30. A Peer-to-Peer Framework for Electronic Markets

Michael Conrad, Jochen Dinger, Hannes Hartenstein, Marcus Schöller, Martina Zitterbart (ITM, University of Karlsruhe (TH))
Daniel Rolli (IWM, University of Karlsruhe (TH))

30.1 Markets as Peer-to-Peer Systems

Markets — in their ideal form — naturally represent Peer-to-Peer (P2P) systems: market participants can be both client *and* server when exchanging offers, general messages, or goods. They can directly address each other, and interact in a decentralized and autonomous fashion. Most market implementations in history, however, were far from this ideal form.

As the construction and application of electronic markets began with the emerging Internet, the prevailing paradigm for network communication was client-server. To find a coherent implementation, market designers at that time fell back on this most self-evident network topology. The design of electronic marketplaces is still influenced by proven traditional implementations. Stock exchanges, for example, that were among the driving forces for electronic markets are one such archetype. They classically fulfill the role of an intermediary by reducing the number of communication links between participants and providing a common contact point for aggregated market information. So, by acting as client-server systems they reduce transaction costs significantly.

And still, at their very core, some non-electronic stock exchanges show — and always have shown — the Peer-to-Peer paradigm, as on the physical trading floor traders (peers) interact directly when exchanging stocks. They exhibit this trade paradigm that, in the course of the development of centralized market software systems, has been losing ground.

With the liberalization of markets, the formerly 'dependent' market participants are now regaining autonomy in their market decisions. Harmonization of markets unifies different groups of participants and thereby increases the number of those that can interact with each other. This constitutes the main theme of this chapter: when a market *per se* is a Peer-to-Peer system and participants regain autonomy, how can we design an appropriate architecture for electronic markets that comprehensively implements the Peer-to-Peer paradigm? While investigating this question, we particularly focus on the support of spontaneity and interoperability by means of self-organization in liberalized and harmonized markets.

R. Steinmetz and K. Wehrle (Eds.): P2P Systems and Applications, LNCS 3485, pp. 509-525, 2005.

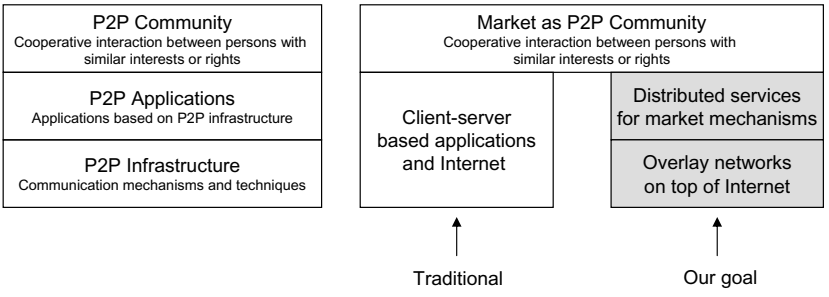

Fig. 30.1: The Peer-to-Peer paradigm at various levels of a system: the traditional architectures as well as our view for electronic markets.

To systematically analyze Peer-to-Peer systems, the Peer-to-Peer paradigm can be seen at three different levels, according to [545]: community, application, and infrastructure. As mentioned above, a market can be regarded as a Peer-to-Peer community at the top level. The remaining challenge is whether, and if so, how the application and infrastructure levels can be developed in compliance with the Peer-to-Peer paradigm. After presenting our approach, we also discuss its benefits and drawbacks.

30.1.1 Service and Distribution Basics

We propose a Peer-to-Peer architecture for electronic markets where all three levels are based on the Peer-to-Peer paradigm (Figure 30.1). A market participant is a peer not only by taking part in the marketplace and its mechanisms, but also by contributing to these mechanisms. For example, peer nodes may provide storage for storing intentions (statements of offers) and processing power for discovery requests. To design such an integrated Peer-to-Peer concept for electronic markets, the structure of the marketplace and the market mechanisms is translated into a distributed service-oriented approach.

Service orientation seems to be a suitable approach because of the clear functional separation. The functional separation facilitates extensions of market mechanisms and reusing existing ones. In our view services can be provided by a single peer or by a set of peers as outlined in Section 30.2.2. For different applications the common services can be reused with additional services building on them. In the following, we call these common services ‘basic services’. The ones tailored to our market application are called ‘application services’.

