A Peer-to-Peer Registry for Network Management Web Services

Torsten Klie
Technische Universität Braunschweig
Institute of Operating Systems and Computer Networks (IBR)
Mühlenpfordtstr. 23
38106 Braunschweig, Germany
Email: tklie@ibr.cs.tu-bs.de

Adrian Belger
Technische Universität Braunschweig
Institute of Computer and Network Engineering (IDA)
Hans-Sommer-Str. 66
38106 Braunschweig, Germany
Email: belger@ida.ing.tu-bs.de

Lars Wolf
Technische Universität Braunschweig
Institute of Operating Systems and Computer Networks (IBR)
Mühlenpfordtstr. 23
38106 Braunschweig, Germany
Email: wolf@ibr.cs.tu-bs.de

***Abstract*—Network management becomes more and more complex with growing networks. Web services are seen as a possible way to deal with the complexity. To exploit the advantages of Web services, a management system must be aware of the management functionality that is offered by the devices as Web services. In a distributed management environment, a centralized registry is not suited. In this paper, we propose a peer-to-peer registry that is based on semantic descriptions of the available management Web services. The idea is to let all devices that offer management Web services take part in the overlay peer-to-peer network, such that no external or central registry server is needed. We show how a policy-based management system that depends on Web services can benefit from such a registry, discuss some of the implementation issues, and give evaluation results.**

I. Introduction

Although networks seem to be almost ubiquitous these days, they are still growing. The vast majority of companies use networks to connect different company locations, to share resources, and to communicate via e-mail and voice over IP (VoIP). Not only the number of connected devices is growing, bandwidth and real-time requirements are growing, too. It is the task of network management to ensure that these requirements are not violated. This task is becoming more and more difficult due to the rising complexity that results from the heterogeneity. Standard management techniques such as the Simple Network Management Framework (SNMP) [1], command line interfaces and management scripts, commonly used by many administrators, are no longer suited to deal with heterogeneity in the network. For example, the management of a Juniper router is very different from the management of a Cisco router.

In the network management community, many people see Web services [2] as a possibility to solve some of the most important problems: First of all, since Web services are platform independent and use standard Internet protocols, they help to deal with the heterogeneity [3]. Second, they offer a unified communication model for network, application, and systems management [4]. Third, with Web service composition mechanisms, automation can be supported [5], [6]. To make use of these advantages, a management system must be aware of the available management Web services and its semantics, e.g. it must "understand" the differences between Juniper and Cisco routers. UDDI [7] is often used for this purpose, although it is difficult to add semantic support to it. Futhermore, UDDI is a centralized approach, and therefore not well suited for a registry of management Web services. The main argument against a centralized registry is the self-management paradigm that is becoming more and more popular [8], [9]. In a self-managing environment, management is done in a heavily distributed fashion. A centralized component is a single point of failure and an impurity [10]. A distributed approach, where the registry is formed as an overlay by the devices that offer the management Web services fits much better into the self-management paradigm.

This paper proposes a Web service registry based on peer-to-peer technology and shows, how this registry can be integrated into a policy-based management architecture for Autonomic Communications. It is structured as follows. Section II discusses related work regarding peer-to-peer Web service registries. PoMAS, a policy-based architecture based on Web services is introduced in Section III, and it is shown, how such a system can benefit from a peer-to-peer registry. In Section IV, we present our concept of a distributed peer-to-peer-based Web services registry for network management Web services. Section V explains some aspects of the implementation and the implemented prototype. Evaluation results are shown in Section VI, before we summarize the paper in Section VII.

II. Related Work

Universal Description, Discovery and Integration (UDDI) [7] specifies a central directory service that should serve two purposes:

1) Registering available Web services and
2) Searching for applicable Web services.

UDDI servers can be open to the public or kept private in an intranet. The information provided by an entry of the UDDI registry can be classified as White Pages (information about the publishing organization, Yellow Pages (categorized list of Web service descriptions), and Green Pages (technical information about services including the URI).

