

An Architecture for a Service Oriented Peer-to-Peer System (SOPPS)

Jan Gerke¹, David Hausheer¹, Jan Mischke¹, Burkhard Stiller^{2, 1}

¹ Computer Engineering and Networks Laboratory, Swiss Federal Institute of Technology Zurich, Switzerland

² Information Systems Laboratory, University of Federal Armed Forces Munich, German

[gerke | hausheer | mischke | stiller]@tik.ee.ethz.ch

Abstract: Peer-to-peer (p2p) systems have regained much attention during recent years. While this is mostly due to their two main applications filesharing and instant messaging, this paper proposes to create a p2p system with generic support for services of any kind, called SOPPS. Furthermore, it proposes to use market mechanisms to manage these services. In order to provide this functionality and to make full use of key advantages of p2p, SOPPS must fulfill more detailed requirements, which have been investigated in this paper. Three major models found the SOPPS architecture and describe in particular the use of SOPPS, its underlying network, and the architecture of a SOPPS peer. Finally, the application of this architecture as well as of its interfaces are described for a content sharing scenario.

1 Introduction

In recent years, an old concept gained renewed attention within the Internet community. After client/ server structures had managed to dominate the Internet, several new distributed systems, which quickly became very popular, made use of peer-to-peer (p2p) structures. These systems provided two different kinds of applications, namely file sharing [1] [2] (e.g., Napster, Gnutella, or Morpheus) and instant messaging (e.g., ICQ [3], AOL Instant Messenger, or Jabber [4]).

Many of these systems might also provide simple functionality of other applications. For instance, a file sharing system might include simple chat functionality or an instant messaging system might support special folders to share files with friends. However, existing peer-to-peer systems do not offer a generic support for arbitrary distributed applications, but they are limited to the two already mentioned ones.

While other systems offer remote access to arbitrary applications (e.g., .NET [5] and SunONE [6]), they rely on centralized structures (e.g., Microsoft's Passport service or UDDI-based service directories [7]). Therefore, they are not able to offer the complete range of benefits a pure p2p-based solution can offer. Especially, there is always a central point of control and, hence,

failure. It is, therefore, highly unlikely that user communities will grow based on these systems, in contrast to p2p-based file sharing and instant messaging systems which have grown very fast. However, existing p2p systems also brought along different problems, especially for commercial companies. Since these systems can not be controlled centrally, they offered an excellent platform for sharing content - often copied illegally. To gain the best from all three worlds (robust p2p structures, generic applications, and control), it is necessary to design an appropriate Service Oriented Peer-to-Peer System (SOPPS). In this paper the architecture of SOPPS is outlined. It is based on p2p structures and offers a generic support of services as well as supply mechanisms to enable market management of those services offered (compare [10]). This goal corresponds to the major objective of the EU-funded project MMAPPS (Market Management of Peer-to-Peer Services) [8], where ETH Zürich participates.

The remainder of this paper is structured as follows: Section 2 states those requirements SOPPS has to fulfill. Section 3 introduces the SOPPS architecture based on underlying models, and an example for the usage of SOPPS in a content sharing scenario. Finally, Section 4 concludes and addresses future work.

2 Requirements

The SOPPS architecture has to meet two fundamental goals: maintain and fortify the advantages inherent to the p2p approach found in existing systems, while avoiding current weaknesses. This leads to the following set of detailed requirements, which form the basis for the architectural design:

Maintain and fortify p2p advantages:

- **Decentralization:** A p2p system must use only p2p mechanisms and must be able to function without any central components. Only this type of approach offers the full advantage of the p2p concept and ensures that no central point of failure exists.

- **Communities:** One of the big advantages of popular p2p systems is that their users feel part of a big community. In order to exploit this advantage, the system has to offer specific community support for its users, e.g., the forming of groups.
- **Performance:** No matter what functionality the architecture offers, the system cannot be successful without offering sufficient performance, especially fault tolerance, extensibility, efficiency, and scalability, the latter two being only partly addressed in current solutions.