As basic services we have identified:

- document service that acts as repository for intentions
- authentication service to authenticate users and intentions
- protocol service that records the various steps of transactions

Examples of application services are:

- optimization service that determines the best offer (according to some specified criteria) out of a set of intentions
- legal mediator that checks whether a potential contract adheres to the legal policies of the market participant.

The various basic and application services (application level) have to be matched with appropriate overlay network techniques (infrastructure level) for providing the required 'quality of service'. Quality of service requirements in this context can be characterized, for example, by robustness of a service, search efficiency or accuracy, and depend on the corresponding service.

It is thus not only the fact that on the community level a market represents a Peer-to-Peer system that commends a Peer-to-Peer-implementation of the application and infrastructure level. For us, the rationale is as follows: a perfect market would offer total information to every participant and would show no communication or exchange delays, i.e. no transaction costs. However, it is quite obvious that any centralized system will impose a lack of scalability and flexibility, barriers to the market, e.g., via transaction costs and a single point of failure. Compared to centralized systems, Peer-to-Peer provides substantial advantages with the main general benefit of scalability. The seamless integration of nodes to the network enables, firstly, the adequate increase of storage or computation power, secondly, the addition of new market mechanisms on the application level, and thirdly, the integration of whole market segments in the course of harmonization. It also allows for the combination of already existing but idly distributed resources to cooperate in a Peer-to-Peer system, thereby immensely reducing the investment costs compared to setting up new resources. A distributed and self-organizing system can also reduce market barriers and transaction costs, as well as encourage spontaneity with respect to market participation. In addition, such an integrated Peer-to-Peer design can improve robustness.

Clearly, the benefits are matched against various challenges of a distributed design. First of all, there is the question of how to find intentions and services. Therefore one needs search methods as well as standardized ways to express intentions. Thus, ontologies come into play. Secondly, secure and reliable operation of market transactions has to be ensured without a centralized trusted third party.

30.1.2 SESAM Project Structure

The proposed architecture has been developed within the framework of the SESAM project of the priority research program 'Internet Economy' funded by the German Ministry of Education and Research (BMBF). The complete project covers three scenarios: multi-utility markets, virtual power plants, and wearable services. The latter two scenarios are of central interest regarding the Peer-to-Peer paradigm. In the virtual power plant scenario, we assume that many small devices producing electricity are deployed at locations such as houses, small companies, and public buildings. The owners of these power plants want to maximize their profit by selling the energy to the bidder with the highest offer. The purchaser on the other hand wants to buy energy as cheap as possible. From this starting point many interesting questions arise: How does a purchaser find the cheapest offer? Can the purchaser and seller enter the contract without personal adhesion? How are mini power plants controlled to maximize profit? What to do if a mini power plant breaks down? How is accounting accomplished?

These questions lead to several subprojects:

Electronic Contracting – Business processes are subject to legal rules. In order to achieve transparency and seamlessness with high spontaneity in the markets, the harmonization of the law must be promoted and signing of contracts must be automated in the network.

Spontaneity, Transparency and Incentives. – The considered scenarios require smooth and comprehensible interaction of various connected components and services. Transparency and incentives for the actors involved are preconditions for the functioning of such self-organizing markets.

Optimization, Control and Business Models – Due to their inherent decentralized nature and dynamics, self-organizing and spontaneous markets require specially adapted business models as well as completely new decentralized optimization and control mechanisms.

Robustness and Security – One important requirement for the commercial success of the applications is the security and robustness of all participating components and processes against active and passive breakdowns and attacks of 'normal' activity.

The results of the virtual power plant scenario get carried onto the field of so-called wearable services. New markets could emerge through communication among small devices like PDAs, mobile phones, and sensors in clothes or worn directly on the body.

To build an integrated Peer-to-Peer system for electronic markets as outlined above, an architecture is needed that brings together service orientation and overlay networks. In particular, the provision of standardized interfaces to a pool of overlay networking techniques is needed. In this chapter we will describe our service-oriented Peer-to-Peer architecture (Section 30.2) and dis-

cuss various architectural considerations concerning security/dependability issues (Section 30.3).

30.2 A Service-Oriented Peer-to-Peer Architecture

As mentioned in the previous section, markets and market mechanisms should be modeled using services, where a service represents a functional unit, and in which services can be loosely coupled through service composition. Examples are a document service for storing intentions or an optimization service that analyzes various offers and determines an optimal one. A service-oriented approach helps to easily create new applications based on already available services, thus, reuse, extensibility, and spontaneity are facilitated.