Thaden, Siberski, and Nejdl [11] presented an approach for a decentralized Web services registry, which allows distributed registration and discovery of Web services and is based on peer-to-peer technology. There is also a prototype implementation based on the Edutella framework [12]. The running example used in the paper is a special purpose image search over a large amount of databases, which offer different Web services for searching. In the prototype, the Web services are described using DAML-S. Moreover, they use an ontology containing concepts such as `ImageService`, `DeliverByMailImageService`, `VectorImage`, `GIF`, etc. Based on this information, candidate services for image search can be found using a matching algorithm proposed by Paolucci et al. [13]. The prototype supports queries in the Edutella query exchange language (QEL). It allows searching, but not service binding and execution. The idea is to connect different UDDI and other private registries with peer-to-peer technology, because neither a global UDDI registry nor globally replicated registry information seem realistic and achievable. The idea has been proposed in [14]. Furthermore, previous work on combining semantic web and peer-to-peer technology has been discussed by Maedche and Staab [15], as well as Nejdl et al. [16].

Castro et al. [17] proposed a universal ring as an overlay joined by all participating nodes. This ring provides a persistent store, a multicast communication service, and a distributed search algorithm. For service advertisement and discovery, a description of the service is stored in the ring. This description can be associated with different implementations of the service.

OWL-S [18] is an ontology for semantic description of Web services and Web service compositions. The description is subdivided into three components: ServiceProfile, ServiceModel, and ServiceGrounding. The ServiceProfile describes what a service does using functional properties (IOPE: input parameters, output parameters, preconditions, and effects) and non-functional properties (e.g. `serviceName`, `textDescription`). Especially the functional properties are used to publish and find OWL-S Web services. In the ServiceModel, a Web service is mapped to one or more processes. The inputs and outputs of processes can be defined using OWL classes [19] or W3C XML Schema types [20], whereas the preconditions and effects can be modeled as logic formulas. Processes can be AtomicProcesses, CompositeProcesses or SimpleProcesses (processes that do not have a grounding and cannot be executed). All processes (except SimpleProcesses) need a ServiceGrounding that binds them to concrete specifications and messages. The grounding makes use of the Web Service Description Language (WSDL) [21].

III. PoMAS - A Policy-based Management System Using Web Services

In this section, we describe PoMAS[1], an architecture for Autonomic Communications [22], [23], which is based on Web services. PoMAS uses a policy refinement algorithm based on automatic Web service composition. This refinement algorithm stresses the need for a powerful registry component to find relevant services that can be included in the composition. The following description of PoMAS is meant as a brief introduction to PoMAS, for more details refer to [23].

A. PoMAS Architecture

PoMAS contains three parts (see Figure 1): the network devices layer, the policy engine, and the user interface.

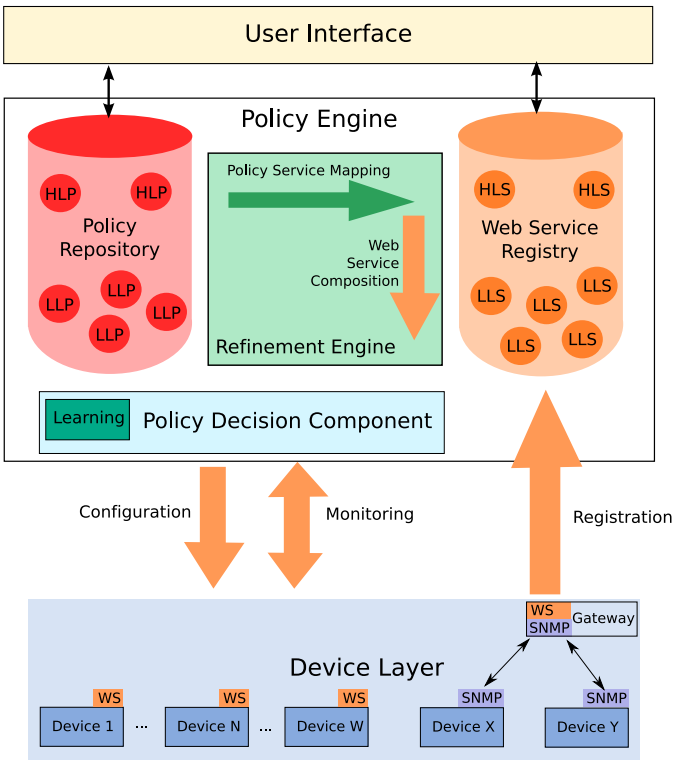

Fig. 1. A Policy-based Architecture for Autonomic Communications

The devices on the device layer are expected to offer Web services for configuration, for monitoring, and/or for notification. These services are called Low-level services (LLS). Network devices register their LLS in the registry, along with information about the structure and semantics of their LLS.

