Chapter 8
Overview of Service Oriented Architecture for Resource Management in P2P Systems

Farag Azzedin
King Fahd University of Petroleum & Minerals, Saudi Arabia

Mohamed Eltoweissy
Virginia Tech - Advanced Research, USA

Salman Ahmad Khwaja
King Fahd University of Petroleum & Minerals, Saudi Arabia

ABSTRACT

The P2P computing is one of the technologies that is having a significant impact on the way Internet-scale systems are built. It is well established for applications such as file sharing and parallel distributed computation. The well establishment for P2P systems is built on the notion of sharing. One of the objectives of resource sharing is to increase some measure of the overall work done by the collection of resources. The authors present a service oriented architecture for sharing and managing resources in P2P systems. As P2P services have shown expedient way of interaction between peers utilizing different services, our architecture will extend the high utilization of P2P networks with service orientation for handling more services with better management and easy extendibility to further capitalize on the success of P2P networks.

INTRODUCTION

Resource-sharing is one of the most popular attributes of Peer-to-Peer (P2P) systems, where peers do not need to set up and operate servers in order to provide resources used by others. Different P2P networks come to existence for different types of resources and purposes. For example, Gnutella is used for file-sharing and SETI@home is used for distributed computing. In order to efficiently utilize the sharing of resources, a Resource Management System (RMS) must be put in place. A resource management system manages the resources that comprise the computing environment (Traversat, 2002).

DOI: 10.4018/978-1-61520-686-5.ch008

Managing resources in a P2P environment is a challenging task because of the lack of centralized control and the intermittent nature of peers. Resource management in contemporary P2P system usually depends on the network overlay. Napster uses a client-server approach in handling resource discovery and uses a P2P approach in transferring the resource. On the other hand, Gnutella uses flooding to discover resources since there is no special peer to handle resource discovery. Furthermore, a FastTrack network dynamically differentiates peers into regular nodes and Supernodes. Supernodes handle most of the resource management tasks

These approaches to RMS in a P2P setting have certain limitations. The use of centralized entity such as in Napster becomes a single point of failure that can make the whole system non-functional. On the other extreme, flooding messages in Gnutella can take considerable bandwidth which results in scalability problems as the number of nodes increases (Ritter, 2001). Lv, Ratnasamy & Shenker, (2002) studied the scalability of Gnutella, where a design was proposed to: (a) restrict the query flow into each node so they do not become overloaded and (b) dynamically evolve the overlay topology so that queries flow towards the nodes that have sufficient capacity to handle them.

The support of only one resource type limits the utilization as well as the widespread of P2P applications such as Gnutella and KaZaA. Furthermore, DHT-based systems assume that the peers have uniform capabilities. This assumption, however, is difficult to be accepted in real dynamic systems (Hu, Wang & Zhu, 2005). Nodes have varying availability rate, bandwidth, storage, and computing power.

We hypothesize that the limitations of P2P systems can be addressed by a Service Oriented Architecture (SOA) approach. Peers can provide and use different services without having to know the internal workings of these services. Each peer does not need to implement all services as it can use services of other peers to perform its tasks. This is when re-use comes in to the game. A peer now is able to quickly build up another solution that will reuse a set of these services while also providing additional services that the application may require. Such services are woven in to a sort of ecosystem of software. In this scenario, they cooperate as a means of achieving some sort of business objective, while possibly participating in some other ecosystems that offer similar service capability for a possibly different end solution.

The basic idea of SOA is not entirely new. The idea is that it would be nice to create complex systems from simple separate parts. SOA can be defined as an architectural style of building reliable distributed systems that deliver functionality as services, with the additional emphasis on loose coupling between interacting services as defined by Treadwell (2005). Furthermore, SOAs stay independent of the internal workings of the services. As such, it needs to provide certain standards defining the services, their inputs and outputs. Anyone can provide the service as long as the standards for the service are met. A peer can choose the best service for its needs as the peers are loosely-coupled. This can encourage service providers to improve their quality of service in order to attract more business.

Binding SOA with a P2P network seems one of the most logical solutions, as P2P is already providing services to the millions of the networked users. As such, adding SOA on P2P systems will introduce more flexibility and more diversity. The flexibility is provided by allowing different types of service and resource providers to participate in the network. These different service and resource providers give rise to diversity of services available to all peers on the network. The recent interest in SOA was spurred by the emergence of web services (Krajicek & Kuba, 2006). Current incarnations of web services and P2P systems suffer from distinct and, in many cases complimentary drawbacks. While web services are strong on standards, they show possible weaknesses on issues such as scalability and resilience to node

Figure 1. Resource management system classification

	Centralized	Distributed
Non-service-oriented	Napster	Gnutella, FastTrack, CAN, Chord, Pasrty, Tapestry, Friese et al., JXTA
Service-oriented	-	SP2A, WSPeer, SNAP, García et al., Elenius, Hasselmeyer

failure. P2P systems on the other hand are strong on these issues, yet lack open and shared standards of service definition and message exchange. We believe that combining the strengths of both, we can create more viable, flexible and ubiquitous service deployment scenarios.

BACKGROUND

Normally P2P networks are classified according to their underlying architecture into two dimensions: centralization and service-orientation. Because service-oriented architecture is inherently decentralized, the classification results in three categories shown in Figure 1.

One of the earliest P2P file-sharing systems, Napster, utilized centralized approach in resource discovery. Connected peers submit their list of shared files to the server. Search queries are also sent to the server which resolves the queries based on the submitted lists. A peer can then commence the actual P2P communication to the peer requested the file (Miller, 2001). This approach has several advantages such as simple resource discovery process and up-to-date resource information. However, the centralized nature of Napster has technical and non-technical vulnerabilities. Successful attacks to the indexing servers (or their controlling entity) will result in complete shutdown of the system. In fact, Napster had to shut down due to legal attack to the controlling entity. Its demise paved the way to the more distributed systems such as FastTrack and Gnutella.

In a pure P2P system like Gnutella, every peer maintains information of its own resources and no special peer has the responsibility to keep track of resources of other nodes. Search requests, usually for file names, are flooded through the network with a specified TTL value. The TTL value determines the maximum distance of the flood (Foster, 2002). If the TTL is close to the network diameter, the requests would be propagated to most of the peers.

An approach using simple hierarchy is used in FastTrack network, a popular proprietary P2P network used by KaZaA P2P application. Peers having fast connections become Supernodes that handle resource discovery. Any peer using KaZaA can become a Supernode if they have a modern computer and are accessing the Internet with a broadband connection. A peer has to connect to a Supernode and submit its list of shared files. The peer will also direct its search requests to the Supernode. When they search, they send the search request to the Supernode. The download will take place between the peer on which the file is shared and the peer that requested the file, not via the Supernode.

Structured P2P networks have been proposed by various researchers. Examples of these are the Distributed Hash Table (DHT) based systems such as CAN (Francis, 2001), Chord (Balakrishnan, 2001), Pastry (Druschel, 2001), and Tapestry (Joseph, 2001). These systems are designed to be efficient and scalable in searches compared to flooding technique usually employed in unstructured P2P networks. In a structured architecture,

the set of connections between peers and resource placement are arranged to improve performance (Cao, 2002). The arrangement involves assignment of IDs to nodes and resources based on a certain hash function. The hashes are then distributed by the system automatically to certain peers based on their IDs.

(Friese, 2003) developed a framework for resource management and information storage in P2P networks. It addresses interoperability problem in P2P networks by providing an additional layer of abstraction to simplify development of new protocols and applications. Our work has similar goals but by using service-oriented approach, the tasks in the resource management layer can be handled by different entities.

