

TRUST AND SECURITY IN ENTERPRISE GRID COMPUTING ENVIRONMENT

Sanjay Goel
School of Business,
University at Albany, State University of New York
goel@albany.edu

Michael Sobolewski
Computer Science, Texas Tech University
sobol@cs.ttu.edu

Abstract

This paper presents a trust and security model for a Grid Computing System. The issues involved in development of this model are investigated in context of a service-based P2P architecture. The hierarchical intergrid architecture has been developed for B2B transactions across trusted intragrids. The paper discusses the issues posed by security implementation in these architectures. A centralized trust mechanism is proposed for the intragrids and a distributed trust mechanism for the intergrids.

Key Words

trust, security, grid computing, services

1. Introduction

P2P systems have become popular in context of sharing of large quantities of data [1], [2], [3]. These systems offer some inherent benefits, such as fault-tolerance, self-healing, self-organization, load balancing and scalability. As of today, more than a petabyte of data has been exchanged using these systems. Though beset with legal and ethical issues, the file sharing systems have demonstrated the feasibility and scalability of these systems. Despite the tremendous success of such distribute systems in the context of free exchange of media files it has been difficult to leverage it for mainstream business applications. The real potential of these systems lies in ubiquitous computing that involves coordinated resource sharing (e.g., CPU, bandwidth, analysis) and problem solving across distributed systems not only for solving computationally intensive applications but also for managing complex processes distributed on the network. In the past such distributed systems have investigated for use in high performance computing where jobs are scheduled across multiple processors on a structured grid for faster throughput of large computations [6]. These systems were typically based on the client-server architecture where a central server coordinates and schedules the jobs on different

processors. As the network bandwidth is increasing, distributed systems for managing complex processes on the Internet are beginning to emerge [7]. These systems based on P2P architectures operate outside of Domain Name System (DNS) and use standardized interfaces allowing them to work with diverse operating systems and dynamic network topologies. Significant research is required to impose controls on such systems in order to harness their power for useful commercial applications.

The research in distributed computing involving sharing of resources and collaborative computing falls under the preview of Grid Computing architectures [4]. Grid computing involves sharing of resources and coordinating processes across multiple nodes of the network. It often involves mobile code that migrates from one node to another and executes in the context of other nodes. Grid computing poses challenges in several areas, including scheduling and coordination of activities on the grid, security, trust and reliability. Though not critical for free file and music sharing utilities, security and trust become critical issues when these systems operate on the public Internet.

Unfortunately the infrastructure for establishing trust and security on the Internet remains fairly weak as is evidenced by the high incidence of fraud and other crimes on the Internet. The Internet architecture is plagued with a lack of security services in its design. When the Internet was conceived, provision for a weak authentication using passwords was provided. However, no provision was made to secure the data between nodes. The TCP/IP protocol stack is broadcast in nature, that is, the packets are released on the network allowing for potentially any node on the network to intercept the packets flowing on the network [10]. These packets can be scooped up from the network, and altered or reassembled causing loss of privacy and integrity of the data. Packets can also be spoofed to send unauthorized messages to unsuspecting machines.

The paper presents the architecture for a service-based grid computing system and for managing a complex set of

engineering processes. In this architecture, autonomous services available on the network federate together in order to process complex transactions. The services are characterized by a lack of identity and can be seamlessly replaced without affecting the quality of the transaction. In this architecture, multiple nodes provide the same service and nodes are selected competitively based on the trust (and reliability) of that node. This architecture poses several unique security requirements. These are addressed in context of two alternate scenarios: 1) B2B communication amount trusted corporate intragrids required for managing engineering analysis and 2) transactions among previously unknown peers on the public intergrids to model complex nested processes. The paper also presents the optimization formulation for competitive service selection during process execution on the network.

The paper is organized as follows. Section 2 describes the service-oriented problem that is being addressed by the proposed architecture. Section 3 describes the architecture in detail. Section 4 discusses the trust metrics and section 5 describes the security architecture for B2B communication. The summary and conclusions are presented in section 6.