Since we argued that services should be distributed, in order to efficiently use idle resources (e.g., to reduce transaction costs or market barriers), a service will not in general reside on a single server but will be supported by peers of an overlay network. Therefore, we introduce the notion of a *ServiceNet* as an overlay network that provides a specific service. Clearly, service discovery is a required first service to locate other services or, entry points to services. Services themselves will be operated by collaboration between peer nodes as it is the case for the document service.

The document service represents a decentralized document pool where participants can insert and search for documents. In our scenarios this service is used, e.g., for offering electricity. This service provides three basic functions: *insert*, *search* and, *revoke*. The *search* functions can be parameterized using a query language. This language allows expressing a keyword-based search. Revoking documents would be the same as deleting them in the event that we could guarantee the deletion. E.g., in unstructured Peer-to-Peer networks where the document locations are 'unknown' the deletion can not be guaranteed. With this document service, organized as a ServiceNet, each participant provides some resources for storage of documents and for processing of queries or *revoke* operations.

When a set of intentions matching a query is found, the market participant (or his agent) might submit this set to an optimization service. This service estimates the best offer out of the given set based on the energy consumption profile of the participant and prices implied in the offers. Here, the user can benefit from a distributed service discovery mechanism (to find an appropriate optimization service) but also from the Peer-to-Peer approach in case the optimization is shared by various peers.

The document and optimization service were selected to serve as representative examples. Other services could focus on contracting and matching. But even with the two examples mentioned above, it can be seen that ServiceNets for various services will require different forms of Peer-to-Peer network organization. For example, the paper [52] outlines that keyword-based queries

are not trivially carried out by a Distributed Hash Table (DHT) based Peer-to-Peer network. Therefore, we need an architecture where service instances on each peer easily can be attached to a specific Peer-to-Peer networking approach, i.e., we need an 'overlay API' such as the one for DHT-based methods [148].

Following the above reasoning, we will now first make a quick digression to traditional service-oriented architecture. Then, we will discuss the concept of a ServiceNet in more detail and outline the peer node architecture that brings together service orientation and overlay networks.

30.2.1 Service Orientation

Service-oriented architectures (SOA) utilize services as basic elements to realize applications [473]. Services are characterized by:

- Service Description. This description includes, in particular, the functionality that is provided by a service. All functionality of a service is provided through interfaces. A well-known interface description language is necessary for such interface descriptions.
- Autonomy. Services are autonomous entities that can be considered a black-boxes, i.e., the concrete implementation cannot be seen from the outside and only the interfaces are visible.

Services that are composed together to an application or system are loosely coupled. Thus, a service can be exchanged without influencing the system itself. The only requirement is that the functionality and interfaces are the same. This is in contrast to distributed object oriented approaches, like CORBA [133], that lead to tight coupling.

Furthermore, traditional service-oriented architectures follow the Publish-Find-Bind paradigm. Services are published by the service provider in a service registry. Service consumers can search in the registry for services and, finally bind a service from a provider.

Web Services are an option to implement a service-oriented architecture. Therefore they propose three main specifications following the Publish-Find-Bind paradigm.

- SOAP [265] is a protocol for exchanging structured information based on XML. It is independent from the underlying transport protocol. So-called bindings are employed to use SOAP over a specific transport protocol.
- WSDL, the Web Services Description Language [115] is used to define interfaces. It is also based on XML and makes it possible to describe a service as a so-called endpoint.
- UDDI, the Universal Description Discovery and Integration [454] defines an interface for a service registry. In such a registry, developers can publish and

find services during the design of an application. Applications themselves can use the registry to find services with suitable interfaces at runtime.

These specifications supplement each other, but can also be used on their own. Furthermore, those specifications can be used to realize platform-independent applications, which do not rely upon specific programming languages. Web Services implementations exist for all major platforms and programming languages.

As identified in Section 30.1, reuse and extensibility in content and mechanism are essential. Service-oriented architectures take these two key points into account. New applications can be rapidly created out of existing services by combining them. New services can easily be deployed and adopted because of the well defined interfaces. Furthermore, the loose coupling is similar to typical Peer-to-Peer systems where peers are loosely coupled. Web Services can be used as building blocks of our architecture because of their clearly defined interface language and message format.