Sun Microsystems started an open-source project JXTA as a platform to develop P2P applications (Traversat, 2002). It standardizes a common set of protocols for P2P virtual network and is designed to be independent of programming languages. Peers can be an edge peer or a Supernode include rendezvous and relay peers. JXTA proposes a hybrid approach combining loosely-coupled DHT with rendezvous walker.

Much work has been done in application deployment on P2P systems. One of the most recent works done in this area is WSPeer (Harrison, 2005), where programmers deploy the classes and objects on the P2P network so that others can access and utilize them in their application development. Similarly, (García, 2006) have developed SNAP, a world wide scalable applications deployment infrastructure. Using SNAP, the developers do not need to reinvent the wheel creating overlay networks and services. The framework also provides a smooth transition from web applications and services to the decentralized P2P model.

(Amoretti, 2005) presented a service-oriented framework (SP2A) to enable P2P resource sharing in Grid environments. It defines the modules that a peer should have in such environment. Examples of these modules are a scheduler, state manager, resource monitor, security manager, router, and message handler. Resources are not exposed directly but can be accessed through a pool of resource provision services.

A service discovery system for ubiquitous P2P network is presented by (Elenius, 2003). In this work, the authors show how the JXTA P2P networking infrastructure can be extended and integrated with current technologies to achieve service discovery as well as remote invocation. An analysis and categorization of the processes involved in service discovery is done by Hasselmeyer (2005). It identifies registration and lookup as the distinguishing features of the various discovery systems. Both these processes are found to be of either dynamic or static. The author surveys a number of service discovery systems and classifies them according to the process types.

Our approach is different from these approaches as we are proposing a framework over peer to peer network to implement Service Oriented Architecture on it. So by proposing a framework to expand P2P network we are stretching the limits of P2P network to include new and more diverse services in a single network. To have more options and functionality available to the P2P users and to allow the P2P network to grow not in one or two specific services but to have all sort of services. As we involve more users to provide services we are allowing limitless opportunities for the P2P network to grow. It will not only allow more services but also will allow to have more diversity and more quality as each type of service provider tries to improve his service and attract more users and consumers.

REQUIREMENTS FOR THE ARCHITECTURE

In a P2P environment, we expect resources and clients to scale up to millions of participants with no central administration nor a global node to coordinate and manage these shared entities. Such a tremendous administration and manage-

ment task is not feasible for an individual peer. Furthermore, the nature of a P2P environment namely site autonomy and intermittency, makes globalization of any sort not a useful option. To perform efficiently under these conditions, the architecture should have the following features:

- **Minimum central or global control**
- **Scalability:** This is a main feature of P2P systems. Hence, our architecture should be scalable as the number of peers increase
- **Support for intermittent participation:** The "on" and "off" joining-basis of the P2P environment should be the norm for the architecture
- **Some redundancy:** This is basically a feature of P2P systems to have redundancy. This redundancy acts as a backup and keeps the system always available. It also stops the system from going down suddenly

PROPOSED ARCHITECTURE

Figure 2 shows the important services which are required for the basic operations. These services can be present at different locations and can still interact and communicate with each other to serve for bigger purpose. Communication between peers is done by sending specification files in XML format. To keep the architecture compact, only the most important services are shown in the figure. Peers can add secondary or pluggable services through the discovery service. There is no limitation on the number of service providers.

The discovery service is responsible for accepting registration of services and matching registered services with searches. The discovery service is also responsible for associating a service provider and his services to the network and making those services available to service requesters. The quality assessment service provides various services regarding assessing different quality of service attributes. The selection service can select a service provider based on given criteria

Figure 2. Block diagram of the proposed architecture

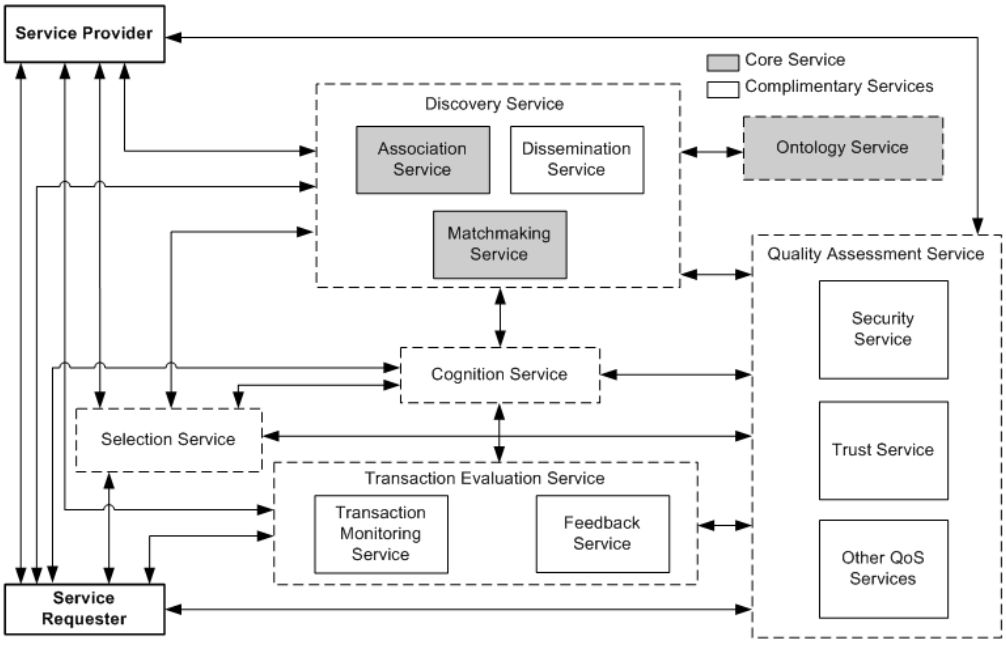
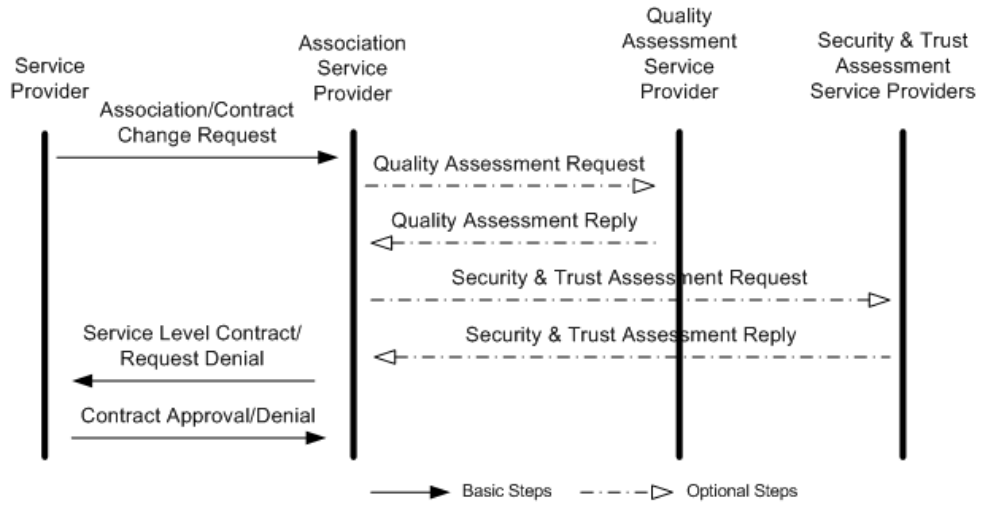


Figure 3. Service provider joining or changing the contract with an association service



provided by the user of the service from a given list of service providers. The transaction evaluation service ensures that transactions take place satisfactorily. It also enables service providers and service requesters to provide feedback about the quality of the transaction. The cognition service provides adaptive learning capabilities that can be invoked by all other services participating in P2P resource management. The ontology service is used to deal with the heterogeneity nature of P2P systems.