The refinement engine uses automatic Web service composition to combine different higher-level services (HLS) to perform aggregated tasks [24]. For example, it is possible to compose the Web services for a certain management task (LLS) into a single HLS that performs the operation in an entire subnet.

With the user interface, the administrator can specify policies and additional (high-level) services.

B. Policy Refinement Algorithm

The policy refinement approach is based on semantic Web services in OWL-S. It assumes a Web service based management infrastructure with management Web services as low-level services (LLS) offered by the devices. Automatic Web

[1]The abbreviation stands for "**Po**licy-based **M**anagement System for **A**utonomic **S**elf-organization".

service composition is used to combine different available Web services in order to obtain a desired higher-level result. The idea is to automatically create OWL-S services that can perform monitoring or configuration tasks needed by a policy.

The task of the policy refinement engine is to break down a high-level policy (HLP) into several HLS descriptions. The refinement engine tries to compose services that perform the tasks of the HLS, using the existing LLS offered by the devices. It receives policies from the GUI and sends a refined policy in the form of OWL-S descriptions of composite services combined with the extracted policy event (if present) to the policy decision component. In order to process the policy, the refinement engine needs access to a Web service repository (see Figure 2).

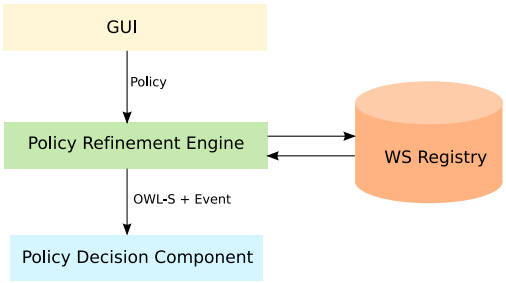

Fig. 2. Interactions of the policy refinement engine and the Web Services registry

In order to find the LLS that have to be combined, a description of the infrastructure and the available (low-level) management Web services is needed. We use ontologies, because they provide concepts and abstractions needed for the semantic description of the devices in the network and their management functionality. OWL-S descriptions of the available services include their IOPEs and make use of these ontologies. The refinement engine is responsible for identifying the needed IOPEs by extracting them from the policies and for finding relevant services.

C. Using a Peer-to-Peer Registry

As described above, a centralized registry such as UDDI is an impurity in a self-management environment. A distributed approach, where the registry is formed as an overlay by the devices that offer the management Web services fits much better into the self-management paradigm.

The idea is to let all devices that offer management Web services take part in the overlay peer-to-peer network, such that no external or central registry server is needed. The registry function is offered by the devices themselves. Figure 3 illustrates this idea.

However, not all devices are powerful enough to participate as "full members" of the overlay. Therefore, different roles with different responsibilities, assigned based on the available resources on the devices have to be supported.

As described in Section III-B, the registry has to be consulted for every policy refinement. Thus, the output of the refinement may vary, if the contents of the registry changes.

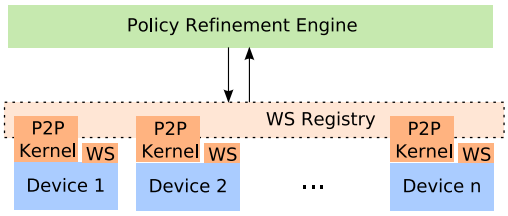

Fig. 3. The Web service registry as a peer-to-peer overlay in PoMAS

However, this reflects the volatile nature of the network. A stable network without frequent changes will have a more stable refinements.

IV. Concept of a Peer-to-Peer Registry for Management Web Services

This section presents the concept of the proposed peer-to-peer registry. The registry has been designed to work with PoMAS, as a "plug-in". Thus, it does not rely on the policy-based architecture and could be also used in other settings and for purposes others than network management.

In the following, we describe the application of the registry with PoMAS and discuss how service are described. Then, we reveal details of the messages that are used for external and internal communication and describe the overlay network. Finally, we sketch some of the used algorithms briefly.