However, these service-oriented architectures emerged from the client-server approach and therefore do not take Peer-to-Peer mechanisms and self-organization into account. For example, UDDI defines interfaces for a service registry and is therefore not centralized, but current implementations are centralized and do not benefit from the Peer-to-Peer approach. In addition, content-based addressing as a key principle of Peer-to-Peer network organization is not taken into account. Current service-oriented architectures focus on addressing systems without considering the content.

30.2.2 ServiceNets

When services are offered through the collaboration of multiple participants, a ServiceNet is created. Peer-to-Peer file sharing and the document service mentioned before are examples of such ServiceNets. From a functional point of view, how many functions a ServiceNet offers is independent of the number of peers forming the ServiceNet. But the non-functional aspects, such as robustness and available content, might differ drastically with respect to the degree of distribution. These non-functional issues, which can be seen as quality or guarantees, also characterize a service, and might provide ServiceNets with the competitive edge over a centralized approach as indicated in Section 1.

A ServiceNet can be used from participating peers as well as from 'external peers' that do not participate in the ServiceNet. Mobile devices could, for example, be such service consumers where active participation within a ServiceNet is perhaps not reasonable because of very temporary availability.

Each ServiceNet uses an underlying Peer-to-Peer network that provides self-organizing mechanisms for network organization. Hence, there is a one-to-one mapping between a ServiceNet and a Peer-to-Peer network. For example,

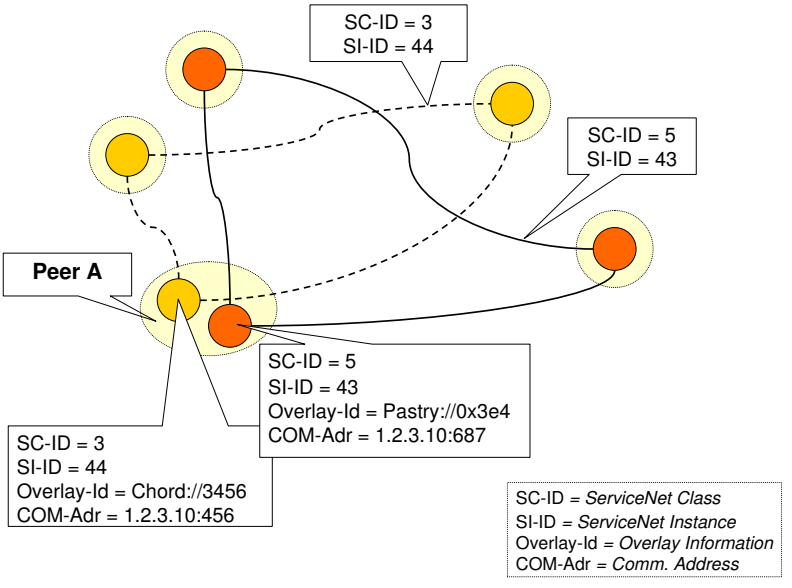

Fig. 30.2: ServiceNet example

an identity management service, which offers functions that are all based on a unique key, could use a DHT-based Peer-to-Peer network. On the other hand, the document service implements a keyword search for which a DHT-based Peer-to-Peer network may not be suitable. For a document service, an unstructured Peer-to-Peer network, like GIA [113], might be more suitable.

Using one Peer-to-Peer network for one ServiceNet has the advantage that the characteristics of the Peer-to-Peer network can be better considered. Clearly, to assist the process of service creation, a 'catalogue' is needed so that one can find the suitable Peer-to-Peer network for a specific service and ServiceNet, respectively, based on the required functionality or constraints. For example, service developers and providers must characterize their service by answering questions such as 'Is there a unique key for data elements?' etc. Afterwards they should get a recommendation for a Peer-to-Peer network their service should use. Papers like [346] provide an aid in building such a catalogue. Currently we are developing a model that gives us a basis for describing the characteristics and behavior of Peer-to-Peer networks. This model will then be used for simulation and evaluation.

A peer in a ServiceNet (Figure 30.2) is characterized by:

– *ServiceNet Class.* A ServiceNet Class (SC) is a unique identifier for the type of service which the ServiceNet offers. Thereby all peers of a ServiceNet offer the same kind of interface and functionality. A document service is one kind of service class.