Discovery Service

The discovery services consist of components which are responsible for making the service providers available and known on the network. It enables service requesters to register, find, and use the services. (Hasselmeyer, 2005) showed that there are two main processes in discovery i.e. registration process and lookup process. In our approach, the discovery service is divided into three main services: (a) association service, (b) dissemination service, and (c) matchmaking service. Association service and dissemination service perform the tasks of registration process while matchmaking service handles the lookup process.

Association Service

The association service is one of the most critical services in the RMS framework. The association helps peers to make their services available to other peers. For the service provider to connect and interact on the P2P network, a service provider needs to associate itself to a peer connected to the P2P network. This peer should almost always be available and keeps a list of services, accepts association requests from service providers, and has the ability to publish services. The peer providing these facilities to the service provider is said to be providing the association service.

Association service is present on Supernodes. These Supernodes keep track of all the services provided by the peers connected to them. As such, every Supernode has a different list. These lists can overlap because an already connected peer might connect to another Supernode. In this case, two different Supernodes will have an entry for this service. This redundancy is not only acceptable but rather it is necessary to keep the system running even in case of failure of some Supernodes

There are two different scenarios for interaction between a service provider and an association service: (a) when a service provider joins the P2P

network through one of the Supernodes' association service for the first time or when the service provider wants to change the contract (Figure 3), and or when the service provider joins P2P network through some other Supernode on which it is not listed at all, (b) when a service provider rejoins the P2P network after being offline with one of the same Supernodes associated itself earlier. In scenario (a), the service provider provides details of the services it has to offer, and there is a (new or renewed) service level agreement between the service provider and the association service. In scenario (b), the service provider needs to only provide its assigned ID and get authenticated and reconnect to the P2P network through the association service.

To ensure that the peer requesting for association is valid and will contribute positively to the network, an association service on the Supernode may invoke quality assessment services to decide whether to accept the association request or not. The quality assessment service may or may not be running on the Supernode. In the case the quality assessment service is running on some other peer, the Supernode can sent a request to check the quality of the peer requesting association. Similarly, authentication and trust information can be provided by the security and trust services respectively. These services also may or may not be present on the Supernode. So, the Supernode may need to contact one or more peers providing such services.

An association service on a Supernode may only support specific type of resources or services. As such, a peer providing different services may be associated with more than one Supernode. In case of the service provider's Supernode gets a request for s specific service which does not support, it will forward this request to other Supernodes on the network. These Supernodes will reply to the service provider's Supernode with an acceptance or a denial of the contract. The response will be then returned back to the service provider by its Supernode.

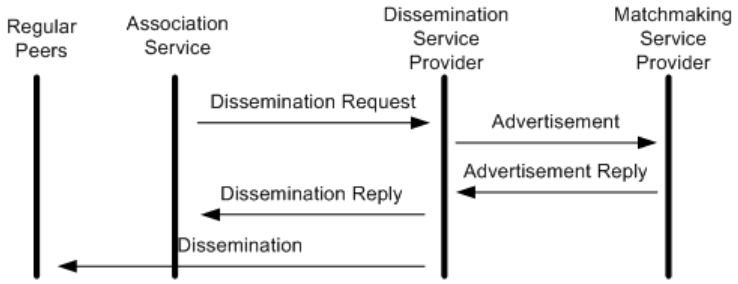
Dissemination Service

Making services known to other peers on the network is an independent task. This task is accomplished by the dissemination service. By propagating the information about the service, more and more Supernodes can avail the services of service providers.

This service will be optional for the Supernodes. Supernodes can advertise the services they provide or they can choose not to advertise. Supernodes can also choose to advertise the service providers which are registered with them. When the Supernode wants to advertise its own services, then the association service invokes the dissemination service, requesting it to advertise the association service's presence and some summary information of the services it provides. The dissemination service then advertises this information and hence the Supernode's reputation as providing many services will propagate through the P2P network. In return, the Supernode gets more requests for association of different services from other peers.

When the Supernode wants to advertise its service providers, it invokes the dissemination service which calls the matchmaking service. The matchmaking service finds information pertaining to the services associated with this particular Supernode and which have applied for advertisement. After getting the information from the matchmaking service, the dissemination service advertises this information to all the peers connected to it. These interactions are shown in Figure 4. As soon as the request from the association service of the Supernode is received by the dissemination service, the dissemination service provider forwards it to the matchmaking service provider. The matchmaking service provider retrieves the list of the services which has requested for advertisement along with the information about the services they provide. This list is then advertised by the dissemination service. This is actually one way of advertising the services in the

Figure 4. Association service invoking dissemination service to advertise to matchmaking services



network. This is the push mechanism. The dissemination service can be accomplished using other mechanisms as well such as pull or on-demand. Similarly, a dissemination service may receive requests from other dissemination services. If the dissemination service provider receives a request to advertise its services; then the advertisement is known as pull-based.

The dissemination service provider is the bridge between service requesters and service providers. The matchmaking service does not need to provide detailed information about all services available on it; it only advertises the summary of what services it has to offer. The dissemination service may take advantages of other dissemination services by sending them the request to advertise the services on its behalf. As mentioned earlier, the dissemination service is optional and a Supernode may decide not to advertise, may decide to advertise on some specific conditions, or may decide to advertise through some other dissemination service providers.

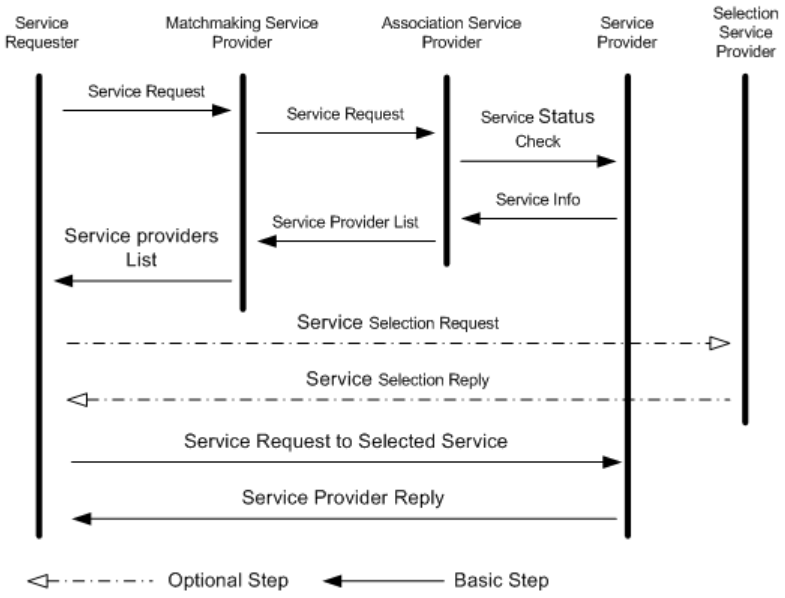
Matchmaking Service

The matchmaking service is the second most important component on a Supernode. A Supernode must have a matchmaking service available on it. This service answers to service requesters with the list of best service providers. Matchmaking service can interact with other services to give different priority levels for the association services based on trust, security, cost, experience, etc. These interactions are shown in Figure 5.

