A Flexible Communication Scheme to Support Grid Service Emergence

Lei Gao and Yongsheng Ding

College of Information Sciences and Technology,
Donghua University, Shanghai 200051, P. R. China
ysding@dhu.edu.cn

Abstract. The next generation grid systems exhibit a strong sense of automation. To address such a challenge, Our previous work has viewed a grid as a number of interacting agents and applied some key mechanisms of natural ecosystems to build a novel grid middleware system, where a collection of distributed agents are searched, assembled, organized and coordinated to emerge desirable grid services. All these actions of agents depend on an effective communication scheme.

In this paper, we design a flexible communication scheme to implement the complicated coordination strategies among agents, including a RMI-IIOP-based transport mechanism, an ecological network communication language, and an ecological network interaction protocol from low to high implementation strategy. To test our hypothesis that grid services with desired properties can emerge from individual agents via our communication scheme, simulations of resource discovery service are carried out. The results prove that the scheme can well support this kind of bottom-up approach to build desirable services in grid environments.

1 Introduction

Grid systems have evolved over nearly a decade to enable large-scale flexible resource sharing among dynamic virtual organizations (VOs) [1]. Next generation grid systems are heading for globally collaborative, service-oriented and live information systems that exhibit a strong sense of automation [2], such as autonomic configuration and management. These "automatic" features, resembling the self-organizing and the self-healing properties in natural ecosystems, give us inspiration for designing next generation grid systems by associating them with some key concepts and principles in ecosystems.

Our previous work have viewed a grid as a number of interacting agents and applied some key mechanisms of natural ecosystems to build a grid middleware system named Ecological Network-based Grid Middleware (ENGM) [3]. Grid services and applications in ENGM can emerge from a collection of distributed and autonomous agents as the creatures living in a large ecosystem. Searching, assembling, organizing and coordinating of agents to emerge a desirable grid service depend on effective communication and cooperation scheme. However, we

O. Gervasi et al. (Eds.): ICCSA 2005, LNCS 3482, pp. 69–78, 2005.

focus the discussion solely on issues of the ENGM architecture and its dynamic service design, and do not refer the communication scheme of ENGM in [3].

The implementation of communication scheme in most of current grid middleware systems relies on traditional communication standards, such as Message Passing Interface (MPI) and Parallel Virtual Machine (PVM). For example, MPICH-G2 [4] is a typical grid-enabled MPI implementation built on top of services provided by the Globus Toolkit. Agent technologies have recently become popular in the grid community, thus the research on grid computing with agent communication is becoming a hotspot. The start-point/endpoint paradigm, a key concept of the Nexus communication, has been successfully used to the world of agents, which resulted in the Southampton Framework for Agent Research [5]. The Control of Agent Based System Grid [6] is based on Java Remote Method Invocation for inter-agent communications, to integrate heterogeneous agent-based systems, mobile agent systems, object-based applications, and legacy systems.

In this paper, we describe how to combine agent communication with grid computing for emerging desired services in ENGM, and explain the rationale behind our design. We expect, via our communication scheme, emergent grid services have not only corresponding common functionalities, but also some life-like properties such as survivability and adaptability. Agent communication is generally broken down into three sub-sections [7]: interaction protocol, communication language, and transport protocol. Interaction protocol is the high-level strategy pursued by the agents that govern their interactions with other agents. The communication language is the medium through which the attitudes regarding the exchange content are communicated. The transport protocol is the actual transport mechanism using the communication languages.

Hence, next section first overviews ENGM architecture. Based on it, our communication scheme is presented including agent inner-communication module, Ecological Network Communication Language (ENCL), transport mechanism based on Remote Method Invocation over Internet Inter-Orb Protocol (RMI-IIOP [8]), and Ecological Network Interaction Protocol (ENIP) in Section 3. Considering interaction protocol is service dependent, we take the example of resource discovery service to depict the design of ENIP. To test our hypothesis about desired grid services can emerge from agents via the schema, a simulation on resource discovery is carried out in Section 4. Section 5 concludes our efforts.

2 The Architecture of Ecological Network-Based Grid Middleware

From the implementation point of view, from the bottom up, the architecture of ENGM system is presented as three-layers: Heterogeneous and Distributed Resources layer, ENGM layer, and Grid Applications for VOs layer.