2. Service-Oriented Problem

Engineering Design for aircraft engine turbines is a dynamic process with a high degree of complexity. It requires a flexible integration of analysis codes in different sequences to match the context of the specific engine requirements. The design process has typically been compartmentalized such that each domain is evaluated independently and interactions between domains are managed via boundary conditions in the problems. There has been a significant push to manage complex design sequences where multiple domains are simultaneously analyzed in order to improve the design performance as well as productivity. In the past, several integration tools have been developed to automate the design processes based on the existing client-server technology. Engineous, an engineering automation framework, [8] was one of the first attempts at creating an automated system that permits different engineering analysis codes to be linked sequentially to match the design process. Although Engineous was successful at combining the individual analysis codes in a predefined sequence, it lacked the flexibility required to allow the design process to change dynamically based on the changing requirements. The resulting process sequences have been fairly rigid, so instead of the process dynamically changing to suit the process, the problem is usually contorted to match the integrated automated process. FRODO [9] was another attempt at creating dynamic sequences of analysis codes. It uses a constraint propagation engine and represents all the dependencies between the analysis codes as constraints. Whenever a

variable value changes in any one domain, FRODO automatically updates all the values in every other domain. FRODO is limited in its flexibility at handling multiple analysis program sequences. It is too complex, too slow and restricted to use on a single network node. The current paper describes Federated Engineering Product Environment (FIPER) [11-[12] and its successor SORCER [16], [17] that were both developed as federated environments for engineering process integration that allows dynamic configuration of processes and service-oriented analysis on the network. The research investigates the security issues involved in the design and implementation of the SORCER environment.

3. Architecture of SORCER

The SOCCER framework being described in this work targets multiparty business transactions. A multiparty transaction is composed of a federation of providers that come together for completing a transaction. A transaction consists of a set of tasks with specific precedence relationships. The service providers do not have mutual associations prior to the transaction. They come together (federate) for a specific transaction. Each provider in the federation performs its services in a choreographed sequence. Once the transaction is complete, the federation dissolves and the providers disperse and seek other transactions to join. Different combinations of providers may come together for any given type of transaction at different times. The architecture consists of three components: 1) Context Model 2) Services 3) Grid Interactive Service Oriented Programming. These are described in the following sections.

3.1 Context Model of SORCER

A SORCER context model [12] is the basic element of SORCER data structures and is based on the percept calculus knowledge representation scheme [11]. It forms the essential structure of the data being processed. A context model in the SORCER system is represented as a tree-like structure of context nodes. A data node (leaf node) is where the actual data resides. The context denotes an application domain namespace, and a context model is its context with data nodes as leaf nodes appended to its context paths. A context path is a name for a data in its leaf node. The leaf node might contain any object and in particular an object that represents a file, for example a URL. A special container object called ServiceNode acts as a wrapper that holds a reference to a remote document object. A provider receives an input context model from a service requestor and returns output context model as a result of a provided service. A collection of attributes can be associated with each node to enable an efficient search for specific data by service providers.

3.2 Services

In SORCER any service is identified by the interface it implements and domain specific attributes used to locate a corresponding service provider. Besides service specific interfaces, all services also implement the common interface called *Service*. Hence all providers are peers, which can communicate with one another via the *Service* interface.

A service provider takes a context as argument of service request and selects data nodes on which it has to provide service. Then, the provider returns an output context that contains the result of provided service. The data, which a provider generates, can be in the form of a file, which is stored in a remote location. An object called *DocumentDescriptor* represents the remote references to files in SORCER. These document descriptors reside inside *ServiceNodes* in the context, which acts as a linkage for the remote document references. Any provider or user agent can remotely access these documents programmatically or through GUI in a seamless way using a file store service. Certain parts of the document can be filtered out with the help of *Filter* associated with its *ServiceNode* [16].

In the SORCER two types of basic exertions (network activities) are defined: tasks and jobs. A task is the atomic exertion that is defined by its context model (data), and by its method. An exertion method defines a service provider (grid object) to be bound to in runtime. This network object provides the business logic to be applied to the exertion context model. The computing framework based on concepts - context model, method and exertion - is called the CME framework for short.

A method is primarily defined by a provider type (interface) and selector (operation name) in the provider's interface. Optionally, additional attributes might be associated with the method, for example a provider's name, location or provider's identifier. The information included in the exertion method allows the SORCER program to bind the exertion to the network object and process the exertion's context by one of its peer's operations which is defined by its published interface.