When a service requester contacts its Supernode for any service, this request is handled by a request handler which is part of the matchmaking service. First, the matchmaking service checks if the Supernode has a list of services matching the requested service. If a match is found, then the matchmaking service gets that list and sends it to the association service provider. The association service provider checks with all the peers on the lists and returns the list of active service providers. Once the matchmaking service provider receives the list, it forwards it to the service requester. The service requester may put some conditions for the service it requires; such as space limit in case of storage service. In such a case the matchmaking service will only forward the list of the services which fulfill such requirements. If the matchmaking service provider finds that the Supernode on which it is running does not have the requested information, then it will need to contact other Supernodes.

The matchmaking service needs to know the type and amount of resource required from the service. It may also need to know the duration of time for which the service is required. For example, for using computing power, the duration of time will be required, whereas for file sharing, the duration of time will be irrelevant. The service requester may also be interested in focusing the search for only such service providers that have a certain minimum trust level, and/or a certain

Figure 5. Matchmaking a request and available services



minimum bandwidth available. The geographical location of the service requester and the preferred geographical location of the service provider may be passed as inputs to the matchmaking service provider. The request may also include optimization choice such as time optimization or cost optimization. The matchmaking service, while returning a list of appropriate service providers, may identify their IDs, their locations, and other details about the services requested.

In producing the list, the matchmaking service provider may use a selection service to filter service providers based on certain quality attributes. The service requester can then initiate the operations to commit transaction with the chosen service provider.

Figure 5 shows a search request sent from a service requester through the matchmaking service provider to association service provider. The association service provider checks with the service providers for availability. After finding the available service provider, the association service provider updates the list. This updated list is propagated from the association service provider back through the matchmaking service provider to the service requester. The selection service is shown to be used by the matchmaking service provider and by the service requester as well. This is actually optional and an implementation may do the selection process at any one place as shown in the figure.

Selection Service

Once a service requester obtains a list of service providers willing to provide the service and can meet the QoS requirements specified by the service requester, the service requester must select the services that best meet its needs. The selection service is very crucial and needs to be done in an efficient and effective manner.

A selection service should select the best service provider in a timely manner as well as minimizing the reselection cycle. A reselection cycle occurs when the service requester keeps selecting service providers because none of the selected service providers successfully finish the service. On the other hand, a service selection

process should be flexible in providing static as well as dynamic-based selection service.

Current techniques for service selection depend on static attributes and do not consider dynamic assessment of QoS attributes. An essential feature of SOA is its capability of dynamically selecting components (Maximilien, 2004). Therefore, the selection service in our proposed architecture should enable service requesters to select based on dynamic assessment of service providers. Dynamically assessing a service provider's QoS attributes is challenging because it depends on how, by whom, when, and where a given service is invoked.

The selection service can be requested by the Supernode while sending the list of the services or it can be service requester after it gets the list of peer providing the requested service. The service requester can then ask s service selection provider to decide for the best service. Similarly, the service requester can request the selection service provider to help in selecting the best suitable service for it.

Ontology Service

Ontologies are used within the context of Spatial Data Infrastructures to denote formally represented knowledge that is used to improve data sharing and information retrieval (Hartmann, 2007). Web Ontologies were proposed to solve the problem of integrating and sharing heterogeneous information resources in the Web. The purpose of the ontology service is to provide a unified knowledge base.

Ontologies are commonly used for a shared means of communication between computers and between humans and computers. To reach this aim, ontologies should be represented, described, exchanged, shared and accessed based on open standards (Bejar, Lacasta, Muro-Medrano, Nogueras-Iso, Zarazaga (2006).

Ontology service is very important in our architecture as many services by many different providers will be available for use. All these services need to maintain a common knowledge. Similarly the user or consumer of these services should also have a very clear understanding about the services they want to access. They also need to know how to request for service in an unambiguous manner. Similarly service provider should also be very clear, precise and unmistakable while defining or registering their services. To attain this goal we want to have a single and unambiguous meaning for each attribute in service and the result of the service.

The ontology service is present on each Supernode, so that it is decentralized and is quickly available for each peer. Supernodes should also interact with each other to have a unified knowledge representation. If a conflict arises between two Supernodes about any context, then there is a conflict resolution service available within the ontology service to solve this problem.

Every service provider before associating its service will check with the ontology service to verify that the service knowledge is in accordance with ontology knowledge. If there is some missing attributes, then the service provider can send a request to its Supernode to update its ontology service. If there is no conflict, then the service association process can proceed. The ontology service will be based on XML standard.

Quality Assessment Service

The quality assessment service provides miscellaneous services that can be invoked from different parts of the resource management system to assess various QoS parameters of any given entity in the system. In this section, we describe security and trust services as examples. (Azzedin, 2004) showed that security and trust issues are not only important in P2P environments but they are also highly challenging issues. These services can also contribute to incentive mechanisms by prioritizing peers that provide secure and trustworthy services. The quality assessment services can be on any peer or can be part of a Supernode.

Security Service

Security issues and authentication methods are complex enough to be dealt with separately and can be encapsulated into an independent service in itself. A security service is a service that can be contacted by any service, peer, or Supernode for the sake of checking the authentication of another entity in the system.

The invoker of the security service passes information about the entity that it wants to be security checked. The security service does authentication and replies to the invoker. The invoker can then make a decision based on the security service reply.

As seen in the association service, there is a need to authenticate a service provider joining the P2P network. Whenever the association service is contacted by a service provider to join the network, it can utilize the security services to authenticate the prospective joining service provider before proceeding with the rest of the association process.

Trust Service

As P2P systems are generalized, we need to manage the resources to deliver predetermined levels of trust. The trust service will check the trustworthiness of an entity. This service will help in protecting the P2P network from malicious service providers or requesters.

One of the requirements to deliver acceptable levels of service is accountability on the entity side (Dingledine, 2001). For example, a resource should be accountable for promising services and not delivering them. One way of holding a resource accountable is to maintain a trust parameter dedicated to the resource and update it accordingly. It has been shown (Azzedin, 2006) that trust is a concern of the service requester as well as service provider. Therefore, there is a need for a trust service that can be deployed to evaluate and maintain the notion of trustworthiness.

When a service requester wants to evaluate the trust of a service provider, it contacts a peer that provides a trust service. The trust service may delegate some of its underlying tasks to other peers. For example, the trust service can invoke a reputation service to compute the reputation of an entity. The reputation service provider receives the reputation request and accordingly computes and replies with the reputation of the entity. After receiving the reputation, the trust service may pass the reputation information to an aggregation service provider. This is done depending on the preference of the service requester and how the aggregation process should be done. The aggregation service performs aggregation of the reputation information and returns the aggregated trust value to the trust service. The trust service returns the aggregated trust value of the entity to the service requester. It should be noted that the aggregation process can be done by the service requester if it has this functionality available.

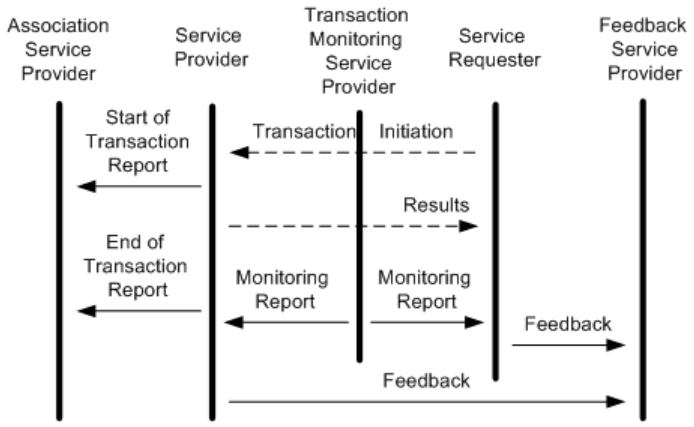
Transaction Evaluation Service

The transaction evaluation service provides the services required during and after a transaction has taken place between peers. These services help the system in providing more reliable and better resource services by identifying and spreading information about the good or bad conduct of any peer in the system. The transaction evaluation service mainly consists of two services, namely Transaction Monitoring Service and Feedback Service. Their involvement in a transaction is shown in Figure 6.