Counter weaknesses:

- **Generic service support:** The SOPPS architecture needs to support completely different services, including management of content and other resources. Hence, mechanisms such as service fusion, i.e. the combination of existing services to create new, value-added services, and service description are required.
- **Market management:** The important goal is the creation of a marketplace of services, managed by true market mechanisms. On one hand, the traditional way for this is pricing of services in any currency. On the other hand, rules which the market participants have to follow are a suitable alternative and have to be supported, e.g., 'you can only use a service if you also provide one'.
- **QoS support:** A p2p system can not completely form a market of services, if it is not able to ensure that services are provided reliably. Thus, it is necessary to employ QoS (Quality-of-Service) mechanisms in the network, but also on peers themselves. Especially, the support of Service Level Agreements (SLAs) is necessary to provide a concise and legal foundation for any service usage.
- **Security:** Finally, if arbitrary services are to be offered over the p2p system, security becomes the key in a commercialized environment. This includes authentication and authorization of users, encryption of data transfers, trust of users in other users, peers and services, and anonymity of users.

3 SOPPS Architecture

The architectural design of SOPPS builds on those requirements introduced and founds on the following three key models (see also [11]). Peers are defined as nodes connected to a network, running the respective SOPPS software. SOPPS consists of peers communicating over a network in order to provide or use services. This concept

leaves three questions to be answered by distinct models:

- How do peers provide or use services? By the Use Model (Section 3.1)
- What do communication and the network look like? By the Network Model (Section 3.2)
- How does a peer look like and how is it decomposed? By the Peer Model (Section 3.3)

3.1 Use Model

Figure 3.1 illustrates the service usage as it is to be supported within SOPPS. A service is the provisioning of resources or the execution of tasks of one or more (temporarily provider) peers on behalf of one or more (temporarily user) peers. A human user or, alternatively, an electronic agent acting as a user, is associated with one peer node in the network. The user is assumed to have an objective that can be fulfilled by the services offered by SOPPS. On the corresponding peer, it invokes functionality to discover an appropriate service and to determine and contact one or more provider peers offering that particular service. Market management requires that SLA be agreed upon before the service can be used or fulfilled, respectively. Access to the service will be negotiated between the involved parties or controlled by service-defined rules.

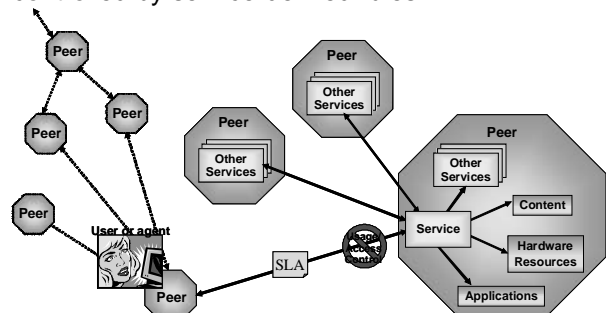


Figure 3.1: The Use Model

In order to fulfill the service, the provider peer can draw on its local resources like hardware, applications, or content. In a more complex scenario, the service may also require to use (and fuse) other services residing on the provider peer. Finally, even the delegation of tasks to or use of resources from further peers is possible.

3.2 Network Model

Peers are attached to and communicate over a network as indicated in Figure 3.2 (left). Usually, this network will be the Internet with all its complexities, like packet routing over multiple hops and across domains; they are hidden in the figure behind the 'network cloud'. Only some of the hosts connected to the Internet are part of

SOPPS (indicated by 'P' in the figure). They maintain end-to-end application-level connections (red lines).

The network model in Figure 3.2 (right) abstracts away from these complexities and considers only participating peers as connected by an application-level overlay network. The indicated links correspond to end-to-end connections in the underlying network and are associated with QoS parameters to allow for QoS modeling and provisioning.

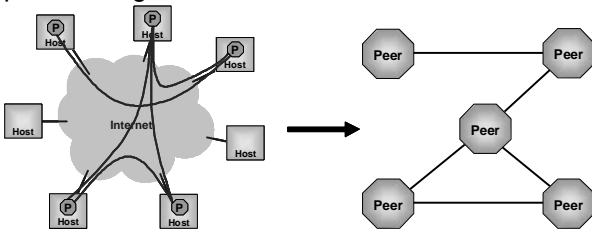


Figure 3.2: From Internet Network Model (left) to Overlay p2p Network Model (right)

This abstraction has three main advantages. Firstly, the network model can be limited to those hosts, which actually participate in the p2p system. Secondly, the contemplation of overlay links vastly simplifies the model without losing relevant information. Finally, an abstraction from the underlying network alleviates the portability of SOPPS onto different networks. All communication and messaging (e.g. for search) amongst peers takes place on the overlay network. Proper construction of the overlay topology and efficient design of overlay routing hence strongly impact the scalability of SOPPS.

