A Service-Oriented Approach for Building Autonomic Peer-to-Peer Robot Systems

Michele Amoretti, Francesco Zanichelli Gianni Conte
Department of Information Engineering
University of Parma, Italy
{amoretti, zanichelli, conte}@ce.unipr.it

Abstract

The forthcoming mass introduction of cooperating robots in everyday environments calls for major advances in the development of flexible, cost-effective, modular, dependable, and robust distributed robotic systems. In this paper, we introduce a conceptual framework and a middleware implementation to support service-oriented distributed robotic applications. Our goal is to provide networked robots with autonomic features, to improve their independence from human operators and survivability, without the need for a centralized IT infrastructure. We propose a novel fusion of Web Service, peer-to-peer, and robot control technologies, with reference to real scenarios involving mobile robots. One major feature of our approach is related to service mobility among peer robots, that is on-the-fly acquisition of knowledge and skills, yielding to improved system performance and robustness.

1 Introduction

A general problem of distributed robotic systems is that their increasing complexity is a limiting factor for further development. Referring to space missions, Rouff *et al.* [17] propose to combine autonomy with *autonomicity*, to support the survivability of remote systems (*e.g.* spacecrafts, robot swarms) in harsh environments, expecially when human control is unfeasible. In our view, the four key properties of autonomic systems [12] are of paramount importance for distributed robotic systems, in order to improve pervasiveness and hide the complexity of their internal processes, in particular those managing the resources for the exposed service applications. We illustrate the four key features in the context of realistic robot scenarios.

- *Self-Configuration* Consider a swarm of networked wheeled robots performing cooperative tasks, such as searching, retrieval, hunt/defend. Based on sensorial data, providing local proximity information and simple description of the environment, system components dynamically configure their behaviour by means of leader election algorithms or consensus agreement algorithms which allow to cooperate on tasks.
- *Self-Healing* When a sensor/actuator is damaged, or a control software module is corrupted, or a member of a mobile robot squad is lost, the system reacts by activating recovery mechanisms, which may be based on redundancy, software mobility, and cooperation.
- *Self-Optimization* Environmental data are collected and spread into the network, along with local internal monitoring information, in order to improve system performance with respect to the defined requirements, by means of sensor recalibration, load balacing, optimal service allocation, and so on.
- *Self-Protection* In wireless ad hoc robot networks, networking services are provided by the nodes themselves. Since selfish behaviour can significantly damage network performance, reputation or virtual currency systems have been proposed to stimulate cooperation among nodes in ad hoc networks. Moreover, all distributed robot systems must ensure proactive identification and protection from arbitrary attacks.

Our approach for achieving these goals combines two increasingly popular IT technologies, namely service-oriented and peer-to-peer computing. In a peer-to-peer based service-oriented architecture (SOA) there is no always-on end system hosting a server at the center of the application. Instead, arbitrary pair of peer application entities communicate directly with each others. One of the greatest strengths of the peer-to-peer architecture is its scalability, since data and control are distributed and requests are load-balanced across the network. On the other hand, because of the highly distributed and decentralized nature of peer-to-peer applications, they can be difficult to manage.

The conceptual framework we illustrate in this paper proposes robot networks whose peer participants do not need centralized IT infrastructures to provide and consume services. Besides publishing information about their own services, and locating useful services provided by other participants, peers are also able to deploy new instances of discovered services, having obtained the equired software packages from other nodes. This appears similar to the well-known file sharing, although the exchanged information are not only static data, but also services which respond to consumers' messages, once deployed to the new host(s). Mobile services are different from mobile agents [6], since they are passive entities without autonomy, social ability, or learning capabilities.

The paper is organized as follows. In section 2 we describe our framework, with emphasis on the internal modules which constitute each peer entity of the system, namely the adaptive message routing and service discovery modules. In section 3 we illlustrate the technologies we have used to implement the framework in the SP2A middleware. Section 4 describes two robotic application scenarios we are investigating to test our architecture. An overview of significant related works is provided in section 5. Finally, conclusions and future work are illustrated in the ending section of the paper.

2 Conceptual Framework

The architecture we propose is based on the peer-to-peer paradigm, being each networked robot both a provider and consumer of services. The main advantages of this approach are the improved scalability and robustness of the system, compared to traditional client/server architectures, which are characterized by communication bottlenecks and single points of failure. Peer robots share the same logical structure (illustrated in figure 1), but obviously they provide specialized services, depending on their resources (*e.g.* flying robots and humanoid robots have different actuation systems and capabilities).