Transaction Monitoring Service

During a transaction, the service requester and the service provider may want to have the transaction monitored. The monitoring service might run on any peer in the network. Service providers at the time of associating themselves with Supernodes may specify whether their services can be moni-

Figure 6. Transaction evaluation service



tored while providing the services. Similarly, a service requester may request for service providers who agree to the option of monitoring. As such, the matchmaking service will only return the list of those services which agree to the option of getting monitored.

So, the transaction monitoring service is highly flexible and independent in the architecture. The transaction monitoring service will work as a proxy between service provider and service requester shown as dashed lines in Figure 6. The transaction monitor service may provide the monitoring results to both parties after the transaction is done. The two interacting peers can then use these monitoring results to update their records and use the result to make decisions for future transactions.

Feedback Service

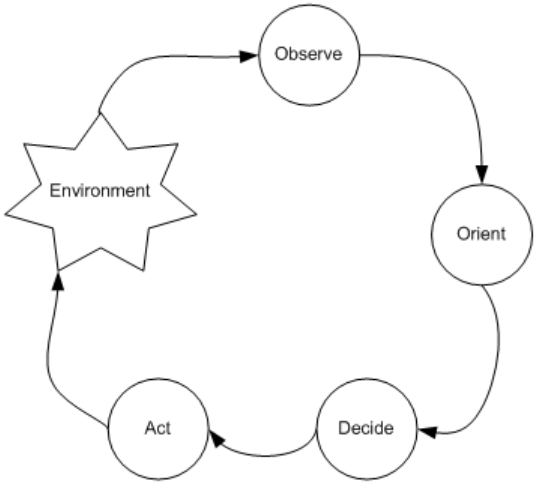
The feedback service is also provided by any peer. After performing a transaction, an entity can give a feedback about another entity that it transacted with. This feedback is then stored along with the transaction result and can be used to deduce the behavior of the service provider or the service requester. The feedback service accumulates feedbacks and manages the difficult task of filtering genuine complaints and appraisals from malicious ones. As this feedback is stored along with the transaction result therefore their authentication and genuineness can be done more easily. In a nutshell, the feedback service can provide useful information about the peers in a form of appraisals or complaints. In our proposed architecture, the feedback service can be provided as a standalone service or as one component of a bigger service.

Cognition Service

The cognition service provides adaptive learning capabilities that can be invoked by all other services participating in P2P resource management. The cognition service may employ the cognitive cycle shown in Figure 7.

The introduction of cognition is motivated by the vital need for the evolution of the architecture in response to emergent behavior in the highly dynamic P2P networks. The cognition service is fed by knowledge and information from the other services. Accordingly, it builds and disseminates network Situation Awareness (SA) which encompasses the network past, present and projected situation. The SA comprises rules, goals, patterns of behavior, resource and use profiles, sources and probabilities of failures, and others. The SA is consulted by other services in order to enhance the decision-making process.

Figure 7. Cognitive cycle (Adapted from Mitola, 1999)



By incorporating cognition, each service learns about previous decisions in the network and the behavior patterns that may lead to enhancing future decisions accordingly. In other words, the services evolve by learning from past experience to enhance resource management operations over time. For example, the quality assessment service invokes the cognition service to evaluate and learn from pervious assessment decisions. This could lead to change in the assessment techniques, assessment rules or refinement of knowledge used by the assessment service. Similarly, cognition within the selection service enhances the future selection decision by evaluating past ones. While the transaction evaluation service identifies good or bad conduct of a peer in the system, as discussed before, a cognitive transaction evaluation is able to recognize new patterns of malicious or selfish behavior and also predicting such behavior before it affects the service quality.

The service provider and requester also benefit from the cognition service. For example, the service specification provided by a service requester may be too restrictive or too general which decreases the matchmaking service quality. As the service requester evolves, it learns from previous service invocation scenarios and adapts its service requests and specifications to realize maximum QoS. The discovery service invokes the cognition service to enhance its association, dissemination, and matchmaking strategies based on past experience. Similarly, the transaction evaluation service is refined based on learning. Evaluation techniques and metrics can be enhanced or new ones introduced.

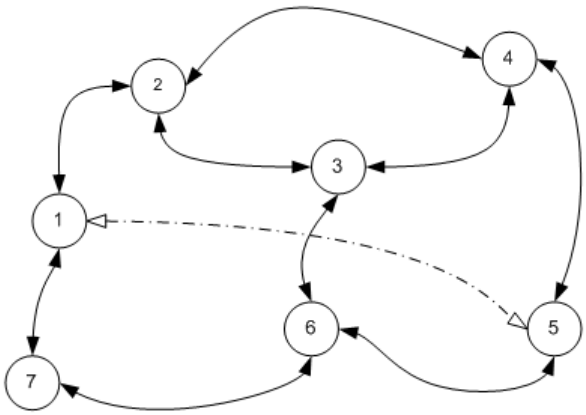
The quality of the cognition service depends on the amount of information and knowledge available to build the SA. However, learning provides tools for decision-making under uncertainty to compensate the global SA deficiency.

PREFERRED P2P NETWORK

Overview

The SOA is highly flexible in providing different services and functionalities in a standardized way. It is designed to work in highly varied and heterogeneous environments. This inherent flexibility in SOA allows us to design an architecture for all types of P2P networks. These network topologies include centralized networks such as Napster, decentralized networks such as Gnutella and FreeNet, and hybrid networks such as KaZaA.

Figure 8. The architecture with a decentralized P2P network



SOA for resource management can be applied on all three types of P2P overlay networks. In this section, we compare the architecture of these three different P2P overlay networks and propose one that is more suitable for our proposed architecture.

A Centralized P2P Network

In the centralized P2P overlay network, the architecture (shown in Figure 2) can be implemented by having all the services provided by the service providers registered with one central server which is also handling association services.

Since peers are connected to a central server, a service provider needs to send a request to this central server. Every request from a service provider will reach the discovery service of the central server.

A service requester will also send a request to the central server. The server looks into its index and finds all relevant service providers for that particular service. The server will also make a match between the service requester and the service providers.

The advantage of such a centralized approach is that all the data is collected in a single place, so there is no redundancy. But the main disadvantage is the single point of failure. As such, the centralized P2P network does not fit very well with our defined requirements.

The core services for the central server are the association service and the matchmaking service. Other complementary services can be part of the centralized server, but having more services will adversely effect the performance of the central server as it will put more load on it and hence will hinder its main function of association and matchmaking.

A Decentralized P2P Network

Figure 8 shows the architecture on a decentralized P2P network. Each peer on the network has a small server part activated on it. A service requester will send a message on the network requesting a specific service. If the message reached a service provider providing that specific service, then the service provider will reply to the service requester. After getting all the replies, the service requester will choose a service provider or it might activate a selection service to decide from which service provider to get the service. The connection between the service requester and service provider is shown a dashed line in Figure 8.

There are two ways for a peer to get registered and offer it services. If the peer is new on the network then it can send information about its service

by flooding the network. Some of the peers which receive this message with information will include this peer in their index. This inclusion for the specified service will depend upon some criteria like how many such service providers are already in the list and what is the trust and security level of this peer. The criteria for adding a peer in the list can be a little customizable for each peer. Second way is when a service providing peer receives a request for that service, then this peer can send information about its service. Similarly if other peers who have taken service from this peer also get a request for this service then they can also send information about this peer service to some third peer requesting for this specific service. In this way the service providing peer is listed in more and more peer for the service it provides.