- *ServiceNet Instance.* A ServiceNet Instance (SI) is a unique identifier of an instantiated ServiceNet Class. If there are two Document Services for example, they are both instantiated from the same Class, but the instances are independent of each other. E.g., if a ServiceNet uses Chord [575] as the underlying Peer-to-Peer network, one ServiceNet instance also corresponds to one Chord ring.
- *Overlay Information.* All 'overlay-specific information' can be summarized herein, such as Peer-to-Peer organization form and so on. A simple example could appear as follows: Chord://3456. Here the first part identifies the Peer-to-Peer network form and the second part the identifier that the node has in a specific Peer-to-Peer network.
- *Communication Address.* The Communication Address corresponds to the one that a peer uses in the communication network. A communication network could be an IP network or a Bluetooth network, for example.
- *Meta Information.* Meta-Information summarizes all additional information about a ServiceNet. This could be class-depended information or instance-depended information. An example: 'This ServiceNet was founded by UKA'.

Because there can be various ServiceNets, service discovery is needed. Service discovery offers publishing and searching functions. Therefore it has the same function as the service registry in service-oriented architectures. If we think of extensibility regarding mechanisms, service discovery is especially necessary for finding and publishing new services. In contrast to a lot of existing service registries, our service discovery is decentralized, i.e., the discovery itself is provided by a ServiceNet.

30.2.3 Peer Architecture

In this section we will outline our peer architecture which integrates the principles mentioned above. This platform offers a basis for current and future services for electronic markets. The architecture of a peer is depicted in Figure 30.3. This architecture can be divided into the following parts:

- *Communication Layer.* This layer provides an abstraction from a concrete communication network like networks based on IPv6. This layer gives the layer above the possibility to send and receive messages.
- *Overlay Layer.* The overlay layer makes various overlay techniques available. Chord and Pastry would be examples for DHT-based Peer-to-Peer networks and GIA could be an example for an unstructured network. This layer knows the appropriate algorithms and is responsible for initializing procedures. For example, when a peer wants to join a ServiceNet, this layer handles the initializing procedure like building a 'finger table' [575].
- *SOAP-Processor.* The SOAP-Processor translates programming language objects into corresponding SOAP messages. These messages will be ex-

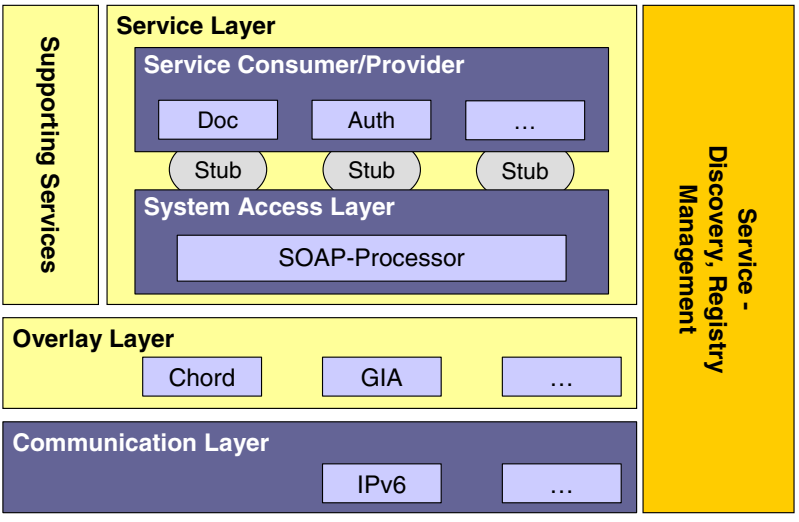

Fig. 30.3: Peer Architecture

changed between peers. These messages are used because of their independence from specific transport protocols that are used in the communication layer, and programming languages that are used for implementing the services.

– *Service Consumer/Provider.* This part summarizes all service consumers and providers. The connection between the underlying SOAP-Processor is handled through stubs that can be generated out of the WSDL description of a service. In this part all existing and future services will appear.
– *Service Management.* The Service Management includes all functions that are necessary to find, publish and bind services. Therefore, the Service Management makes use of the ServiceNet called service discovery, which was mentioned in the previous section. If a service is bound, the management has to inform the overlay layer so that a proper initialization can take place.
– *Supporting Services.* These are additional services that support the service developer by simplifying recurring tasks. For example, supporting services can support the handling of transactions. This would involve session handling and possibly calling roll-back functions, and compensations handlers, etc. In such cases, existing specifications like WS-AtomicTransaction [377] can be used.