(1) *Heterogeneous and Distributed Resources* consist of different types of resources available from multiple service providers distributed in grids. Via a java virtual machine, an ENGM platform can run on a heterogeneous distributed sys-

tem that established in a network node. (2) *ENGM* provides the services support a common set of applications in distributed network environments. It is made up by ENGM functional modules, ENGM core services, grid agent survivable environment, and emergent grid common service. (a) In ENGM functional modules layer, low-level functions related to management of networks and systems are dealt with. (b) ENGM core services layer provides a set of general-purpose runtime services that are frequently used by agents such as naming service and niche sensing service. (c) Grid agent survivable environment is runtime environment for deploying and executing agents that are characterized by high demand for computing, storage and network bandwidth requirements. (d) Emergent grid common services part is kernel of middleware and responsible for resource allocation, authentication, information service, task assignment, and so on. These common services are emerged from those autonomous agents. (3) *Grid Applications for VOs* use developing kits and organize certain agents and common services automatically for special purpose applications.

ENGM is fully implemented by Java language. Technically, an agent in it is a combination of a Java object, an execution thread, a remote interface for network communication, and a self-description. These four simple pieces together create an agent, a full-fledged autonomous process in ENGM.

3 Agent Communication Scheme

3.1 Agent Inner-Communication Module

Each agent follows a set of simple behavior rules (such as migration, replication and death) and implements a functional component related to its service. An application is created out of the interactions of multiple agents. A collection of agents can form a community as to a certain law. A community is able to emerge a higher-level service. These formed communities can be viewed as emerging agents that organize a higher-level community to provide complex services.

These agents fall into two categories: grid user agents (GUAs) and grid service agents (GSAs). A GUA represents a kind of user tasks. While GUAs are used to comprise the main components of ENGM, such as Grid Information Ser-

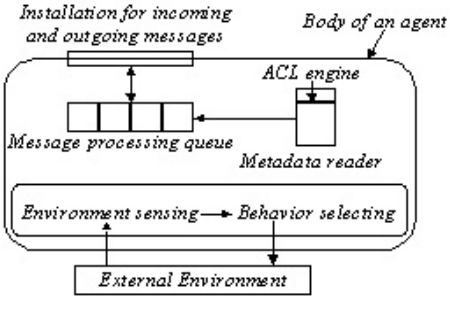

Fig. 1. Structure of agent inner-communication module

vice Agent (GISA). An agent is represented on its unit function that is defined as a metadata structure. Fig. 1 demonstrates the structure of a generic agent inner-communication module. The *metadata reader* is used for an agent to obtain another agent's metadata. The metadata of an agent consists of *agentID* (a global unique identifier of an agent), *agentAddress* (location of this agent), *serviceType* (service type of this agent), *serviceDescription* (description of service information) and *relationshipDescription* (information about the its acquaintances, agents know each other are called *acquaintances*). When an agent receives a message via the *installation for incoming and outgoing messages* from other platform or other agent, it inserts the message in its *message processing queue*. Each agent uses an independent thread to continuously sense the nearby environment, invoke most suitable behavior for the current environment condition.

3.2 Ecological Network Communication Language

An agent's communications are structured and constrained according to predefined speech act-based messages, usually referred to as *performatives*, which together make up an agent communication language (ACL). We reuse FIPA ACL [9] to design a high-level language called Ecological Network Communication Language (ENCL), as oppose to traditional specified interface definitions to achieve flexible interactions.To communicate using this language, each agent has to have an interpretation engine, as shown in Fig. 1.

An ENCL message contains a set of one or more parameters that can be arranged randomly in the message. Precisely needed parameters for an effective communication will vary according to the different situations. These message parameters are selected from those in FIPA ACL, including *performative-name*, *sender*, *receiver*, *content*, *language*, *ontology*, *conversation-id*, *reply-with*, *in-reply-to*, and *interaction-protocol*. We reuse part of FIPA ACL performatives and specify some new performatives. Performatives available in the ENCL are listed in the following: *request*, *agree*, *refuse*, *not-understood*, *inform*, *failure*, *defray*, *query-ref*, *query-if*, *advertise*, and *convene*. Among them, *advertise*, *convene*, and *defray* are specific to ENCL. An agent may send an advertise message to notify nearby agents of its existence (publishing its information and its capabilities such as the service it provides). An agent may send convene message to ask for interaction partner(s) to complete a certain task. Whenever receiving a service, the GUA returns a defray with a credit (a reward or a penalty) to an agent for its service. Also, we use XML (eXtensive Markable Language) as primary coding language of ENCL because of its strong capability of data description and metadata aspect. By combining the advantages of the ACL and XML, ENCL can flexibly implement the complicated coordination strategies.