The advantage in this type of peer-to-peer network is that it is totally decentralized, but it also increases redundancy to an unacceptable level. There is also huge increase in bandwidth requirement due to flooding for requesting services and sending information about services. This architecture also require each client to have some basic and necessary server portion on it, so it will put extra burden on thin clients.

The core services required to be running on every peer is: (a) Matchmaking service. The recommended complimentary services which can be running on every peer is: (a) Association service. Here it should be mentioned that the match making service can vary greatly in each peer. If a peer is very thin and is not providing any service then the match making service can only consist of the module which is just receiving the service request and forward it to the peers it is connected with, or just replying an empty list to the peer or even it can configure not to take any action on the request. This service is required so that every service from other peer can be handled in a proper way.

A Hybrid P2P Network

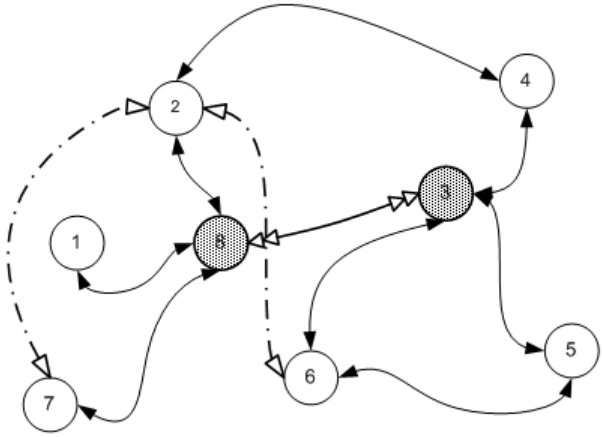
In the hybrid peer-to-peer network approach there are Supernodes, which keep track of regular peers. Each peer when comes on the network is attached to at least one Supernode. In this hybrid approach, the architecture can be implemented by having some very basic services running on the Supernodes. These services include discovery service and match making service.

When some peer want to provide some service it will send its information to the Supernode it is connected to. The Supernode will add this information in its list. Supernodes are connected together also, and these Supernodes can exchange information among themselves also about different services their peer provides. Supernodes are shaded in the Figure 9.

If a peer request for a service, it will send the request to its Supernode, the Supernode will check its list and it will return the relevant peers to the requesting peer. If the Supernode does not have any relevant peer providing a service its looking for then it will forward the request to other Supernode These peers will check in their respective list and send a reply. The request initiating Supernode will compile the result and will forward the request to the peer who has asked for the service. The requesting peer will establish a service connection with one of the service provider from the list which is shown by a dashed line in the Figure 9.

The advantage in this approach is that it is decentralized and is also highly extendible. The peers are grouped in smaller cluster which are managed easily and if one Supernode goes down other Supernode can take its place and the peer connected to the downed Supernode can connect with some other Supernodes. This approach does add extra traffic of the communication between Supernodes but the scalability, flexibility, and performance attained is worth much more. The network is highly decentralized and the benefits gained from this hybrid network approach make it a better approach for our architecture.

Figure 9. The architecture with hybrid peer to peer network



The core services for Supernodes are: (a) association service (b) matchmaking service and (c) ontology service. The recommended complimentary services for Supernodes are: (a) dissemination service, (b) trust Assessment service, (c) security assessment service and (d) selection service. If these services are not available on the Supernode, then the Supernode can contact its regular peers who are providing these services.

Remarks

As discussed, a hybrid P2P network provides better set of characteristics required by our proposed architecture. It provides high level of scalability, as the number of regular peers increases the number of Supernodes can also increases to better balance the load of services. It should be noted that although management issues go through Supernodes, the actual transactions are done between regular peers.

Moreover if few Supernodes go down, the system can continue working without any disruption of services and the load can be transferred to other Supernodes. This provides uninterrupted services and better scalability at the expense of very little overhead of extra traffic on the network, which is introduced due to the communication between Supernodes. But this extra traffic is very little in comparison to flooding the P2P network.

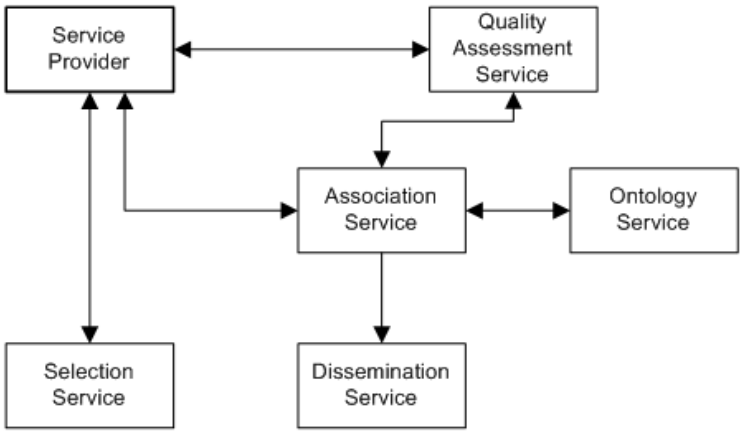
ILLUSTRATION OF THE SYSTEM

Scenario for Providing a Service

Figure 10 is an illustration of how the architecture described in previous sections can be used to provide a service in a P2P network. A service provider interested in providing its services, may utilizes the quality assessment service to assess the quality of a given association service provider. If the service provider is satisfied with the quality of the association service provider, it will associate itself with this particular association service provider. Similarly, the association service provider might be concerned with the quality of the service provider and can approach the quality assessment service as well.

Furthermore, the association service can use the dissemination service to advertise its registered services matchmaking service providers. It should be noted that the scenario shown in Figure 10 is only one way in which a system could provide a service, and that the SOA described in the previous sections can be used in several different ways to provide services.

Figure 10. Flow chart showing the procedure for providing a service

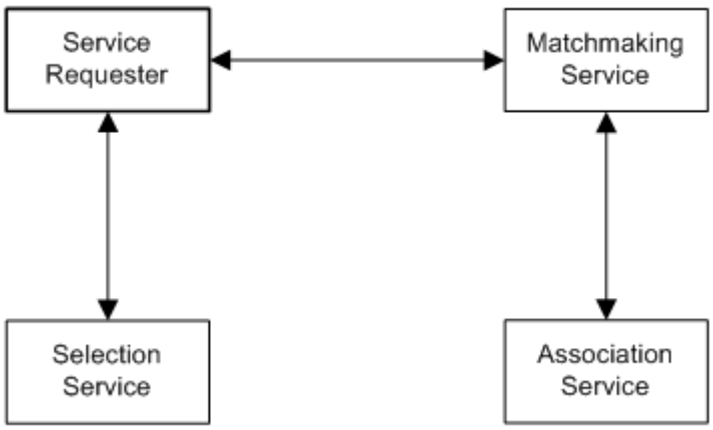


Scenario for Requesting a Service

Figure 11 shows a service requester initiating a service request process. The service requester contacts a matchmaking service with its request. The service requester may be informed of a service request failure in several circumstances: (a) if the matchmaking service did not have information about any appropriate association service providers and also the Supernode it is connected with also does not contain the list for the specific service, and (b) if all of the appropriate service providers are not currently available to provide the service requested.

If the matchmaking service provides has a list of service providers that can satisfy the service request, the matchmaking service checks with the association service to find if the service providers are currently available to provide the service. The association service sends the list of all currently available service providers to the matchmaking service. This list of service providers is then sent back by the matchmaking service provider to the service requester.