Some of the following modules have been already formalized as design patterns [11]. The **Message Endpoint (ME)** is responsible for connection establishment, *i.e.* **Message Channel (MC)** creation, and messaging among peers. **Resource Provision Services (RPSs)**, are the services which characterize each peer. RPSs in general are stateful, since they provide access to local resources, whose state can change (*e.g.* the sensors of a mobile robot, but also its position in the environment it is exploring). The **Resource Monitor (RM)** analyzes the state of local resources, providing reports on-demand. When the system is no more under human control, this module becomes the key for system's autonomicity. RPS execution is managed by the **Scheduler (SC)**, depending on the information provided by the Resource Monitor. Moreover, RPSs can move from a peer to another peer, thanks to the **Service Loader (SL)** module, which is responsible for the acquisition of new RPSs and their deployment at runtime. The simplest case is weak mobility, *i.e.* only the service code is asynchronously transferred between nodes, not the service state. Security issues related to this approach have been analyzed in several papers, among which we remind [7]. In a peer-to-peer scenario, the most interesting solutions for shared resource reliability refer to the concept of reputation [14], and on authentication/authorization mechanisms based on certificates provided by distributed authorities [15]. The **Security Manager (SEC)** is responsible of protecting shared resources (exposed by the services), and to check the reliability of the acquired service code. In a more complex scenario, destination nodes not only must be provided with the adequate resources, but also must be able to replicate the resource configuration (state) of the sender peer. The **User Interface (UI)** is the only module which is exposed to human operators, whose commands are translated in messages to be processed by internal modules. The UI also provides output messages to human operators. Distributed control messages for service publication/discovery and connectivity are propagated by the **Message Router (MR)**, which chooses the correct Message Channel, according to the strategy which best fits the estimated overlay network topology.

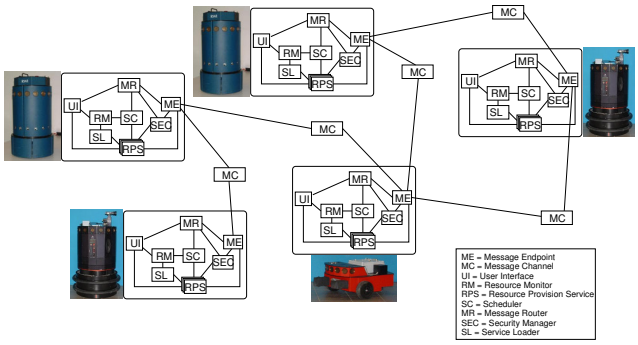

Figure 1. Example of a small network of peers depicting peer internal structure and interconnection.

2.1 Adaptive Routing of Distributed Control Messages

Generally speaking, a peer-to-peer architectural model is characterized by an overlay network topology and a routing strategy. The Message Router module we introduced above adapts its behaviour to the network topology, in order to implement the self-configuration property of the autonomic

system. This is particularly important for large distributed robot systems with multihop wireless ad hoc network infrastructure, in which networking services are provided by the nodes themselves. Cooperation among nodes cannot be taken for granted, but it has been shown that the network topology and the communication patterns have a significant impact on the existence of spontaneous cooperation [9].

To cover a wide range of peer-to-peer systems, both structured and unstructured, we consider the Selective Query Model (SQM), which defines overlay architectures in which peers with higher process and storage capacity act as supernodes, assuming the responsibility of propagating control messages (*i.e.* publication and search messages), while peers with less capacities (leaf nodes) must connect to supernodes to send control messages in the network, but do not contribute to the overall routing process. Typically, each leaf node is connected to a single supernode. On the other side, supernodes maintain many leaf connections, as well as a small number of connections to other supernodes. This hierarchy leads to scalable systems, in which supernodes shield leaf nodes from virtually all ping and query traffic. The model is illustrated in figure 2.

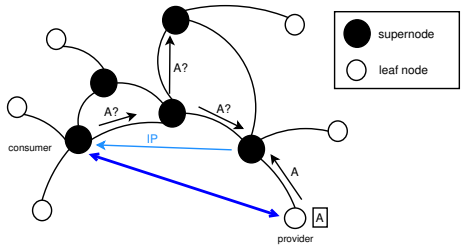

Figure 2. The SQM architectural model.