3.3 Transport Mechanism

ENGM message transport is required to offer a reliable, orderly and accurate messaging stream, to detect and report errors to the sender, and to facilitate agent mobility. ENGM uses RMI-IIOP as transport protocol for ENCL messages. RMI-IIOP, the Sun Microsystems-IBM vision for distributed communi-

cation, provides the robustness of CORBA (Common Object Request Broker Architecture) and the simplicity of Java RMI. It is an application-level protocol on top of TCP. What's more, RMI-IIOP makes it possible that serialized Java objects can transmit among different components. Adopting this technology is also based on the following considerations: 1) It takes Java technology as the core, which offers simple and effective way for Java-based agent distributed computing. 2) It can find, activate and collect the useless agents. It allows the migration of data and codes, which facilitates the software implementation of autonomous and mobile services. 3) Using it, ENGM can flexibly implement agent-to-agent, community-to-community and agent-to-community communications.

Although a community is formed by the interactions of multiple assembled agents, it stresses on the global pattern. That is, it can act as a whole to communicate with other agents or communities. For example, when interacting, an agent needs to send a message to a community that provides a specific service. To address such a challenge, we introduce the concept of "channel" that represents a specific queue. If two communities use different channels, they would not affect each other. Sending or receiving messages in a certain channel is similar to tuning to a certain TV channel or radio frequency.

3.4 Ecological Network Interaction Protocol

To ensure interoperability among agents in wide-area grid environments, a sequence of interactions between two agents (a community can also be viewed as an emerging agent) is defined by an Ecological Network Interaction Protocol (ENIP). ENIP provide a reusable solution that can be applied to various kinds of message sequencing we encounter between agents. Only the agents that have implemented the same an ENIP are allowed to interact. Besides some conventional ENIPs defined by the ENGM platform, service developers may also define an ENIP relevant to their applications.

In the paper [10], we proposed a social network-inspired resource discovery approach, including some key models such as resource matching model, request forwarding model, and service defraying model. In this section, we further develop the approach as a resource discovery service of the ENGM platform and take the service as an application case to introduce ENIPs that are depicted as the Agent Unified Modeling Language (AUML) [11].

The next generation grid systems will exist in an unstable environment where a large number of nodes join and leave the grid frequently. Resource discovery service is a critical activity in such an environment: given a description of resources, it will return a set of contact addresses of resources that match the description. We consider a collection of agents that form a relationship network to discover desired resources. The presented approach is proposed as three-phases: GISA relationship construction, request processing strategy and trust-based reconstruction of relationship. GISA relationship construction is responsible for collecting and updating information on the currently participating peers and forming the relationship network; request processing strategy performs the search itself;

trust-based reconstruction of relationship makes the necessary preparations for a more efficient search. We describe the latter two phases using AUML.

Fig. 2 depicts a protocol expressed as an AUML sequence diagram for the ENGM request forwarding protocol. When invoked, a GISA acting as a request sender sends a request message with a given TTL (Time to live) to an acquaintance agent that has biggest probability to respond the request. The request receiver can then choose to respond to the sender by: refusing to process the request, responding the request and then forwarding the request (or only forwarding the request, which depends on the conditions), notifying the attempt to process is failed, or indicating that it did not understand. Depending on the conditions it contains, the symbol of decision diamond indicates it can result in zero or more communications. To implement an ENIP, the protocol specification

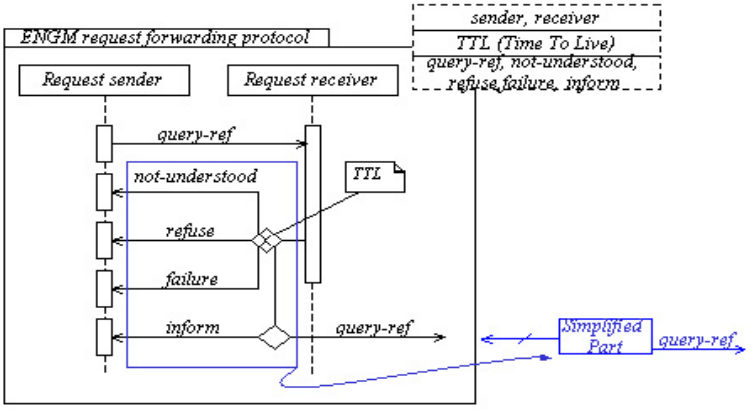

Fig. 2. A generic ENGM request forwarding process expressed as a template package