Figure 11. Flow chart showing the procedure for requesting a service



BOOTSTRAPPING THE SYSTEM

Services must somehow communicate some information before they can be consumed. Some of the information includes: (a) means to announce the availability of services, (b) knowledge about the services' existence, (b) details of how to find and access the services, (c) and details of how to bind to the services.

The system will start by having at least one Supernode. This default Supernode can be set as the starting point for all the peers. When a peer wants to join the network, it will search for the addresses of the Supernode(s) it was associated with. If it finds none, then it will connect to the default Supernode which is provided by the application. That is, the address of the default Supernode is hard-coded in the application.

After connecting to the Supernode, the peer can become a Supernode, a service provider, or a service requester. When a peer connects to a Supernode, then the address of the Supernode is added in the peer's list of Supernodes. As such, if this peer wants to rejoin the network, it will check its list of Supernodes for availability and connection.

With new peers that are unknown to the system, there is always a chance that a rogue service provider wants to take advantage of an unwitting peer by pretending to offer assistance for malicious hidden motives. Being part of the system, a Supernode will not know much about a new peer that has not provided any service. So the question becomes how to check for trust, security and credibility of these new peers?

If the system uses an optimistic stranger policy, new peers would always be considered as non-malicious to give them a chance to be part of the system. If the system uses a pessimistic stranger policy, high security measures can be enforced and these security measures can be relaxed as the new peer proves itself to be non-malicious.

Another solution will be to assume that a new peer knows a set of trusted peers based on off-line relationships. The new peer can initially interact with its trusted peers and when this new peer wants to associate with a Supernode, recommendations can be collected from its trusted peers.

CONCLUSION

We have presented a service-oriented architecture for resource management in P2P systems. We believe that we can maximize the benefits of service-oriented architecture by representing the tasks of resource management as services.

There are three core services in our architecture: (a) the association service helps service providers making their service available to the rest of the P2P community, (b) the matchmaking service that enables a Supernode to find the list of the best matching service providers, and (c) the ontology used to provide a means of communication between computers and humans based on open standards.

Figure 12. Block diagram of trust model

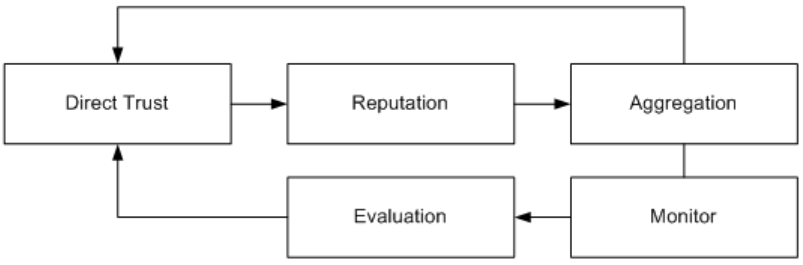
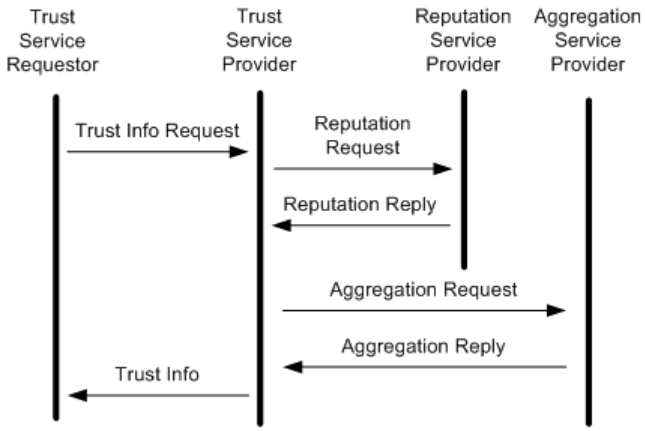


Figure 13. Trust service



Some of the services in our architecture are complex enough to be decomposed further into smaller services. For example, trust computation which comes under quality assessment service is a complex task. It involves different tasks such as computation of first hand information (direct trust) and second hand information (reputation), aggregation, behavior monitoring and evaluation (see Figure 12).

The service-oriented trust modeling and the various subtasks of trust computation can be performed by different entities acting as services as shown in Figure 13.

REFERENCES

Amoretti, M., Zanichelli, F., & Conte, G. (2005). SP2A: A service-oriented framework for P2P based Grids. In *ACM International Conference Proceeding Series: Vol. 117. 3rd international workshop on Middleware for grid computing* (pp. 1-6).

Azzedin, F., & Maheswaran, M. (2003). Trust modeling for peer-to-peer based computing systems. In *IPDPS: Year of Publication 2003. International Parallel and Distributed Processing Symposium* (pp. 99.1). Washington, DC: IEEE Computer Society

Azzedin, F., Maheswaran, M., & Mitra, A. (2006). Applying a trust brokering system to resource matchmaking in public-resource grids. *Journal of Grid Computing*, *4*, 247–263. doi:10.1007/s10723-006-9041-9

Bayardo, R. J., Crainiceanu, A., & Agrawal, R. (2003). Peer-to-Peer sharing of Web applications. In *Proceedings of WWW2003*, Budapest, Hungary.

Deriaz, M. (2005). *A thesis proposition: Service Oriented Computing in a P2P Architecture*. Retrieved July 13, 2008, from http://cui.unige.ch/~deriazm/publications/ThesisPropositionSOC.pdf

Dingledine, R., Freedman, M. J., & Molnar, D. (2001). Accountability. In *Peer-to-Peer: Harnessing the Power of Disruptive Technologies* (pp. 217-340). Sebastopol, CA: O'Reilly & Associates.

Elenius, D. (2003). *Service discovery in peer-to-peer networks*. Unpublished master's thesis, Inköping Institute of Technology, Sweden.

Friese, T., Freisleben, B., Rusitschka, S., & Southall, A. (2003). A framework for resource management in peer-to-peer networks. In *Revised Papers from the International Conference NetObjectDays on Objects, Components, Architectures, Services, and Applications for a Networked World* (LNCS 2591, pp. 4-21). Berlin, Germany: Springer.

Harrison, A., & Taylor, I. J. (2005). *Dynamic Web service deployment using WSPeer*. Retrieved July 7, 2008, from http://www.mardigrasconference.org/conf_2005/2005/Presentations/Harrison.pdf

Hartmann, J., Palma, R., & Sure, Y. (2005). *OMV–Ontology Metadata Vocabulary*

Hasselmeyer, P. (2005). On service discovery process types. In *Service-Oriented Computing - ICSOC 2005* (LNCS 3826, pp. 144-157). Berlin: Springer.

Krauter, K., Buyya, R., & Maheswaran, M. (2002). A taxonomy and survey of grid resource management systems. *Software, Practice & Experience*, *32*, 135–164. doi:10.1002/spe.432

Kuba, M., & Krajicek, O. (2006) *Literature search on SOA, Web Services, OGSA and WSRF*. Institute of Computer Science, Masaryk University. Botanická, Brno.

Lacasta, J., Nogueras-Iso, J., Bejar, R., Muro-Medrano, P. R., & Zarazaga-Soria, F. J. (2006). A Web ontology service to facilitate interoperability within a spatial data infrastructure: Applicability to discovery. *Data & Knowledge Engineering*, *63*(3).

Lv, Q., Cao, P., Cohen, E., Li, K., & Shenker, S. (2002). Search and replication in unstructured peer-to-peer networks. In *Proceedings of the 16th ACM International Conference: Year of Publlication 2002. International Conference on Supercomputing* (pp. 84-95). New York: ACM.