3.3 Peer Model

The most important model is the model of a peer itself, which is depicted in Figure 3.3. Each peer has a number of local resources available at the bottom layer. These resources consist of hardware and software resources and can be used either for providing services to other peers or for support of the Core Functionality. This functionality includes all mechanisms which are necessary to run SOPPS itself, such as resource management or service discovery. In order to provide some of these mechanisms, it is necessary for the core functionality to communicate with other peers. For instance, service discovery is based on a distributed protocol where peers forward service search requests amongst each other.

The service layer supports two different types of services. Resource encapsulation services are very simple services which only provide access to local resources, e.g., storage space. The encapsulation is required to enable remote access to these resources over standardized interfaces

and to allow market management through access rules and negotiations. Advanced services, on the other hand, contain much more functionality and may offer complete applications. In order to access local resources they can either directly build on them or on resource encapsulation services. The advantage of the second approach is the possibility to later include remote resource encapsulation services, e.g., if the local storage space is getting scarce.

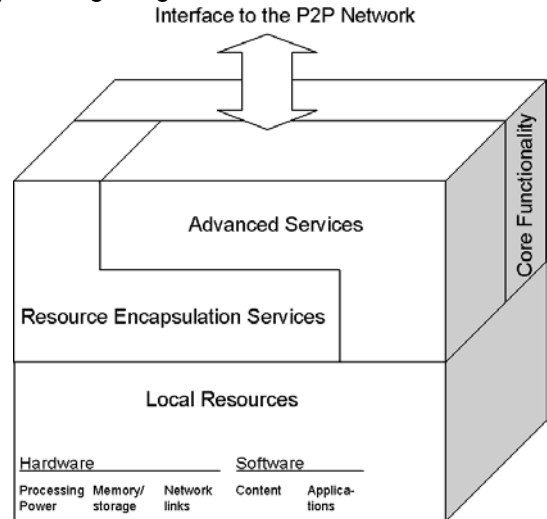


Figure 3.3: The Peer Model

The peer model presented here describes the architecture of a single peer. Several of such peers, working together arbitrarily, form SOPPS. To detail the SOPPS architecture further, it is necessary to break down the peer model into smaller components as well as to investigate the interaction of these components and the interaction between different peers according to the use and network models. Therefore, the detailed peer model is shown in Figure 3.4.

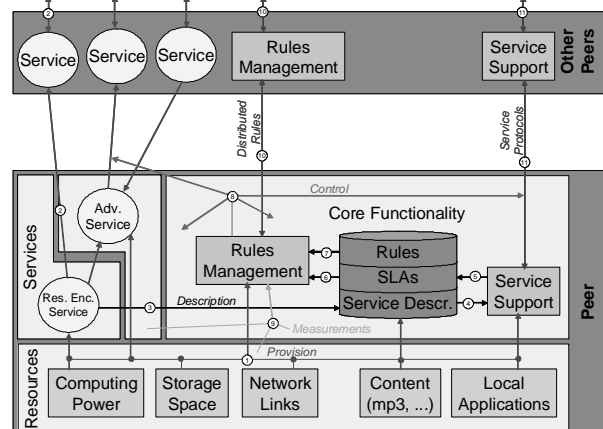


Figure 3.4: Detailed Peer Model

It depicts a peer interacting with another peer, representing all the other peers of the p2p system. The peer contains all those four layers, shown in light grey, as introduced in Figure 3.3, namely the resource layer, the resource encapsulation and

advanced services layer, and the core functionality layer. Within these layers major components have been identified, denoted as circles (services), boxes (resource and core functionality components) and database symbols (central storage component), as well as the interfaces between these components, denoted as arrows.

3.3.1 SOPPS Components

Instead of designing a peer as one monolithic block, several components have been identified in Figure 3.4. These components are entities, which provide functions dealing with similar problems. This enables for the independent investigation of different components and relations between similar and diverse functions become visible. In more detail, those components maintain the following tasks:

Resource components: Five different types of resources have been identified. All of them can be offered to other peers via services or can be accessed by the core functionality in order to provide the SOPPS functionality. For the desired generic service support, it is necessary to avoid a limitation of the type of resources offered or the way how they are accessed. Content can consist of files like audio or video files, which are transmitted as a whole, but must also be accessible via streaming techniques. It should also be possible to share complete local applications within SOPPS. Computing power and storage space are used by both previous resources, but should also be accessible separately as independent resources. Finally, it is necessary to use network links for the data transfer between services, but also for the signaling between the core functionalities of different peers.