The SORCER middleware consists of basic providers: *Jobber* (a service broker that coordinates execution of exertions within a job), *Cataloger* (a catalog of dynamically discovered SORCER services), *File Store* *Service* (grid-oriented files store service), and *Persister* (a data store service that stores and retrieves contexts, tasks and jobs) as depicted in the SORCER functional architecture in Figure 1. Domain specific providers used by SORCER programs (jobs) can be developed using SORCER development tools. All layers depend on the SORCER CME framework. Web clients and stand-alone requestors submit jobs to be executed by SORCER

middleware in concert with domain specific providers that complement SORCER grid interactive programming. In the case of web clients, exertions also can be submitted to a SORCER interportal or extraportal. The extraportal is used for HTTP-based B2B secure communication between intragrids over security firewalls.

Sun's Jini Network Connection Technology [18] is used to implement the functionality specified above. The discovery and lookup protocols that Jini defines allow for a dynamic federation of services to be created.

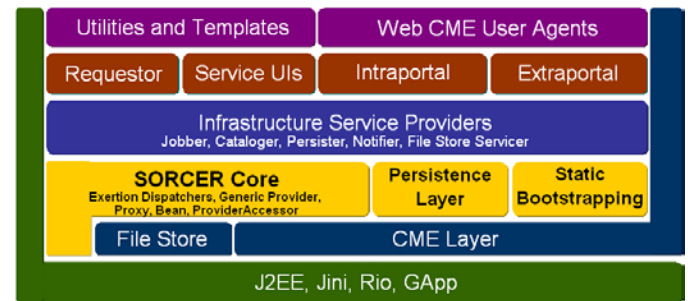


Figure 1. SORCER functional architecture

3.3 Grid Interactive Service-Oriented (GISO) Programming

Conventional programs are still based on writing hundreds of lines of codes. In SORCER, once providers are deployed, jobs (service-oriented programs) can be built using interactive tools. Thus we can create programs in which each task is like a line of code, which is actually a distributed activity in the network. The SORCER web based software development tools allow the user to create grid programs in terms of exertions interactively. The basic work unit or building block within the GISO programming [15] environment is an exertion. Each exertion (task or job) contains a remote method and a context model; both created interactively and persisted in a program store. The remote method specifies what action to be taken in a given step in the process. The context model contains all the data/information the remote method operates on or generates. A job defines the process. It consists of one or more exertions, the execution strategy for the process (sequential, parallel, looping and conditionals), and the mapping/relationship of data shared between exertions. It is worth noting that recursion of jobs is supported, that is any of the exertions within a job can be a job itself.

The relationship between the SORCER program objects and the general description of a nested transaction is as follows; a job represents the process, the method represents the remote action, and a context represents the data/information. The task acts as a container holding the method and context creating the basic unit of work that is passed between various service providers.

4. Trust

Trust is at the core of any system that is involved in bipartite or multi-party transactions. Entities involved in the transactions can be passionate or rational. Passionate entities are entities with free will (i.e. humans) and rational entities do not have the capacity for free will. Thus only passionate entities are able to ultimately control the outcome of the operations. Rational entities make decisions but based on the instructions from the passionate entities. This work considers trust in context of transactions between passionate entities. In P2P system peers are considered passionate entities since they often act as proxies for human beings that are passionate entities. Trust among passionate entities is as an expectation by one party that the other party will deliver on its promises. Such trust is usually established by the past performance history of an organization or by validation of a third party which is trusted. In brick and mortar companies, trust is established by the brand recognition that a company develops over a period of time. Similarly, in web commerce, the identity of the parties involved in a transaction is known via domain names and fixed IP-addresses, so trust is established based on domain name recognition. In the eBay model, for example, the trust exists for its reputation as a provider and trust exists among its participants that use its services for conducting transactions. Former type of trust is derived from the name recognition and the latter type of trust is computed by using a feedback of peer's performance from other peers.

In SORCER we propose, multiple parties interacting together use standard interfaces and do not have name recognition. This assumption is essential to allow a seamless integration of services via object types (defined by their interfaces) and to allow for one service to be transparently substituted with any other service implementing the same type. To establish trust in such an environment, an umbrella of external protection must be provided to the user. In SORCER, a concept of service rating or service quality is introduced to provide an external reference for the service. At the start of a transaction, the service requestor announces its requirements on the network, obtains a list of the relevant service providers (via a lookup service) and a rating for the services and then selects the service provider. The service selection is based on the rating, the cost, and the time required for completion of the service. Typically an optimization formulation with an objective function comprised of cost and quality and a constraint on the service completion time is used for the selection of services.