A good SQM formalization is provided by JXTA [18], a set of open, generalized peer-to-peer protocols that allow a vast class of networked devices (smartphones, PDAs, PCs and servers) to communicate and collaborate seamlessly in a highly decentralized fashion. JXTA aims at reducing the complexity otherwise required to build and deploy peer-to-peer applications and services by providing an open source software platform and deployed virtual network.

In JXTA, all resources, services, peers and peergroups are described by XML documents, the so-called advertisements. Resource sharing and discovery is based on two mechanisms: the *Shared Resource Distributed Index (SRDI)*, and the *limited range walker*. The SRDI module implemented in each JXTA peer is used on one hand to extract entries from resource advertisements and push them to the network, and on the other hand for lookups. When an advertisement is published by a leaf peer, its entries (which are attribute-value pairs) are sent to the connected supernode. Supernodes store the entries in dynamic indexes including also, for each entry, the ID of the peer which originated them and an expiration time. Moreover, each index entry is replicated from the supernode to another supernode using a mapping function [18]. Search is based on the same mapping function, but also on the limited range walker to resolve inconsistencies within the dynamic supernode network. The process is illustrated in figure 3, for both the ideal case of complete supernode network, and the more common case of uncomplete supernode network.

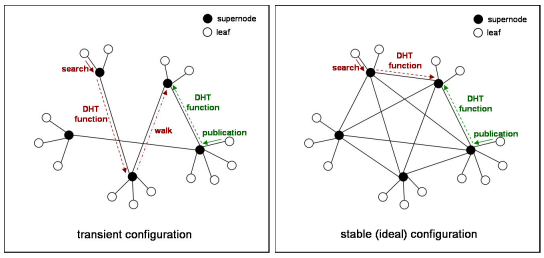

Figure 3. JXTA routing with the SRDI strategy.

The more the supernode network is near to completeness, *i.e.* the peerview is consistent across all supernodes, the more this routing strategy is efficient. By means of simulations, we observed that JXTA's SRDI algorithm is suited for Poisson or quasi-complete supernode topologies. Searching for a complementary strategy, to adapt message routing also to scale-free networks, *i.e.* networks with few highly connected nodes, we defined *HALO*. HALO is based on high-degree node search, *i.e.* control messages for both search and publication are routed choosing at each step the most connected neighbor, and using the SRDI mapping function for corrections and for the final hop. Figure 4 shows the efficiency (% of resolved queries) of JXTA's SRDI and HALO in the two kinds of topology (with a large number of nodes).

2.2 Service Sharing and Orchestration

In this section we consider complex tasks which may be accomplished by several interacting services, provided by different robots/sensors/devices. Web Service orchestration and choreography, service interoperability and e-procedures, transactional semantics are some of the most emerging standardization efforts of the IT community (*e.g.* Workflow Management Coalition, and Business Modeling & Integration Domain Task Force). In this context, Service Level Agreements (SLAs) are a key point because they provide guarantees about how a service will be provided, or consumed by establishing both functional and non-functional requirements that must be fulfilled by parties during service execution. The process of locating, selecting, negotiating, and creating SLAs is defined as *service trading* [16], and its automation is a key characteristic fea-

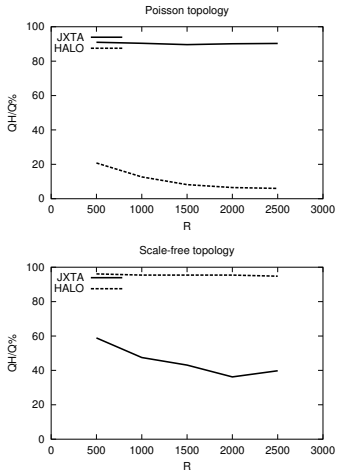

Figure 4. Efficiency of JXTA's SRDI and HALO routing strategies in different topologies; R is the number of supernodes, $i.e$ peers which are allowed to propagate control messages.

ture also for fully autonomous and autonomic distributed robot systems.

With reference to the conceptual framework depicted in section 2, service-oriented resource sharing among networked peer robots can be summarized by the following three phases.

1. For each resource, a Resource Provision Service is deployed and its description (interface), including information such as security requirements, is made available to the Message Router. On resource discovery, (possibly complex) query messages are delivered to the Message Router. No centralized registry is used, since publication and discovery processes are fully distributed.