requires demonstrating the detailed processing that takes place within an agent. The AUML statechart can express the process that higher-level agents consist of aggregations of lower-level agents in ENGM interactions. In addition, it can also specify the internal processing of a single agent. Fig. 3 depicts the detailed request-respond process of proposed discovery approach using a combination of diagrams. On the bottom of it, the statechart that specifies request processing that takes place within a request originator. Here, the sequence diagram indicates that the agent's process is triggered by a query-ref message and ends with a defray message which can update and strengthen the relationship network to perform discovery better. To show explicitly, the sequence diagram has been simplified. We use a box with "Simplified Part" to replace some communicate operations, as shown in Fig. 2. A GISA relies in some GISAs and mistrusts in others to achieve its purpose. The reliability is expressed through a *trustCredit* value with which each GISA labels its acquaintances. *trustCredit* stands for how a relationship partner performed in discoveries in the past. In addition, information on similar requests user evaluated previously in *collaborationRecord*

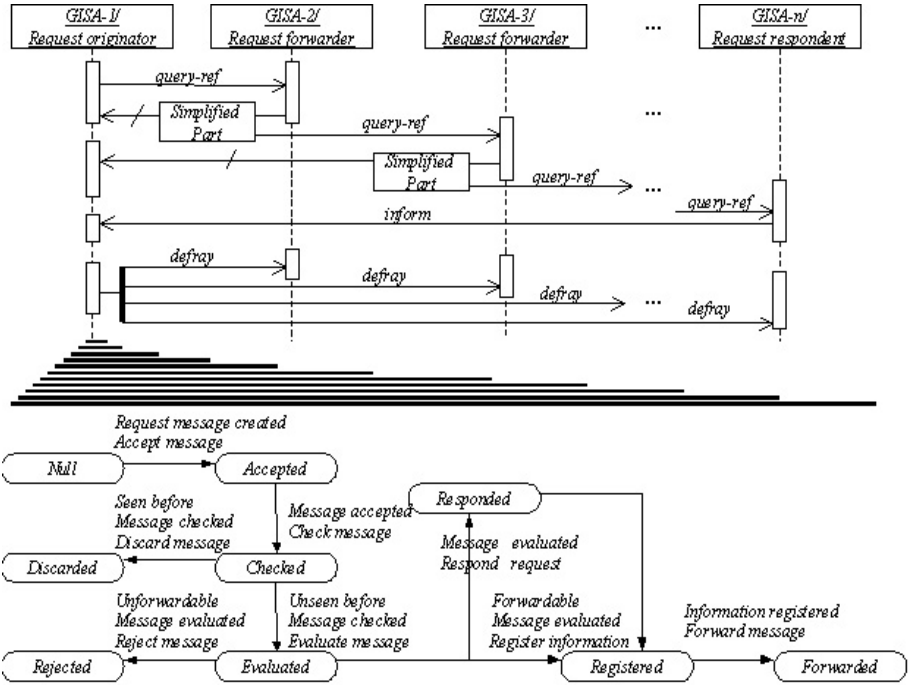

Fig. 3. The interaction protocols of discovery request-respond process using a combination of diagrams, on the bottom of which is the statechart that specifies request-processing behavior for a GISA (request originator)

attribute may help users get desired resources. GISA can use *collaborationRecord* to distinguish which of the two GISAs are more likely to be useful for satisfying user demand. Thus, a decision can be easily made by a GISA regarding which acquaintance is more likely to lead towards desired grid resources. If the condition is satisfied according to *request forwarding model*, the GISA will forward the request message. The GISA will discard the message if it neither responses the request (according to *resource matching model*) nor forwards the request.