The Jobber selection algorithm automatically federates an optimum set of service for the transaction using the performance attributes of the service. Over time new services enter the network and existing services that are

not competitive may leave the network. The evolution of the network follows the Darwinian principle of natural selection and survival of the fittest: the weaker providers who fail to improve and adapt to the evolving environment are not selected and thus eventually eliminated from the network via attrition, while the stronger providers continue to innovate, improve, and grow. Such an environment leads to a robust and efficient system that manages itself.

Peer-to-peer systems have a pressing need for both, anonymity and trust which are non-orthogonal. Trust is established by the reputation of the entity, whereas anonymity is established by making the entity nameless and faceless. Because of the conflicting requirements development of a universal trust metrics suitable for all contexts seems infeasible as is evident from the existing schemes which are all fairly complex, have poor scalability and context dependent. The current work investigates two typical scenarios with vastly different security requirements for establishing trust in a service-based architecture, that is, a corporate intranet and the public intergrid. A centralized trust mechanism is proposed for the Intranet and a distributed trust mechanism for the Internet.

In a centralized trust mechanism rating is computed by a centralized authority that maintains performance statistics of services on the network and in a distributed fashion chains of trusted intermediaries are used to establish trust. Once the federation is formed, all peers in the federation trust each other completely. We describe in the next section the rating mechanisms and algorithm for service selection using an optimization formulation.

4.1. Centralized Rating Model

Normally trust is based on a provider's reputation established by brand name recognition and a perception of the users based on their experiences. In an automated system such as the one we propose, when a selection algorithm is picking the service providers, a quantifiable numerical measure of the vendor quality is required. The metrics measure attributes such as the service quality, reliability, and compliance with the provider's interface. Rating of a service can be computed based on data gathered by the rating agency on a number of metrics like length of service existence, number of complaints received and number of compliance defects. Equation 1 gives a simple formula that can be used to compute the quality rating.

$$Q_{ij} = \left(1 - \frac{\alpha_{ij}}{cm_j}\right) \times \left(\frac{t_{ij}}{tm_j}\right) \times \left(1 - \frac{\beta_{ij}}{\beta m_j}\right) \quad \dots(1)$$

Where, Q_{ij} is the rating for a Service S_j for a given provider P_i , α_{ij} is the number of complaints against a specific vendor P_i for a service S_j , cm_j is the average

number of complaints for service S_j , t_{ij} is the duration for which a service S_j from a Provider P_i has existed, and tm_j is the max duration of existence of a service S_j from any vendor, β_{ij} is the number of defects in compliance for Service S_j from provider P_i , βm_j is the maximum number of compliance defects possible for the given service S_j . Even though this scheme incorporates numerous elements of peer evaluation it is still vulnerable to eBay style fraud. In such frauds the peers build trust based on small transactions and commit fraud on large transactions. Thus the rating scheme has a value bound for the trust rating that is computed as a weighted mean of the transaction values. This is computed in equation (1a) below:

$$\tau_{ij} = \frac{\sum_k \gamma_{ijk} \times C_{ijk}}{\sum_k C_{ijk}} \quad \dots(1a)$$

Here τ_{ij} is the nominal value at which the trust is based. γ_{ijk} can take the value of 1, 0, -1 based on whether the feedback was positive, neutral or negative. C_{ijk} is the value of each of the transactions done by the vendor. If the value of the current transaction exceeds the threshold, then the rating value drops precipitously. The confidence of a transaction can be estimated statistically by using the standard deviations and the number of samples to compute the rating parameters.

4.2. Distributed Rating Model

One of the criticisms of centralized resources (e.g. trust) is that such resources are themselves vulnerable to attacks and reduce the scalability and reliability P2P systems. Despite the impact on robustness, some structure and control in P2P architectures is required to enable them to support a wide set of business applications.