30.3 Security, Robustness, and Privacy Challenges

Challenges with respect to security, robustness, and privacy can again be expressed in terms of community, application, and/or infrastructure requirements. Needs of the higher layers can be translated into infrastructure requirements while lower layer characteristics influence the security, robustness and privacy of the whole system. Conflicts between ease-of-use, security, robustness and market requirements have to be resolved: our project demands an open system that encourages potential users to participate in the electronic market. On the other hand, it is also a requirement that we are able to conclude a legally binding contract. Therefore, contracts have to be traceable, such that they can be retraced by a judge.

Moreover, natural persons and companies alike want to control the disclosure of information. There are differences between the US approach to privacy regarding personally identifiable information and the EU approach to data protection. According to [209], the US law is driven by 'the right to be alone' and the European law by the 'right of informational self-determination'. Hence in the US the main concern is to protect personal information against unauthorized usage. Besides that, (natural) persons in Europe have the right to determine what happens with their personally identifiable information. For example, they can cause that their email address is deleted within 30 days. These issues have to be considered in the system design.

Thus, security, robustness, and privacy can all be considered as cross-layer issues and the proper placements of respective functionalities on the various layers as well as their appropriate interactions are a key to success. In the following we first present a threat analysis in a top-down fashion. Afterwards we list technical challenges that arise primarily from following the Peer-to-Peer paradigm on the 'infrastructure' and 'application' layers. We then present three of these issues — persistent signatures, privacy aware data handling, and trust models — in greater detail.

30.3.1 Attack Classification/Threat Analysis

Following the classification of [458] we can differentiate between profitable attacks and 'merely' malicious ones. Profitable attacks take direct advantage of the attack, e.g., a participant can try to modify the search algorithms such that his offers always are found and hence the offers of competitors are not available. Malicious attacks intend only to destroy something and do not directly profit from that.

We can further classify attacks according to their target in: market place, market participants, and market objects. The aim of attacking the market place is to influence the system as a whole. These attacks range from denial-of-service attacks that cause a system breakdown to 'free riding' [366], which normally does not bring the system down but influences it negatively. The

attack against a market participant tries to prevent or influence a single participant or group from trading without aiming to influence the whole market. For example, a company can try to prevent a competitor from trading in order to sell more of its own products. The attack against market objects arises especially in Peer-to-Peer systems because there is not necessarily a dependency between market participant and object. A market object can be any good or item related to trading in the market place, e.g., an electricity offer.

In addition, we can distinguish between threats of external and internal origin. External threats result from someone that is not participating in the system, whereas internal threats result from participants. Internal attackers can make use of their knowledge of topologies and protocols to influence the system. Solutions for external threats are not our focus. Furthermore when we can deal with internal threats in an open system, we can inherently deal with external threats. However, the distinction can be used to clarify the new challenges.

To protect ourselves against internal attackers, we must evaluate the mechanisms of the involved Peer-to-Peer networks. Attacks can be directed at the routing and searching mechanisms, but also at the storing and replication mechanisms. The analysis gets more complex since most modifications of one mechanism have a direct impact on one or more of the others. In Section 30.2 we noted that we are using various Peer-to-Peer networks. This means that we must evaluate them all regarding security threats, but to do this efficiently we discuss common requirements that we have for a secure and robust system in Section 30.3.2. Based on these requirements, we will outline some generic solutions that can be applied to various Peer-to-Peer networks and systems in Section 30.3.3.

30.3.2 Peer-to-Peer-Related Challenges

Besides traditional issues like end-to-end security and service robustness, new challenges arise from using a Peer-to-Peer network:

Partial encryption. In contrast to traditional networks where routing and data is always clearly separated in header and payload, Peer-to-Peer systems sometimes mix those up. For example, performing a search in GIA [113] means that every involved node has to search in its local store of objects. If we could assume that all nodes store only their own objects, then there would not be a problem. But if we think about replication, then object encryption and integrity become problematic. If we encrypt the whole object like the payload in typical network protocols, nodes will not be able to search for these objects. Hence, we have to encrypt the object in such a way that essential information is accessible for routing and searching.

Persistent signatures. In some Peer-to-Peer systems replication mechanisms ensure that data is replicated to several places for safety reasons. If we use communication channel encryption, we can ensure that the data has not been exchanged between two parties. However, we can not guarantee the originator of the data, if the data was replicated on an intermediate peer. Therefore, we have to ensure that the signatures created by the originator are not lost on intermediate peers.