As shown in the sequence diagram of Fig. 3, on receiving a request hit, the request originator returns a defray message including a collaboration record and a credit that stands for user evaluation of request hit, according to *service defraying model*. If the relationship of a GISA does not contain the corresponding collaboration record, the collaboration record including initial user evaluation is added to *relationshipDescription* attribute. A credit could be a reward or a penalty, which indicates the degree of its preference of the received request hit. This message is propagated through the same path where the discovery request was originally forwarded. When an intermediate GISA on the path receives message, it adjusts the *trustCredit* value of the relationship that was used to forward the original request. The *trustCredit* value is increased for a reward (i.e., high degree of the request originator's preference), and is decreased for a penalty (i.e.,

low degree of the request originator's preference). Some relevant designs and models such as *resource matching model*, *request forwarding model* and *service defraying model* are described in our paper [10]. Due to focusing on communication schema here, we will not carry on discussing the detailed contents.

4 Simulation Experiment

To examine: (a) whether the design of ENIP is expressive enough to describe complex grid services and (b) whether desired services can emerge from our service-building approach and communication solution, we develop a simulator of discovery service on ENGM platform to check above two aspects.In the simulator, minimum capabilities of ENGM such as *agent communication service*, *agent evolution state management* and *community niche sensing* are implemented.

4.1 Simulator Implementation

(1) Initial Network Topology and Simulation Time We adopt the generation method of network topology proposed in [12]. Suppose that each time there is only one request message sent by a GISA and the same message can be sent only once by the request originator during the entire simulation. Define that a cycle starts from a request message sent by a GISA and ends till all the relevant messages disappear in the system, and 100 cycles is a generation.

(2) Resource distribution. We make a set of common resources (contains 20000 different resource vectors) and a set of new-type resources (contains 2000 different resource vectors that are completely different from common resources). We experiment on two strategies. (a) Balanced distribution strategy: initially each GISA provides 3-5 resource vectors, which are randomly picked out from the common resource set. (b) Unbalanced distribution strategy: a few number of GISAs provide most of the resource vectors from common resource set, while the large number of GISAs share the small part of the resources.

(3) User requests and user evaluation. Requests are initiated at a fixed percentage of randomly selected GISAs and contain resource vectors. The resource attributes of each request have the same weight. Two user request patterns are studied: (a) unbiased user request scenario (requesting all the vectors randomly) and (b) biased user request scenario (using a great probability to request a small part of and special vectors available in the simulation network).

4.2 Simulation Results

We study the effect of different resource distributions and user request patterns on discovery service. A parameter η is introduced to act as the weight given to *trustCredit* and *collaborationRecord*. As shown in Fig. 4, the random forwarding has the lowest efficiency, though it is low-cost (no need to store any discovery information in GISAs). For four experimental environments and different η values, in general, our discovery service can improve its performance adaptively. At the beginning of simulation, relationships of agents are random, and discovery

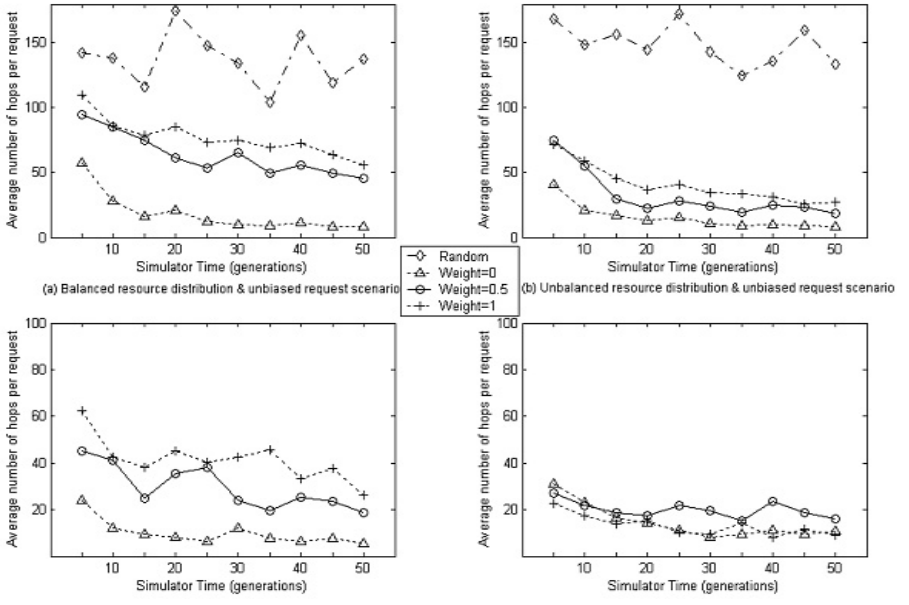

Fig. 4. Average number of hops per request as a function of simulation time in four environments. (a): balanced resource distribution and unbiased user request scenario; (b): unbalanced and unbiased; (c): unbalanced and biased; (d): unbalanced and biased. The number of GISA is 5000.The TTL values are set to 200