The existing P2P file-sharing systems range from centralized to distributed, based on the location of registries and the search mechanism. In a centralized system such as Napster [1], a central registry stores all the links to media files and resources are located by a query on the central registry. However, in a distributed system such as Gnutella, each peer maintains a local registry containing a list of its own resources and a list of peers in its neighborhood. During the search, the query is broadcast to its neighbors who either fulfill the request or pass the request to their own neighbors until the request is satisfied or if the maximum hop-count is exceeded. Thus a chain of intermediaries is used to identify the peer with the requested resource.

Similar to the Gnutella approach, if trust is considered transitive, it can be computed in a decentralized fashion by creating chains of intermediaries through trusted neighbors. In this scenario each node has a set of trusted nodes based on its previous interactions. The trust with a node with no previous encounter is obtained by locating

trusted nodes that either have a direct trust relationship with the new node or have relationship with other nodes that have prior encounter with the node. Trust can be computed by aggregating the trust in the entire chain of intermediaries.

Let trust between nodes i and j be τ_{ij} such that $0 \leq \tau_{ij} \leq 1$. τ_{ij} is defined between node i & j if they have at least one interaction with each other. If a set of nodes $T_j = \{X_1, \dots, X_P\}$ exists such that $\tau_{X_k X_{k+1}}$ is defined for the entire set. Then trust between X_1 and X_P is given by equation 1c.

$$\tau_{(X_1 X_2) \dots X_P} = \left[\prod_{k=2}^{k=P} \tau_{X_{k-1} X_k} \right] \quad \dots(1c)$$

Let there be N such chains between nodes X_1 and X_T then the trust between them is computed by equation 1d.

$$\tau_{X_1 X_T} = \frac{\sum_{j=1}^{j=N} \tau_{(X_1 X_2) \dots X_P}}{N} \quad \dots(1d)$$

If no path exists to a node that is required a node can still be used if other nodes do not provide the same service.

The schemes defined above provide a generic template and can be adapted to problem context and scale of the application. Trust however, is a critical component of such systems and a part of selection of services.

4.3. Optimum Service Selection

For a transaction involving a single service the selection problem can be formulated as an optimization problem with the rating and completion time as constraints.

$$\begin{aligned} & \text{Minimize } C_{ik} \\ & \text{subject to } Q_{ik} > Q_k^m \\ & \text{and } T_{ik} < T_k^m \end{aligned} \quad \dots(2)$$

Where, C_{ik} is the cost of completing service S_k by a provider P_i , Q_{ik} is the quality rating of service S_k by provider P_i , Q_k^m is the minimum quality of service acceptable for Service S_k , T_{ik} is the service completion time for Service S_k provided by Provider P_i and T_k^m is the maximum completion time allowed for service S_k . This is a simple discrete optimization problem and can be solved easily using standard discrete optimization methods.

Let us now consider a more complex transaction involving multiple tasks with precedence relationships among the tasks. Such a nested transaction can be represented by a directed acyclic graph. In order to get the optimum set of services for the entire transaction, a composite rating score needs to be computed based on the ratings of individual services comprising that transaction. An average of all the ratings may not be the most

appropriate measure since different tasks will have different impacts on the quality of transaction. Specifically, the impact of quality of a given individual service on a transaction would depend on its contribution to such a transaction, its presence in the critical path, and dependence of other services in the transaction on it. The tasks on the critical path can be determined from the transaction graph. For a service graph for transaction T_j with p tasks the quality of transaction R_j can be computed as shown in equation 3.

$$R_j = \frac{\sum_{k=1}^p W_k * Q_{ik}}{\sum_{k=1}^p W_k} \dots (3)$$

where, W_k is the perceived importance of a task k in a transaction T_j and Q_{ik} is the rating of service S_k provided by Provider P_i . The optimization problem then becomes

$$\text{Minimize } \sum_{k=1}^p C_{ik}$$

such that $R_j > R_j^m$... (4)

$$\text{and } \sum_{k=1}^r T_{ik} < T_j^m$$

where, C_{ik} is the cost of the task S_k from provider P_i , R_j is the rating of the transaction, R_j^m is the minimum rating of the transaction allowed, and T_{ik} is the time to completion of task r (r belongs to the set of tasks on critical path), T_j^m is the maximum time allowed to complete the transaction.

In addition to trust between entities security needs to be established between the peers during transactions. The next section describes the security implementation for an Intergrid B2B application of the architecture.