2. Based on the received input, the Message Router creates new messages and determines their destination. On resource discovery, queries can be either syntactically or semantically evaluated. This process can affect the choice of next destination peer.

3. The Message Router sends query-related messages to the computed destination.

Consumers are allowed to check if a discovered RPS is available, if it performs a certain function or a set of functions, if it operates under specified constraints, and if it can be invoked through a specified means, including inputs that the service requires and outputs that will form the response to the invocation. Capturing service functionalities is a difficult goal to achieve. This aspect needs to be expressed in a way that is generally understandable by service consumers, but able to accomodate a vocabulary that is sufficiently expressive for the domain for which the service provides its functionalities. This may include, among other possibilities, the definition of RPS semantics, which are the shared expectations associated with the resource. RPS consumers perceive the semantics as broken in three main parts: the data model, the process model, and the behaviour. The latter is the intended real world effect of using a RPS. Semantic description of RPSs allows for fine-grain tuning of the discovery process, in order to increase its efficiency, fault tolerance, scalability and autonomicity.

3 Enabling Technologies

The conceptual framework has been implemented in the SP2A middleware [5], which enables semantic annotation of services, secure peergroup creation and selection, reputation management, and service code mobility.

SP2A is based on Java technologies, taking advantage of its strong support for Web Service specifications, peer-to-peer networking, code mobility and runtime service deployment. In particular, Java provides a programmable mechanism, the class loader, which allows to instantiate classes in the Java Virtual Machine (JVM) at runtime. With this facility, it is possible to support both weak mobility, for which services are downloaded and deployed as new entities with no memory of their state at the source, and the more complex scenario in which resources at destination node must reflect the configuration of those at the source node.

SP2A is based on JXTA, namely on the JXTA-SOAP [4] component we are developing. JXTA-SOAP's purpose is to fill the gap between JXTA and Web Services, two technologies which may complement each other. By focusing on messages, the Web Service model is completely language, platform, and object model-agnostic. For these reasons, Web Service technologies are more suitable to define high level tasks rather than Distributed Object Computing (DOC) technologies like CORBA. On the other side, JXTA's peer-to-peer underpinnings can provide efficient scalable alternatives for Web Service discovery and communication to current centralized solutions such as UDDI. Currently, JXTA-SOAP's API allows to dynamically deploy stateless Web Services wrapped in JXTA services, which can be published and discovered according to the previously illustrated strategies. Moreover, JXTA virtual channels are used to transport SOAP messages for Web Service interactions, with transport-level security (based on TLS) or application-level security (based on WSS).

Semantic annotation of RPSs in SP2A is based on OWL-S [2], which is a Web Service ontology supplying a core set of markup language constructs for describing the properties and capabilities of Web Services in unambiguous form.

4 Application Scenarios

Our testbed is a system of networked robots and PCs, each one hosting a SP2A peer (figure 5). The main robot platforms we use are a Pioneer 1-DX, provided with a Via C3 1.3GHz processor and 256MB RAM, and a Nomad 200, equipped with a Pentium III 1GHz processor and 256MB RAM. Both robots deploy a RobotDescriptionService, which exposes the robot's characteristics and current status, and an ExplorationService, which is able to perform many operations, such as getPosition and getGlobalMap. To access local resources (sensors, actuators), the service interacts with the Player robot server, which is a widely used robot control interface [8].

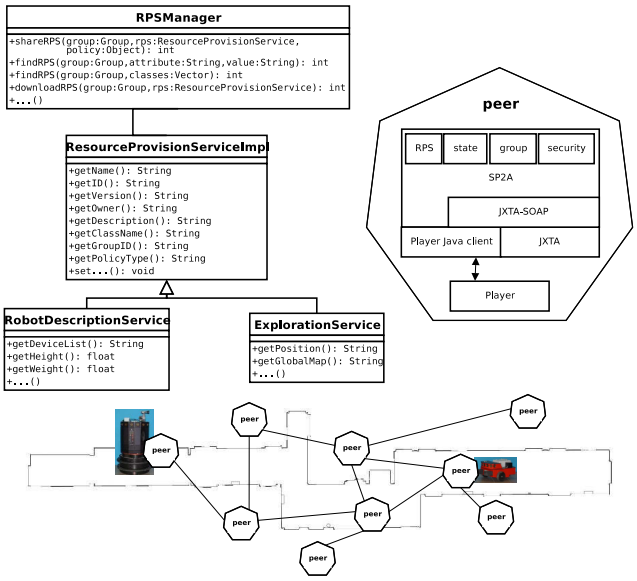

Figure 5. An overlay network including two peer nodes hosted respectively by a Nomad 200 and a Pioneer 1-DX. All nodes are equipped with SP2A. Robot peers perform collaborative exploration tasks in an indoor environment, sharing the services which are illustrated in the class diagram.

