of interest. An appropriate strategy is selected for answering the query, either by dynamically generating a SQL statement which is issued against the relational database, or by using text indexes to perform full text search (Step 7 in fig). For accessing the SGML text and

the images corresponding to a patent, we have experimented with two different approaches. In one, we use the prototype DB2/HPSS system mentioned above, and in the other, we use the SRB to provide seamless access to data stored in HPSS. The latter case works as follows. First, the search interface issues a query to MCAT, to determine which available databases contain patent data. Next, it issues a query to determine available access functions for the database, which can retrieve individual patents and generate corresponding HTML pages for display. We have implemented one such function where the HTML pages generated contain links to the patent inventors and to other patents referenced by the current patent. Links are also included for accessing the *Brief Summary* and *Detailed Description* sections of a patent, and any related images. Following any of these references, causes a corresponding SQL query to be executed for retrieving the relevant information from the patent database.

3. Security

Computer security is of central importance in a distributed system such as DOCT, especially when the application involves sensitive information. Modern computing practices have begun to address these needs, and several systems are now available that can be combined to create reasonably secure metacomputing systems. These systems include PGP[18], SSL[19], HTTPS[20], the Secure Shell[21], Kerberos[22], intrusion detection software[23], integrity checking tools, and underlying encryption technology such as RSA, DES, and RC5[24].

Since the economic value of patent application information is high, strong security and encryption techniques are needed to protect this information from illegal access. The software systems must be able to withstand strong attacks from third parties. Among other requirements, this may require the use of relatively long encryption keys to reduce the chance of messages being decrypted using exhaustive analysis. Our security research and prototype effort in DOCT has focused on three areas: secure electronic filing, secure infrastructure, and intrusion detection/response. These are described briefly in the following subsections. A more complete description is available in the DOCT Goal Security Architecture document[6].

3.1 Secure Electronic Filing

Secure electronic filing via the Internet is supported by means of a certificate-based security scheme. The steps involved are, (1) *Acquire a certificate from the DOCT certificate authority*. The certificate authority restricts access to approved individuals. The applicant connects via a secure web browser (HTTPS), with one-way authentication (and encryption), to the Certificate Server and requests a certificate. After manual approval, the applicant is added to the access control database, and an X.509 certificate[25] is generated and sent by electronic mail to the user, using PGP-encryption. (2) *Register with USPTO*. Using the X.509 certificate, the applicant connects to the DOCT Filing Web server, via a two-way authenticated HTTPS session. The applicant completes a registration form (via a Java application) which is then submitted to the Web server. The registration is processed, and an ID number is assigned and recorded, and an electronic mail message is sent to the applicant. This ID is used with all future filing activities. (3) *Download data.* The applicant uses the X.509 certificate to download additional information such as, filing application information, PTO Public Key, and FTP userid/password for application submission. (4) *Submit application.* First, the applicant sends his/her PGP-generated public key to the USTPO. Next, the electronic filing software guides the user through the application submission procedures. All files that are part of the application are compressed (via WinZIP), signed, encrypted, and transferred to

the USPTO FTP server, which is write-only to general users. An electronic mail notification is returned from the USPTO to the applicant. The application is then validated and a second electronic mail message is sent to the applicant. The application is then routed to the internal systems where it is processed, as described earlier in Section 2.

3.2 Secure Infrastructure

At each step of the document dataflow of Figure 1, an appropriate security mechanism is employed, to satisfy the security requirement of that particular step. When the patent application first arrives at the electronic mailroom from the Internet, we use the secure electronic filing mechanisms described above. The other place where the system interacts with the external world (i.e., Internet) is at the search interface (Step 6 in fig). Here, a secure HTTP protocol (HTTPS) is used to support secure connections to the system. For internal DOCT communications, both wide-area as well as local-area, we employ the SDSC SEA security system described below. Since these communications occur mainly among Java-based software agents, an alternative is to employ Java-based encryption. One option would be to use a Java DES encryption implementation, based on a shared secret key. This would provide security against network monitoring, although the secret key protection itself is not strong. Another option would be to use a Java SSL implementation, which would provide the necessary network security, as well as better key protection by exchanging the symmetric key, e.g. DES, via RSA.