5. Security: Intergrid B2B Scenario

In this section we discuss the use case scenario of B2B Fuel Nozzle and Combustor design across enterprise GE Global research Center (NY) and Parker Hannifin (OH). In the interactive design of combustor, one of the important aspects of the analysis is a fuel nozzle validation when the combustor geometry has been updated. Parker-Hannifin specializes in nozzle design and validation services and the GE GRC combustor service requires these services to be accessed for complete validation of the updated combustor geometry. The layout of the solution for this B2B interaction is shown in Fig. 2.

The GRC Perimeter Network or GRC DMZ consists of the standard GRC HTTP proxy server, screening security router(s) and the FIPER DMZ host used as a SSL FIPER Extraportal (FIPER interportal proxy server). Currently the FIPER DMZ host allows incoming connections from selected Parker hosts and only outgoing connection to

interportal host. Thus services can communicate with each other with the help of extraportals in both ends.

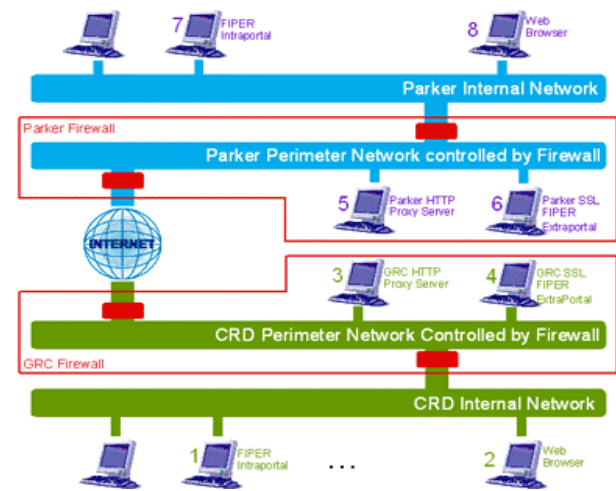


Figure 2. GE/Parker B2B conceptual architecture

Services in both ends can access data stored on the other side via document descriptors communicating with the File Store Service (FSS) sitting in the other end. FSS not only can be used to share information across services, they can also be used for smart filtering of data using the Filter Service provided by all FSSs. Thus we can avoid transfer of large chunks of data across firewalls. The ACL Manager takes care of guarding individual business objects (including documents), based on the Access Control rules set by the user or individual services (whoever is responsible for creating the document). All validation of requests is done by FSS based on credentials passed from the other end on whose behalf the current service is being executed.

The basic unit of each service provider remains the service proxy registered in lookup services. When requestor finds a proxy—based on the interfaces and specific attributes the provider implements—code for the proxy object not already available at the client downloads into the client following Jini mobile code semantics. Following that, the client invokes methods on the proxy. A client does not care or know how the proxy implements those methods: some proxy methods may involve network communication with a remote service implementation, whereas others may execute locally to the proxy.

The SORCER interportal security model based on Jini 2.0 adds three steps to the conventional programming model [19]. First, it allows a requestor to decide whether to trust a proxy object or not. If a service proxy is supposed to represent a remote service provider, how do you know the proxy you just downloaded is in fact the proper proxy? The Jini security framework provides a solution with the bootstrap proxy and the verifier. The bootstrap proxy

resolves all needed classes locally, so they are trusted classes, thus avoiding downloaded classes.

Next, once you place some trust in a downloaded proxy, you must ensure the proxy offers some guarantees in performing its work. If a requestor wants to invoke a method on that proxy, you may want to perform that action only if the proxy can guarantee the integrity and confidentiality of the information traversing the network. The Jini security model lets you place such constraints on a proxy.

Finally, you want to grant privileges to a proxy based on the trust you placed in that proxy. If a service requestor requires a network connection to a Parker's server while executing a job by GE GRC jobber, you must grant that connection permission to the proxy. On the other hand, if that proxy wishes to reformat your hard disk, you want to deny it that permission. The Jini security model's final aspect is its ability to dynamically grant permissions to a downloaded service proxy.

Rather than making programming changes to a proxy, the security-related information is injected into that proxy at the time its service provider makes its service available in the SORCER environment. Making a service available for remote invocation is broadly referred to as exporting that service. The SORCER security framework defines an exporting mechanism that supports security-conscious exporting of a SORCER service using Jini Extensible Remote Invocation configuration-based approach.