Network links between peers are links in the overlay network model and are assumed to be end-to-end connections of the Internet transport layer as described in Section 3.2. These end-to-end connections can also provide QoS, using QoS technologies like Differentiated Services.

The exact way of resource provision heavily depends on the platform the local peer is based on, especially its operating system. For instance, if the operating system does not provide necessary process scheduling mechanisms, it becomes very difficult to manage computing power in a deterministic way.

Services: Services are offered by one peer to other peers. They form a common interface to the local resources and can be simple resource encapsulation services or complex advanced services as described in Section . They are mentioned as a component here, since they must provide also standard functionality, for instance interfaces to the rules management component to

report their status, or to other services for initialization of the service usage.

Rules Management: The rules management component is the key management component within a peer. It controls all activities taking place between the peer and other peers and might also control local activities. It ensures that the activities comply with rules defined. In order to decide, whether a rule is fulfilled, it must be able to receive measurements from all other peer components.

Rules themselves must support the definition of entities like groups or methods of reputation building based on common initial definitions. This way, the required market management of services is implemented building on rules that service providers and users have to follow. More details about rules, groups and reputation can be found in [12].

Service Support: The service support component is responsible for handling service related issues. This includes among others the management of service descriptions which are used in service discovery and negotiation processes, leading to the creation SLAs.

Central Storage: The central storage is a database where all SOPPS relevant data is stored, especially rules, service descriptions and SLAs. The central storage is central to one peer, not to the overall SOPPS. Rather, each peer within SOPPS contains its own central storage.

3.3.2 SOPPS Interfaces

All interfaces shown as arrows in Figure 3.4 determine a component-to-component interaction. Apart from interfaces #3, #10, and #11 all interfaces are local interfaces located within one peer and enable the necessary communication between its components. Interface #3 is a remote interface from one peer to another to enable the access to services. Of special interests are the two remaining remote interfaces #10 and #11, which take the form of distributed protocols transporting information between many peers. The interaction of all interfaces is refined as follows:

- 1 **Resources** → **Services and Core Functionality Components:** Local resources have to be provided to services running on a peer, but also to SOPPS' components themselves, e.g., they need storage space for the central storage, computing power for running algorithms and network links for running distributed protocols like service search.
- 2 **Services** → **Services:** It must be possible to access services offered by one peer from another peer. This requires that the remote peer runs a service which handles the

communication with the local service via this interface.

- 3 **Services** → **Core Storage**: For each service a peer offers it must maintain a service description inside its central storage, which is an important part of service search and negotiation. In practice it will be almost impossible to automatically generate service descriptions. Therefore, this interface most likely consists of a system manager entering an appropriate service description for a new service into the central storage.
- 4 **Central Storage** → **Service Support**: In order to enable the service support component to respond to service search messages arriving from other peers and to successfully negotiate the remote access to services, it must access the service descriptions stored in the central storage.
- 5 **Service Support** → **Central Storage**: After a service negotiation has been successfully carried out, the resulting SLA has to be stored inside the central storage.
- 6 **Central Storage** → **Rules Management**: During the provision of a service (i.e. during a session), the fulfillment of the SLA has to be ensured. This can be monitored by the rules management component, which treats the SLA as a rule which has to be fulfilled. Therefore, the rules management component can access the SLAs inside the central storage via this interface.
- 7 **Central Storage** → **Rules Management**: The rules which are to be applied to SOPPS can be retrieved from the central storage by the rules management component via this interface.
- 8 **Rules Management** → **Local peer**: To fulfill the defined rules, the rules management component must be able to control all other parts of the peer, which is done via this interface.
- 9 **Local peer** → **Rules Management**: In order to decide when an action has to be taken in order to fulfill a rule, the rules management must get measurements from all other parts of the peer, which arrive via this interface.
- 10 **Rules Management** ↔ **Rules Management**: A special case of rules are distributed rules which can only be fulfilled by several peers working together. The communication which is necessary to organize this cooperation is done via this interface. This can also include the conducting of a ballot, e.g., to decide whether a new peer should be accepted into a peer group. The necessary voting can be carried out via this interface.
- 11 **Service Support** ↔ **Service Support**: All communication between different service

support components takes place at this interface, which consists of several distributed protocols supporting different uses. For instance, when a peer wants to use a service from another peer it must first have knowledge of this service. This knowledge can be gained by carrying out a service search. This distributed protocol searches a number of peers for a specified service. While the detailed nature of the protocol is not defined within the architecture, the interaction takes part via this interface. After a peer has discovered a service which can fulfill its requirements, it can enter service negotiation which is also carried out via this interface but will use a different protocol. During this negotiation phase the parameters of the service are fixed for the duration of the usage. If the negotiation is successful, a SLA is created.