Lv, Q., Ratnasamy, S., & Shenker, S. (2002). Can heterogeneity make Gnutella scalable? In *The 1st International Workshop on Peer-to-Peer Systems: Vol. 2429. Revised Papers from the First International Workshop on Peer-to-Peer Systems (IPTPS)* (pp. 94-103). London: Springer-Verlag

Maximilien, E. M., & Singh, M. P. (2004). A framework and ontology for dynamic Web services selection. *IEEE Internet Computing*, *8*, 84–93. doi:10.1109/MIC.2004.27

Miller, M. (2001). *Discovering P2P*. Alameda, CA: Sybex Inc.

Mitola, J. (1999). Cognitive Radio for Flexible mobile Multimedia Communications. In *Mobile Multimedia Communications, 1999. (MoMuC '99). IEEE International Workshop on Multimedia Communications* (pp. 3-10).

Mondéjar, R., García, P., Pairot, C., & Skarmeta, A. F. G. (2006). Enabling Wide-Area Service Oriented Architecture through the p2pWeb. In *Proceedings of the 15th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE'06)*. Washington, DC: IEEE Computer Society.

Pairot, C., García, P., Gómez, A. F., & Skarmeta, S. (2004). Dermi: A New Distributed Hash Table-based Middleware Framework. *IEEE Internet Computing*, *8*, 74–84.

Paolucci, M., & Sycara, K. (2003). Autonomous Semantic Web services. *Internet Computing*, *7*, 34–41. doi:10.1109/MIC.2003.1232516

Pierre, G., & Steen, M. V. (2006). Globule: a collaborative content delivery network. *IEEE Communications Magazine*, *44*(8), 127–133. doi:10.1109/MCOM.2006.1678120

Ratnasamy, S., Francis, P., Handley, M., Karp, R., & Shenker, S. (2001). A Scalable Content-addressable Network. In *Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications. Applications, Technologies, Architectures, and Protocols for Computer Communication* (pp. 161-172).

Ripeanu, M., Foster, I., & Iamnitchi, A. (2002). Mapping the Gnutella Network: Properties of Large-scale Peer-to-peer Systems and Implications for System Design. *Internet Computing*, *6*, 50–57.

Ritter, J. (2001). *Why Gnutella Can't Scale. No, Really*. Retrieved July 17, 2008, from http://www.darkridge.com/ ~jpr5/doc/gnutella.html

Rodriguez, P., Tan, S., & Gkantsidis, C. (2006). On the Feasibility of Commercial, Legal P2P Content Distribution. In *ACM SIGCOMM Computer* [New York: ACM.]. *Communication Review*, *36*, 75–78.

Rowstron, A., & Druschel, P. (2001). *Pastry: Scalable, Distributed Object Location and Routing for Large-scale Peer-to-peer Systems*. Paper Presented at Proceedings of the 18th IFIP/ACM International Conference on Distributed Systems Platforms (Middleware 2001), Heidelberg, Germany.

Sharman Networks. (2005). *How Peer-To-Peer (P2P) and Kazaa Software Works*. Retrieved June 29, 2008, from http://www.kazaa.com/us/help/new_p2p.htm

Stoica, I., Morris, R., Karger, D., Kaashoek, F., & Balakrishnan, H. (2001). Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications. In *Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications* (pp. 149-160). New York: ACM.

Traversat, B. (2002). Project JXTA 2.0 Super-Peer Virtual Network. *Project JXTA, Sun Microsystems, Inc*. Retrieved July 8, 2008, from http://www.jxta.org/project/www/docs/JXTA2.0protocols1.pdf

Treadwell, J. (2005). *Open grid services architecture—Glossary of terms*. Global Grid Forum.

Wang, H., Zhu, Y., & Hu, Y. (2005). To Unify Structured and Unstructured P2P Systems. In *Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium: Vol. 1* (pp. 104.1). Washington, DC: IEEE Computer Society.

Zhao, B. Y., Kubiatowicz, J. D., & Joseph, A. D. (2001). *Tapestry: An Infrastructure for Fault-tolerant Wide-area Location and Routing* (Tech. Rep. UCB/CSD-01-1141). Berkley, CA: Univesity of California, Berkeley.

ADDITIONAL READING

Arsanjani, A. (2000, January). *Principles of advanced software engineering: Variation oriented analysis, design and implementation*. Retrieved May 5, 2009, from http://cs.mum.edu/cs525/Refs/VOD-12-27-99.PDF

Arsanjani, A. (2004, November 9). *Service-oriented modeling and architecture*. Retrieved May 5, 2009, from http://ibm.com/developerworks/library/ws-soa-design1

Arsanjani, A. (2005, December 2). *Towards a pattern language for service-oriented architecture and integration, Part 2: service composition*. Retrieved May 5, 2009, from http://www-128.ibm.com/developerworks/webservices/library/ws-soa-soi2

Ciampa, M. (2008). *Security+ Guide to network security fundamentals*. Florence, KY: Course Technology.

Druschel, P., Kaashoek, F., & Rowstron, A. (2002). *Peer-to-peer systems*. Berlin: Springer.

Erl, T. (2004, April 26). *Service-oriented architecture: A field guide to integrating XML and Web services*. Upper Saddle River, NJ: Prentice Hall PTR.

Erl, T. (2005, August 12). *Service-oriented architecture (SOA): Concepts, technology, and design*. Upper Saddle River, NJ: Prentice Hall.

Erl, T. (2007, July 28). *SOA principles of service design*. Upper Saddle River, NJ: Prentice Hall.

Gradecki, J. D. (2002). *Mastering Jxta*. New York: Wiley.

Moore, D., & Hebeler, J. (2001). *Peer-to-peer*. New York: McGraw-Hill Companies.

Papadopouli, M., & Schulzrinne, H. (2008). *Peer-to-peer computing for mobile networks*. Berlin: Springer.

KEY TERMS AND DEFINITIONS

Association Service: Similar to: Peergroup; Associated in the manuscript with: "service oriented architecture" and "Peer-to-Peer network"

Cognition Service: Also known as: "adaptive learning"; Associated in the manuscript with: "service oriented architecture" and "Peer-to-Peer network"

Discovery Service: Similar to: UDDI; Associated in the manuscript with: "service oriented architecture" and "Peer-to-Peer network"

Dissemination Service: Also known as: n/a; Similar to: n/a; Associated in the manuscript with: "service oriented architecture" and "Peer-to-Peer network"

Feedback Service: Associated in the manuscript with: "service oriented architecture" and "Peer-to-Peer network"

Matchmaking Service: Associated in the manuscript with: "service oriented architecture" and "Peer-to-Peer network"

Ontology Service: Also known as: "web ontology"; Associated in the manuscript with: "service oriented architecture" and "Peer-to-Peer network"

Peer-to-Peer: Also known as: "p2p" and "Peer-to-Peer network" and "p2p network" and "Peer-to-Peer system" and "p2p system"; Similar to: "distributed computing" "networked computers"; Associated in the manuscript with: "computer networks" and "networked systems"

Security Service: Similar to: Authentication Service; Associated in the manuscript with: "service oriented architecture" and "Peer-to-Peer network"

Selection Service: Associated in the manuscript with: "service oriented architecture" and "Peer-to-Peer network"

Service Oriented Architecture: Also known as: "SOA" and "service oriented"; Similar to: "web services"; Associated in the manuscript with: "distributed computer system" and "networked systems"

Transaction Monitory Service: Also known as: "monitoring service"; Associated in the manuscript with: "service oriented architecture" and "Peer-to-Peer network"

Trust Service: Similar to: Credibility Service; Associated in the manuscript with: "service oriented architecture" and "Peer-to-Peer network"