6. Conclusions

The paper presents architecture for an intergrid computing system that would allow complex federated transactions to be modeled on the intragrid. It presents two schemes for estimating trust among peers using a centralized and a distributed scheme. A security model is also presented for B2B transactions across different intragrids that is based on Jini and Java security frameworks.

References

- [1] Napster, (2003) homepage <http://www.napster.com>.
- [2] Oram, A. (2003), Gnutella and Freenet Represent True Technological Innovation. The O'Reilly Network. See <http://www.oreillynet.com/lpt/a/208>. Last accessed on March 21, 2003.
- [3] Kazaa website <http://www.kazaa.com>
- [4] Foster, I., Kesselman, C., Tuecke, S., "The Anatomy of the Grid: Enabling Scalable Virtual Organizations" *International J. Supercomputer Applications*, 15(3), 2001.
- [5] Lv, Q., Cao, P., Cohen, E., Li, K., and Shenker, S., "Search and replication in unstructured peer-to-peer networks," in *Proceedings of the 16th annual ACM International Conference on Supercomputing (ICS)*, 2002.
- [6] Feitelson, D. G., and Rudolph, L., eds. *Job Scheduling Strategies for Parallel Processing, Lecture Notes in Computer Science*. Springer-Verlag, 1995-1998.
- [7] DeFanti, T., Foster, I., Papka, M., Stevens, R. and Kuhfuss, T., "Overview of the I-Way: Wide area visual supercomputing", *International Journal of Supercomputing Applications and High Performance Computing*, 10(2):123-131, 1996.
- [8] Nicklaus, D., Tong, S., and Russo, C., (1987) "ENGINEOUS, A Knowledge Directed Computer Aided Design Shell", *The Third IEEE Conference on Artificial Intelligence Applications*, Feb. 22-28, 1987.
- [9] Kolb, M. A. and Bailey, M. W., (1993), "FRODO: constraint-based object modeling for preliminary design," *Proceedings of the 19th Annual ASME Design Automation Conference*, Albuquerque, NM, Vol. 65, pp. 307-318.
- [10] Bhimani, A., "Securing the Commercial Internet", *Communications of the ACM*, Vol. 39, Issue 6, pp. 29-35, 1996.
- [11] Sobolewski, M., *Knowledge-Based System Integration in a Concurrent Engineering Environment*. J. Komorowski and Z.W. Ras (Eds.) *Methodologies for Intelligent Systems, Lecture Notes in AI*, No 689, Berlin: Springer-Verlag, pp. 601-611, 1993.
- [12] Zhao, Shuo, and Michael Sobolewski, 2001, *Context Model Sharing in the FIPER Environment*, *Proc. of the 8th Int. Conference on Concurrent Engineering: Research and Applications*, Anaheim, CA.
- [13] Sobolewski, M., *Federated P2P Services in CE Environments*, *Advances in Concurrent Engineering*, A.A. Balkema Publishers, 2002, ISBN 90 5809 502 9, pp. 13-22, keynote paper (2002).
- [14] Sobolewski, M., 2002. *FIPER: The Federated S2S Environment*, *JavaOne, Sun's 2002 Worldwide Java Developer Conference*, San Francisco (2002), (<http://servlet.java.sun.com/javaone/sf2002/conf/sessions/display-2420.en.jsp>)
- [15] *Grid Interactive Service Oriented Programming*, Michael Sobolewski, Texas Tech University, Raymond Kolonay, Air Force Research Laboratory, WPAFB, SORCER Lab Technical Report, 2003.
- [16] *Service-Oriented File Sharing*, Michael Sobolewski, Sekhar Soorianarayanan, Ravi-Kiran Malladi-Venkata, *Proceedings of Intl. Conference on Information and Knowledge Sharing*, 2003.
- [17] SORCER Web site, <http://www.sorcer.cs.ttu.edu>
- [18] Edwards, W.K. (2000). *Core Jini*, 2nd ed., Prentice Hall, ISBN: 0-13-089408.
- [19] Frank Sommers, *Jini Starter Kit 2.0 tightens Jini's security framework*, *JavaWorld 2003*, http://www.javaworld.com/javaworld/jw-05-2003/jw-0509-jiniology_p.html.