3.2.1 The SDSC Encryption and Authentication (SEA) Library

The SEA library is a strong but light-weight authentication and encryption package similar to Kerberos or SSL but tailored specifically for metacomputing and high performance computing environments. For example, it provides non-time-limited credentials, which are suitable for use in batch queuing and metacomputing environments, and multiple trust models for initial registration of users including, *password, trusted host,* and *self-introduction* schemes. The SEA library is based on RSA and RC5 encryption technologies and includes RSA key management components. It provides user/process authentication and encryption capabilities between two processes communicating via TCP/IP sockets. Authentication is accomplished via an RSA challenge/response using a random key value. Encryption is accomplished via an RSA exchange of a random key, which is used for RC5 encryption/decryption. The default SEA encryption/authentication configuration uses 512-bit RSA keys and 14-round RC5, with SDSC-added Cypher Block Chaining. This is meant to provide a moderately strong scheme, which is relatively efficient. This can be modified to provide stronger encryption, using longer RSA keys and/or more RC5 rounds.

The SEA library is available on a wide variety of Unix systems including, SunOS, Solaris, IRIX, DEC OSF1, AIX, Sun/Cray CS6400, and the Cray C90 and T3E (this includes a port of RSAREF 2.0 to the Cray C90 architecture). It is used by the SRB client/server software to (1) authenticate clients to servers, across a TCP/IP socket, (2) authenticate between SRB servers and, (3) to, optionally, encrypt control and data communications.

3.3 Intrusion Detection/Response

Regardless of the security mechanisms employed, a system must also incorporate defensive mechanisms, such as intrusion detection and response, to account for the situations when the security systems may have been compromised. The DOCT system incorporates intrusion detection software as the final defensive mechanism to detect

security attacks. The intrusion detection and analysis tools employed include well-identified security contacts at each site, secure (encrypted) electronic mail between security contacts, enabling of system logs at every site, and use of integrity checking tools such as Tripwire[26]. In addition, the system also employs centralized logging using the PICS syslog facility[27]; integrity and configuration checking tools, such as MD5[24], COPS[28], Tiger[29]; and additional analysis tools such as *lsof*[30] and *ifstatus*[31]. More details on the security architecture and intrusion detection/response are provided in the DOCT Goal Security Architecture document[6].

4. Software Agent Framework

To support application development, the DOCT system provides a Software Agent Framework (SAF) along with a software Workbench Server, which provides clients easy access to various DOCT system services. The DOCT SAF is a software layer which operates on top of the DOCT data handling and security systems. It consists of a set of agent *base classes*, an agent communications infrastructure, and a collection of APIs that provide access to the various services in the metacomputer testbed. In addition, the SAF provides interfaces to external clients and software agents, which allow the clients/agents to connect into the framework and communicate with other agents in the framework.

The DOCT Workbench Server provides a Java-based interface for connecting clients to the metacomputer testbed. The clients are Java-based applications, which log in with the Workbench Server upon first connecting to the system, and which use services offered by multiple other agents. The Workbench Server performs user/client authentication upon first connect, and subsequently also monitors the status of agents in the system. At first connect, a client receives an X.509 security certificate, which allows it to access and use other agent resources. A Workbench Service can be a general use interface (e.g., monitoring status of the metacomputer), or a specific application used within the workflow and document management functions provided by the metacomputer (e.g., an in-box and electronic file wrapper for a particular document that is being processed).

4.1 Types of Software Agents

Software agents may be *transient* or *persistent*. Transient agents are created to respond to specific requests and are terminated upon completion of that request. Persistent agents run as *daemons* and can be of two further types. First, persistent agents with "internal" services are those in which the services provided are executed within the single agent process that is started. Second, persistent agents with "internal and transient" services are those where services provided may be executed within the original persistent agent process, or transient processes may be spawned to perform the requested services, or both. The transient processes can be run within the metacomputer or across the Internet, as appropriate.

Each agent has a set of base classes that allows for common security, communications, monitoring, and persistent storage capabilities. The SAF provides a scheduler and interfaces for starting and running agents in the metacomputer. The agents implement specific tasks and intelligence models in Java or C++, depending on the tools with which the agent communicates. The client component of the agent is always written in Java, allowing for a lightweight, heterogeneous client. The Java clients can be combined to create DOCT Client Workbenches for specific classes of users of the system. Clients communicate with the corresponding servers incorporated in the Workbench Server, which serves as an *agent factory*. The agent client/server communication is handled using the Java Remote Method Invocation (RMI) system.

On the server side, both Java and C++ server components may be needed, depending upon the server functionality. In general, a server-side component is needed for each distinct server operating environment. For example, if the server supports a Legion environment, then a Legion component would be present. Communication between the Java and C++ server components is currently implemented either with Java Native Interface (JNI) calls, or using a standard socket interface, with Java sockets on the client and C++ sockets on the server. It is also possible to create CORBA interfaces to the agents. This would allow for interfaces other than sockets to be used for Java/C++ integration.