performs poorly. Many hops need to be visited to hit the target GISA (hops is a measurement of node amount). As more simulation cycles elapse, GISA gradually obtains many relationships similar to them, leading to improved performance in discovery process. We find that such improvement results from the clustering of request-matched GISAs. With enough hops, GISAs are likely to meet some relative resource clusters during the discovery process and enter the clusters to find the required resources. The clusters have not formed in the process of random forwarding simulation, and the discovery will go aimlessly till it meets the matched GISA. It can be seen that the discovery service can form clusters and improve the discovery performance.

The result has proved that, through exploiting the ENGM platform and its communication implementation, the above process can be well-carried out. The design of ENIP is expressive enough to describe complex discovery service and support a group of agents to emerge discovery service that can adapt to dynamically changing environments.

5 Conclusions

We design and implement a communication schema that couples grid techniques to enable flexibly searching, assembling, organizing and coordinating of agents for

emerging desired grid services. This schema is built on the ENGM platform that would not only be more resistant to failure, but also would provide many useful functions and services for the agents to invoke. To evaluate the communication design and its supported ability of service emergence, we take the service of resource discovery in grid environments as an application case and conduct a series of simulation experiments. The experimental results demonstrate via our communication solution, a service with desired properties can emerge.

Acknowledgments. This work was supported in part by the National Nature Science Foundation of China (No. 60474037 and 60004006) and Specialized Research Fund for the Doctoral Program of Higher Education from Educational Committee of China (No. 20030255009).

References

1. Foster, I., Kesselman, C., and Tuecke, S.: The anatomy of the grid: enabling scalable virtual organizations. Int. J. Supercomputers Applications.**15** (2001) 205–220
2. De Roure, D., Jennings, N. R., and Shadbolt, N. R.: The Evolution of the Grid. In Berman, F., Fox, G. and Hey, A. J. G.(eds.): Grid Computing: Making the Global Infrastructure a Reality , NJ: John Wiley and Sons Ltd. Publishing (2003) 65–100
3. Gao, L., Ding, Y.-S., and Ren, L.-H.: A Novel Ecological Network-based Computation Platform as Grid Middleware System. Int. J. Intell. Sys. **19** (2004) 859–884
4. Karonis, N. T., Toonen, B., and Foster, I., MPICH-G2: A Grid-enabled Implementation of the Message Passing Interface, J. Para. and Dist. Comp.**63** (2003) 551–563
5. Moreau, L., Zaini, N. M., Cruickshank, D., and De Roure, D.: SoFAR: An Agent Framework for Distributed Information Management. In Plekhanova, V. (eds.): Intelligent Agent Software Engineering, PA: Idea Group Publishing, (2003) 49–67
6. Kahn, M. L. and Cicalese, C. D. T.: CoABS Grid Scalability Experiments. Autonomous Agents and Multi-Agent Systems. **7** (2003) 171–178
7. Finin, T., Labrou, Y., and Mayfield, J.: KQML as An Agent Communication Language. in Bradshaw, J.(Eds.): Software Agents,AAAI/MIT Press (1997)
8. Java RMI-IIOP Documentation. http://java.sun.com/j2se/1.3/docs/guide/rmi-iiop
9. Labrou, Y., Finin, T., and Peng, Y.: Agent Communication Languages: The Current Landscape. IEEE Intelligent Systems. **14** (1999) 45–52
10. Parunak, H. V. D. and Odell, J.: Representing Social Structures in UML. in Wooldridge, M., Ciancarini, P., and Weiss, G.(eds.): Agent-Oriented Software Engineering Workshop II. Springer publishing, Berlin (2002) 1–16
11. Gao, L., Ding, Y.-S., and Ren, L.-H.: A social Network-Inspired Resource Discovery Approach in Grid Environments. IEEE Trans. Sys. Man and Cybe. submitted
12. Barabsi, A.-L. and Albert, R.: Emergence of scaling in random networks. Science. **286** (1999) 509–512