3.4 Example Scenario

The adaptability of the three above presented models to a real environment is illustrated in a content sharing scenario. SOPPS also supports more innovative applications, e.g., distributed virtual worlds or distributed marketplaces. However, this traditional application was chosen, because it is well known, and therefore demonstrates the differences to common p2p systems very well.

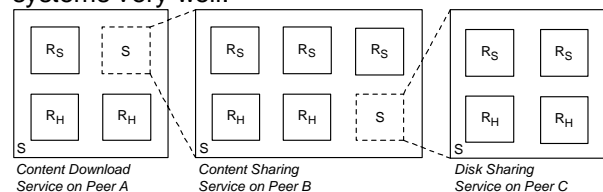


Figure 3.5: The Service Model

The example shown in *Figure 3.5* represents three services - located on three different peers A, B, and C - which are involved in the overall content sharing process. On every peer several local hardware (R_H) and software (R_S) resources exist, which are encapsulated or directly used by these services. In the presented example a 'content download service' on peer A retrieves a software resource, e.g., a music file, from a content sharing service on peer B. In order to provide the 'content sharing service', peer B would need to integrate a couple of local resources, such as the service application as a software resource itself, some storage space, and processing power, to store and process the service application, and network links to deliver the provided content. Additionally, peer B might integrate some additional services, e.g., a 'disk sharing service' from peer C to store the according content data. As it can be seen in

Figure 3.5, the service offered by peer C is a 'resource encapsulation service' providing remote access to local resources. On the other hand, peer B offers an advanced service, since it also contains the service of peer c. The service on peer a is local advanced service, since it contains other services but is not offered to other peers. In the following the interaction of the different components as shown in Figure 3.4, which is necessary to provide the example services, is shown.

Service search phase: Before using a service a user must first find the service he wants to use. In this example he is looking for some specific content, e.g., a live video broadcast from a sports event. In order to find peers which offer this content, he creates a service description describing the content and his peer sends out a service request containing this description to other peers, using interface #11 shown in Figure 3.4. This request is forwarded by the peers according to a search protocol. The user's peer then gets the results of the search, in the form of service offerings by Peers, which provide a service according to the service description. While the exact nature of the search protocol (e.g., centralized as in Napster or flooding as in Gnutella) does not influence the architecture, it must be able to scale to a large number of peers to prevent a large signaling overhead filling the available network resources. Since it might not be possible to find an exact match for the user's service description, it is also necessary to support a matching of service descriptions according to a predefined accuracy. The matching is carried out by comparing the received service descriptions with the descriptions of local services, which previously have been stored in the database via interface #3 and are now retrieved using interface #4.

Service negotiation phase: After a user has found one or several possible services he enters a negotiation with their providers to define the exact terms of service usage. This includes the fixing of parameters, e.g., the format the content will be encoded in, the data rate for sending the content, the price the user might have to pay for the service. This negotiation also takes place via interface #11, though a different protocol is used. If user and provider agree on the terms of service usage, they record this in a service level agreement which is stored in the provider's central data storage via interface #5.

Service usage phase: After the terms of service usage have been successfully negotiated the user can start using the service via interface #2. Especially, this includes the remote invocation of methods offered by services during an initialization phase. Afterwards, the data, e.g., the sports event video stream, can be exchanged via

sockets and special protocols agreed upon in this phase. This communication can then also be carried out using an end-to-end QoS technology, if the SLA specifies this. During the service usage the service uses local resources of its peer using interface #1. Especially, this includes the usage of network links in order to communicate with the peer using the service.

Application of rules: Throughout all the processes described above the rules management component watches over the fulfillment of previously defined rules. To this end it can receive status information from all other peer components via interface #9, and influence the actions of these components via interface #8. For instance, during the service search phase the rules management component might decide to offer services only to peers which are members of a previously defined group or who have a high enough reputation for paying for services used. During the service negotiation phase special conditions for usage might be offered to peers of a specific group or only advance payment might be offered to peers having a bad reputation for paying. During the service usage phase the rules management component is responsible for ensuring the fulfillment of the SLA. If the SLA is violated it can trigger a recompense action defined in the SLA. The rules management component can retrieve rules and SLAs from the central storage using interfaces #7 and #6, respectively. Furthermore, for the management of distributed rules, it must also contact the rules management components of other peers. For instance, it might be necessary to ask other peers about what they know about a potential service user, to determine his reputation.