4.2 Example: Workflow Agent

An example of a SAF agent is the Workflow Agent, which provides access to the workflow information and the patent *Application Database*, which is the database of all currently active patent applications. The workflow information and application database are managed by the Texcel Information Manager, which provides C language APIs to access documents and workflow information within Texcel. To support the flexibility of having remote Java agents and clients access the workflow and Application Databases from anywhere in the metacomputer system, we had to extend the Texcel APIs by creating a socket interface for accessing the API functions. In addition to this Workflow Agent, several other agents have also been created in DOCT, to support various features mentioned earlier such as electronic "wrapping" of patent applications, archiving of audit trail information, and patent search.

5. Summary

This paper described the DOCT project and the USPTO patent and amendment processing application that was implemented using the prototype distributed, heterogeneous testbed provided by DOCT. The project required distributed development of software across geographically dispersed locations (California, Illinois, Virginia), and across multiple companies and organizations. The project has resulted in the construction of a distributed, heterogeneous, metacomputer testbed containing geographically dispersed computational and storage resources interconnected via high speed networks. Prototyping the USPTO application in this testbed demonstrated the feasibility and advantages of employing such a testbed for processing of complex documents.

Acknowledgements

We thank Larry Cogut and Pamela Rinehart, of the USPTO, for their constant guidance and help in this project. A number of other USPTO employees have also helped in defining the various applications requirements. We also acknowledge the help and support provided by Gary Koob of DARPA. The DOCT project is funded jointly by DARPA and the USPTO under Project F19628-96-C-0020.

References

[1] The High Performance Storage System (HPSS), http://www.sdsc.edu/HPSS/.

[2] *Agent Framework Requirements*, Technical Report G1-1, DOCT Project, SDSC, 10100 Hopkins Drive, La Jolla, CA 92093.

[3] The Storage Resource Broker, http://www.sdsc.edu/SRBhello/.

[4] *Evaluation of Queuing Systems*, Technical Report F1-1, DOCT Project, SDSC, 10100 Hopkins Drive, La Jolla, CA 92093.

[5] The Network Weather Service, http://www.cs.ucsd.edu/groups/hpcl/apples/nws.html.

[6] *DOCT Goal Security Architecture*, Technical Report E1-1, DOCT Project, SDSC, 10100 Hopkins Drive, La Jolla, CA 92093.

[7] Texcel, http://www.texcel.no.

[8] OpenText, http://www.opentext.com.

[9] Oracle, http://www.oracle.com

[10] DB2, http://www.software.ibm.com/is/sw-servers/database.

[11] Objectstore, http://www.objectstore.com

[12] The DB2/HPSS Integration project, http://www.sdsc.edu/MDAS.

[13] Kohonen Self Organizing Maps, Documentation for SOM_pak software tool, http://nucleus.hut.fi/nnrc/som_pak

[14] Learning Vector Quantization, Documentation for LVQ_pak software tool, http://nucleus.hut.fi/nnrc/lvq_pak

[15] Neural Network Research Center Home Page, Neural Network Research Center, http://nucleus.hut.fi/nnrc

[16] Informal Conversations with Michael Welge and Nina Mishra in 6/97, NCSA

[17] FreeWAIS, http://www.cnidr.org/welcom.html.

[18] Pretty Good Privacy, Inc. http://www.pgp.com/ and The International PGP http://www.pgpi.com/.

[19] "Secure Sockets Layer" http://www.netscape.com/newsref/ref/netscape-security.html.

[20] "The Relationship Between SSL And SHTTP", http://www.netscape.com/newsref/ref/netscape-security.html#rel.

[21] Secure Shell (SSH), http://www.cs.hut.fi/ssh/.

[22] "Kerberos: The Network Authentication Protocol", http://web.mit.edu/kerberos/www/.

[23] "Intrusion detection software," by Tom Perrine, *SDSC Gather/Scatter*, http://www.sdsc.edu/GatherScatter/GSspring96/perrine.html.

[24] "Applied Cryptography," by Bruce Schneier, 1997.

[25] "Question 165. What is X.509?," http://www.rsa.com/rsalabs/newfaq/q165.html.

[26] Computer Operations Audit and Security Technology (COAST), Purdue University, http://www.cs.purdue.edu/coast/coast-tools.html.

[27] The Pacific Institute of Computer Security (PICS), http://www.sdsc.edu/Security/.

[28] "Computer Oracle and Password System," ftp://ftp.cert.org/pub/tools/cops.

[29] Tiger, ftp://ftp.win.tue.nl/pub/security/tiger-2.2.3.tar.gz

[30] The lsof utility and ifstatus, ftp://vic.cc.purdue.edu/pub/tools/unix/lsof/.

[31] The ifstatus utility, ftp://coast.cs.purdue.edu/pub/tools/unix/ifstatus/ifstatus.tar.Z.