Privacy-aware data handling. Regarding privacy two main challenges arise. Through the replication mechanism, data and personally identifiable information may be distributed over the network. In contrast to the primary goal of the replication mechanism, the goal of privacy is to minimize the number of stored data objects that include personal information. Furthermore, the user (in Germany) has the right [97] to modify, block or delete personal information about him or herself. Such obligations are difficult to guarantee in a Peer-to-Peer system.

Topology robustness. Another issue that arises is fault-tolerant routing. As mentioned before, most Peer-to-Peer systems provide fault-tolerant routing for safety, but we also have to ensure that no malicious node can influence the routing table of nodes in such a way that some nodes can not be accessed, objects can not be found, or the whole market breaks down. Therefore, the routing and stabilizing algorithms of the involved Peer-to-Peer networks have to be reviewed. The algorithms should offer multiple or by-pass routes and they have to be stable against denial-of-service attacks.

Incentives. Besides the reachability attack mentioned above, an attacker can try to benefit from the network by 'free riding', i.e., using the resources of others but not contributing his or her own. To avoid such attacks incentives and fairness control measurements are necessary.

Trust models. In traditional systems there is always a central trustable entity. All system participants trust this entity and therefore it can be used for authentication issues. Certainly, this is not always a single system, but the authentication and trust problem is solved by centralizing it. Examples of such architectures are PKI and Kerberos. Also some Peer-to-Peer systems (e.g., PAST [526]) are based on some external trusted entities. As a result of this authentication problem, encryption and integrity become more difficult in Peer-to-Peer systems. Although it is easy to verify the integrity, without be able to verify the authentication this integrity check is useless. For designing appropriate distributed authentication mechanisms, threats like whitewashing [204] and the Sybil Attack [177] also have to be kept in mind.

Multiple signatures. Many trust models use cryptography to secure their trust information. Most often integrity of the trust information is guaranteed by attaching a signature. The requirement for such multiple signatures arises in some distributed trust models. Typical centralized models usually attach a single signature for each item of trust information. In distributed trust

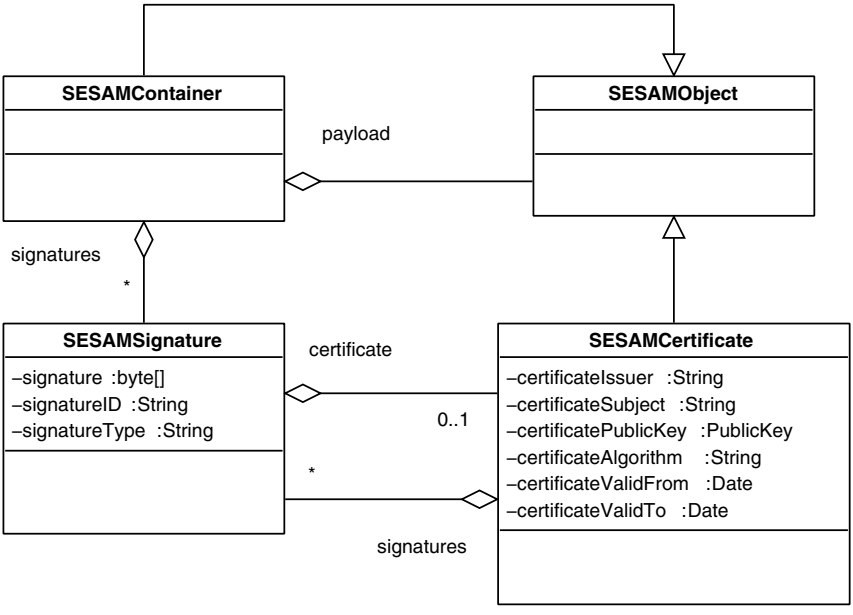

Fig. 30.4: The SESAM Data Model

models multiple issuers may sign the trust data as is done in PGP. Besides that, alternative trust models, like reputation or recommendation systems, require multiple signatures attached to a single item of trust information.

30.3.3 Selected Issues

Having analyzed the robustness and security requirements for our framework we now focus on persistent signatures, privacy-aware data handling and trust models.

Persistent Multiple Signatures

As described in Section 30.3.2, a mechanism to provide persistent signatures on data objects is required. We designed our data model based on the concepts of S/MIME [500] and PGP [645]. As shown in Figure 30.4, the signatures and certificates are an integral part of the data objects. This implies that any component which distributes or stores the objects will automatically serialize the object together with its signatures and certificates. Thus, the signatures are persistently coupled to the object they sign.