In the following we illustrate two robotic application scenarios we are investigating with the deployed SP2A-based system.

In the first scenario, a mobile robot (R_A) explores a known environment (*e.g.* the rooms of a museum at night) and offers a human recognition service (*e.g.* for intrusion detection). Suppose the robot has almost discharged batteries, and will not be able to serve its peers during the recharge process. For this reason R_A starts searching the network for a substitute. Each peer robot exposes the RobotDescriptionService. The substitute for robot R_A must be provided with the same hardware resources (with comparable or better performance), but also be able to quickly reach the environment in which R_A operates (which means that proximity is a considerable parameter). Service discovery mechanisms based on adaptive message routing (illustrated in section 2) allow R_A to find R_B, which we suppose to be the best substitute for R_A. Assuming reciprocal authentication has been performed, R_A sends its coordinates to R_B, eventually with a multi-hop path if R_B is not directly reachable, then R_B move towards R_A. Once R_A and R_B are able to establish a direct connection, robot R_A sends the human recognition service code to robot R_B, which immediately deploys it, and publishes its description in the network. On the other side, robot R_A redirects its service consumers on robot R_B, undeploys its human recognition service, and move towards a power charging station. The overall process should be as *smooth* as possibile, with consumer peers noticing only a reduced level of service, rather than complete failure.

The second application involves an ad-hoc network of mobile robots which enter an unknown environment, with the global goal of exploring it, covering the whole area as quick as possible. Once they are inside the target area, robots move around, each one choosing a different direction, updating its local map, and communicating with its neighbors to build a global map. Each robot exposes the ExplorationService which provides the robot's global map. The global map of each robot is updated periodically, or when a neighbor asks it for urgent map merging. There are several situations in which urgent map merging is required. For example, a robot can be blocked by an obstacle, or damaged, for which it provides its global map to its neighbors, and also its current position. This process can also be started when a robot needs to stop for power charge exhaustion, too.

5 Related Works

In the research field of networked robots, a recent work by Ha *et al.* [10] proposes the automated integration of distributed robots, sensors and devices into ubiquitous computing environments based on semantically enriched Web Services. The framework consists of three major components: an intelligent planning and semantic Web Service requester agent (SA), an environmental knowledge repository (EKR), and the device Web Service (DWS). The SA relies on Hierarchical Task Network (HTN), an AI planning methodology that creates plans by task decomposition. In the prototype system, the SA is hosted by a dedicated server, as well as the EKR. A very similar framework is proposed by Kim *et al.* [13], in which Web services are on the robots, and a system of external servers constitutes the knowledge base of the system. Our approach is more oriented to reactive archi-

tectures, and service composition is a consequence of interactions with the environment and with other peers, not the result of planning. In our framework, no dedicated servers are needed for planning and knowledge storage, since no planning is required, and information are distributed in the peer-to-peer robot network.

In [3], Ahn *et al.* propose an approach to utilize Universal Plug and Play (UPnP) as a middleware for networked robots. UPnP defines an architecture that offers pervasive peer-to-peer network connectivity of PCs, intelligent appliances, and wireless devices, enabling seamless proximity networking in addition to control and data transfer. Moreover, UPnP provides a service-oriented programming model based on XML. The major issue of UPnP is the complexity of its security protocols, and the lack of protocol built-in Access Control List (ACL) to define who can access and send orders to UPnP devices.

The Microsoft Robotics Studio (MRS) [1] commercial tool provides a service-oriented architecture which combines key aspects of traditional Web-based architectures with pieces from Web Service technologies. In particular, the MRS runtime adopts the Representational State Transfer (REST) model as its foundation, but extending it with structured data representation and event notifications from the Web Service world.