4 Conclusions and Future Work

In this paper a new peer-to-peer system called SOPPS has been proposed. SOPPS is able to support services of any kind, going beyond filesharing or instant messaging, by being able to share local resources and applications in general. Basic requirements for SOPPS have been presented and its architecture has been described. This architecture is based on three models, which demonstrate the use of SOPPS, its view of the Internet-based overlay network connecting all peers, and the architecture of peers themselves. Finally, a sample application of SOPPS has been shown in a content sharing scenario, in order to demonstrate its principle feasibility.

The next steps on this work include the definition of components and their interfaces in a pre-implementation method, covering detailed interaction diagrams for this distributed system.

Classes, methods, and suitable protocols for the communication between all peers will be designed and implemented. Especially, when designing those protocols for communication between different peers, it is of utmost importance to fulfill performance requirements and scalability. Therefore, a peer will be constructed in such a way, that different protocols can be plugged into the implementation of SOPPS without having to make changes to other parts of the peer.

In the meantime, the architecture has been refined. Especially, it now contains components supporting security, charging, and group management. Currently, a quite likely basis for a SOPPS implementation is JXTA [9], since it is platform independent, its source code is available, and in contrast to SunONE or .NET it builds completely on the p2p paradigm. For instance, in order to maintain SOPPS' decentralized nature, security can be provided by using distributed techniques, e.g., the JXTA project Poblano. A first investigation of JXTA indicates that it is very well suited for the implementation of SOPPS.

Acknowledgment: This work has been performed partially in the framework of the EU IST project MMAPPS 'Market Management of Peer-to-Peer Services' (IST-2001-34201), where the ETH Zürich has been funded by the Swiss Bundesministerium für Bildung und Wissenschaft BBW, Bern under Grant No. 00.0275. The authors want to thank their partners from Hewlett Packard, British Telecommunications, the Athens University of Economics and Business, and the Technical University of Darmstadt for many fruitful discussions and their valuable input to this work.

5 References

- [1] Frode Kileng: *Peer-to-peer file sharing technologies - Napster, Gnutella and beyond*; Telenor Forskning og Utvikling, 2001
- [2] Dejan Milojevic: *Peer-to-Peer Computing*; HP Labs Technical Report, 2002
- [3] ICQ inc.: *ICQ.com homepage*; <http://web.icq.com/>, 2002
- [4] Jabber Software Foundation: *Jabber homepage*; <http://www.jabber.org/>, 2002
- [5] Microsoft .NET Framework: *Product Overview*; <http://msdn.microsoft.com/netframework/productinfo/overview/default.asp>, 2002
- [6] Sun Microsystems: *SunONE homepage*; <http://www.sun.com/software/sunone/>, 2002
- [7] Tom Bellwood et al.: *UDDI Version 3.0 Published Specification*; <http://uddi.org/pubs/uddi-v3.00-published-20020719.htm>, 2002
- [8] MMAPPS Consortium: *Market Management of Peer to Peer Services*; <http://www.mmapps.org/>, 2002
- [9] Project JXTA: *JXTA.org homepage*; <http://www.jxta.org/>, 2002
- [10] Junseok Hwang, Praveen Aravamudham, Elizabeth Liddy, Jeffrey Stanton, Ian MacInnes: *Charging Control and Transaction Accounting Mechanisms Using IRTL (Information Resource Transaction Layer) Middleware for P2P Services*; Proceedings of the ICQT 2002, Zurich, Switzerland, 2002
- [11] Burkhard Stiller, Jan Gerke, David Hausheer, Jan Mischke: *Peer-to-Peer Services Architecture*; Deliverable of the MMAPPS Project, 2002.
- [12] Huw Oliver, Alan Smith: *Technical Requirements Specification*; Deliverable of the MMAPPS Project, 2002
- [13] Rita Chen, William Yeager: *Poblano - A Distributed Trust Model for Peer-to-Peer Networks*; <http://www.jxta.org/project/www/docs/trust.pdf>, Sun Microsystems, 2002