The root class of the data model is `SESAMContainer`, which consists of two attributes: payload and signatures. The payload is always a subclass of `SESAMObject`. Every `SESAMObject` can carry a list of `SESAMSignatures`, where every list element contains a single signature. The `SESAMSignature` class defines a set of common attributes and the attribute `certificate` referencing the related certificate. The class `SESAMCertificate` includes the public key, which was used to create the signature, and attributes containing information about issuer, subject and validity of the certificate.

The content (payload) of a data object can be integrity-protected with one or more independent signatures of different peers. To be able to remove one or more of these signatures without invalidating other signatures of this object, every two signatures must be independent of each other. Therefore, no previously existing signature will be included in the generation of a new signature. This proceeding can be useful if the validity period of the data object should be longer than the validity period of a signature and the corresponding certificate.

Trust Models

Using the data structure introduced in the last section (Figure 30.4) we can ensure the integrity of data objects. To enable applications for a distributed marketplace where electronic contracting is supported, authenticity of a peer's identity is a major issue.

While the integrity of data objects is provided by signatures, peer identity will be provided by digital certificates. Each signature contains an attribute which links a certificate to the public key used. This attribute itself must included in the generation of the signature, otherwise an attacker is able to change the used certificate. A certificate itself contains one or more signatures. This mechanism is suitable for building any trust model from simple certificate lists up to more complex certificate trees such as in reputation systems.

The diversity of applications in a Peer-to-Peer marketplace requires for different trust models to verify identities. Therefore, we develop trust model plugins which implement various trust models. First, we implement common trust models such as Certificate Authority based (X.509 [629]) or distributed models (PGP [645]). Later, plugins to support reputation and recommendation systems are added. All plugins are used by the trust component which offers a common interface for checking identities. The certificate and the chosen trust model are the input parameters of the verification method of the trust component.

Privacy-Aware Data Handling

Implicit routing in Peer-to-Peer networks can be used to guarantee anonymity as outlined in [169, 17, 124, 168]. Anonymity is a good way to establish privacy, because without personally identifiable information there are no privacy issues regarding personal information. But some market scenarios, like concluding an electricity contract require personal information for the delivery and accounting. Thus complete anonymization is no solution for that scenario. Anonymization as mentioned above 'only' anonymizes on the communication level, but on the higher level personal information can still be exchanged. For example, let's assume the communication between a browser and web server is not traceable, but the user enters his name on a registration form. Then the communication itself is anonymous, but not the transaction as a whole on the application layer.

Therefore we have to integrate mechanisms for correctly handling personally identifiable information. The project 'Platform for Privacy Preferences (P3P)' [620] is an approach by the World Wide Web Consortium (W3C) that facilitates for website operators the expression of their privacy policy. This standard can be used to inform users, but not to enforce the correct handling of data. The goal of the Enterprise Privacy Authorization Language (EPAL) [35] is to 'provide the ability to encode an enterprise's privacy-related data-handling policies and practices'. Thus policies can be easily exchanged between different applications and also be automatically enforced. In [427] an approach of identifier-based encryption (IBE) is presented to control the disclosure of personal information. This approach goes beyond EPAL by involving a third party in the process of information disclosure.

Through our modular approach, we can integrate the Peer-to-Peer networks mentioned above that guarantee anonymity in order to offer anonymity mechanisms to service developers. For scenarios where this is insufficient, our next step is to integrate the principles from EPAL and further approaches like IBE.

30.4 Summary

In this chapter we have argued that markets should be viewed as Peer-to-Peer systems on the community, application and infrastructure levels to achieve scalability, spontaneity and reliability in liberalized and harmonized electronic markets. While many proposals favor a tight connection of an application to a specific overlay network, the challenge we address is that of bringing together service orientation (as the Peer-to-Peer paradigm on the application level) with various overlay network techniques (as Peer-to-Peer networks on the infrastructure level). We presented our concept of 'ServiceNets' and the peer node architecture. We argued that there exist many benefits of such a

distributed approach to electronic markets. However, securing this approach is a difficult task. We presented the basic security challenges as well as some ideas to tackle them. The proposed framework is a first step to enable Peer-to-Peer markets: future work will have to come up with a catalogue of criteria for deciding which overlay network organization to use for which service, a good Peer-to-Peer network API, and a complete security framework. Thus, an easily deployable framework can be created that makes market participation more spontaneous and less costly.