6 Conclusions and Future Work

In this paper we described a conceptual framework enabling service-oriented peer-to-peer networked robot applications. The framework allows for peer robots connectivity (within unstructured overlay networks with supernodes), adaptive strategies for control message routing among networked peers, as well as service deployment and mobility.

The implementation with our SP2A middleware has been described, along with two application scenarios of cooperating mobile robots taking advantage of the capabilities offered by the service-oriented framework to improve system performance and robustness.

Currently, only weak service mobility has been implemented in our SP2A middleware. Next step is to provide the JXTA-SOAP component with WSRF support and then endow SP2A with resource state mobility.

Another challenging problem we are facing is the implementation of lightweight routing strategies for semantic service discovery. This would allow better interoperability among robots whose services expose different interfaces but are equivalent in the tasks they perform.

References

[1] Microsoft Robotics Studio. http://msdn.microsoft.com/robotics/.

[2] OWL-S: Semantic markup for web services. Technical report, W3C, 2004.

[3] S. C. Ahn, J. H. Kim, H. Ko, Y.-M. Kwon, and H.-G. Kim. UPnP Approach for Robot Middleware. In *International Conference on Robotics and Automation (ICRA05)*, Barcelona, Spain, 2005.

[4] M. Amoretti, M. Bisi, and A. Panisson. JXTA-SOAP Project. http://soap.jxta.org.

[5] M. Amoretti, F. Zanichelli, and G. Conte. SP2A: a Service-oriented Framework for P2P-based Grids. In *3rd International Workshop on Middleware for Grid Computing, Co-located with Middleware 2005*, Grenoble, France, 2005.

[6] P. Braun and W. Rossak. *Mobile Agents - Basic Concepts, Mobility Models and the Tracy Toolkit*. Morgan Kaufmann Publishers, 2005.

[7] R. R. Brooks. Mobile code paradigms and security issues. *IEEE Internet Computing*, 8(3):54–59, 2004.

[8] T. H. J. Collett, B. A. MacDonald, and B. P. Gerkey. Player 2.0: Toward a Practical Robot Programming Framework. In *Australasian Conference on Robotics and Automation (ACRA 2005)*, Sydney, Australia, 2005.

[9] M. Felegihazi, J.-P. Hubaux, and L. Buttyan. Nash Equilibria of Packet Forwarding Strategies in Wireless Ad Hoc Networks. *IEEE Transactions on Mobile Networks*, 5(5):463–476, 2006.

[10] Y.-G. Ha, J.-C. Sohn, and Y.-J. Cho. Service-oriented integration of networked robots with ubiquitous sensors and devices using semantic web services technology. In *International Conference on Intelligent Robots and Systems (IROS 2005)*, Edmonton, Alberta, Canada, 2005.

[11] G. Hohpe and B. Woolf. *Enterprise Integration Patterns*. Addison-Wesley, 2003.

[12] P. Horn. Autonomic computing: IBM perspective on the state of information technology. In *Agenda 2001*, Scottsdale, Arizona, USA, 2001.

[13] B. K. Kim, M. Miyazaki, K. Ohba, S. Hirai, and K. Tanie. Web services based robot control platform for ubiquitous functions. In *International Conference on Robotics and Automation (ICRA05)*, Barcelona, Spain, 2005.

[14] S. Y. Lee, O.-H. Kwon, J. Kim, and S. J. Hong. A reputation management system in structured peer-to-peer networks. In *14th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprise*, Linkoping, Sweden, 2005.

[15] M. Narasimha, G. Tsudik, and J. Yi. On the utility of distributed cryptography in p2p settings and manets. In *IEEE International Conference on Network Protocols (ICNP'03)*, Atlanta, Georgia, USA, 2003.

[16] M. Resinas, P. Fernandez, and R. Corchuelo. Towards Automated Service Trading. In *International Joint Conference on e-Business and Telecommunications (ICETE 2006)*, Setubal, Portugal, 2006.

[17] C. Rouff, M. Hinchey, J. Rash, W. Truszkowki, and R. Steritt. Autonomicity of NASA Missions. In *Proc. of the Second International Conference on Autonomic Computing (ICAC'05)*, Seattle, Washington, USA, 2005.

[18] B. Traversat, A. Arora, M. Abdelaziz, M. Duigou, C. Haywood, J.-C. Hugly, E. Poyoul, and B. Yeager. Project JXTA 2.0 Super-Peer Virtual Network. Technical report, Sun Microsystems, 2003.