A. Application Scenario

Adding users to an organization's network system is a typical task for an administrator. In the university context, for example, new user accounts are needed not only for new employees, but also for students participating in special practical courses, for conference participants, or for specific other external guests. Usually, the administrator has to take care that the different user accounts get the correct permissions according to their roles. Sometimes, they are assisted by scripts or predefined work-flows.

The network for the user management example consists of different servers and services, including wireless access, Mail service, DB servers, VPN, file server, etc. There are different company locations. Each location has a router and several different servers. Furthermore, there are wireless LANs at each location. The company is divided into departments. The system offers services such as SPAM filtering, mailing lists, and file backup service.

If all these devices offer their management functionality as Web services and PoMAS is used as a policy-based management system, configuration for new users can be automated. An administrator has to add a new user to the system. PoMAS will notice that it has to call the Web services of the devices to check whether all policies for users (such as "all employees have an email account") are fulfilled. If not, PoMAs will call the Web service of the email server to create the missing email account. This Web service could be a composed one, e.g. it could consist of a Web service to determine the relevant email server for the new user, another one the generate an email

address for the new user which is not already used, and the actual Web service to create the account.

1) Scenario 1 – Adding a Component: When a new component such as an email server is connected to the network, it will try to register its management Web services at the registry. The registry can be found via broadcast or a service location protocol. For registration, the semantic description of the service (see Section IV-B) is used. Thus, a service is stored in the registry according to the URIs of its concepts for the IOPEs. After the registration, the management Web services of the new component can be found by consulting the registry.

2) Scenario 2 – Performing Configuration Changes: If a configuration change (such as adding a new user to the system) has to be performed, PoMAS uses the registry to search for all relevant services. Relevant in this case means the Web service belongs to a desired target device and it performs the requested operation (i.e. providing needed properties for the new user). To determine which services are relevant, their semantic description (see Section IV-B) is used. By calling all the Web services that are the result of the query, the configuration change can be done without requiring the administrator to have any further knowledge about the location of the devices, the network topology, or other specific information.

B. Service Description

The Web services that offer management functionality are described using OWL-S. These semantic descriptions are stored in the peer-to-peer registry. Using semantic descriptions allows searching a service by a desired effect, not only by name. Inference engines can be used to find services providing equivalent outputs or effects. An example of such a conceptual equivalence is "duplex mode" and "double-sided printing mode". Also, inheritance can be exploited. For example, if a service producing a URI is needed, a service that outputs a URL will also fit.

In order to support such queries, the objects, i.e. the service descriptions, have to be distributed in the peer-to-peer network not according to their names (or hashes on it) but according to its concepts. Since we want to consider both output parameters and effects, a service description may be stored more than once. For example, a service changing the printing mode and returning an error code and an error text may be stored under the concept "printing mode", "error code" and "error text".

In order to effectively use these ontologies in a production system, they must cover all needed aspects of networks and system. Although no "world ontology" is needed, a set of standardized management ontologies should be used – which currently does not exist. However, we used parts of the DEN-ng [25] information model as a starting point to create the needed ontologies.

C. Messages from Clients to the Registry

Communication with the peer-to-peer registry is done using two request/reply message pairs, which are shown in the following. Since we are working in a SOAP, WSDL, and UDDI environment, it seems obvious to choose an XML-based [26] message protocol. For transport, User Datagram Protocol (UDP) is sufficient, because we do not use state-based communication but a simple request/reply protocol. Reliable transport is also not needed since messages can be repeated if they get lost. Fig. 4 shows the grammar of the registration message in W3C XML Schema. It takes an RDF document containing the OWL-S description of the Web service as argument.

```
<xsd:element name="XMLServiceData">
  <xsd:complexType name="rdfType">
    <xsd:sequence>
      <xsd:element ref="rdf:RDF">
        <xsd:annotation>
          <xsd:documentation xml:lang="en">
            OWL-S service description in an RDF document
          </xsd:documentation>
        </xsd:annotation>
      </xsd:element>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

Fig. 4. Excerpt of the XML Schema of the registration message

A query is done in the same way as the registration. An OWL-S description of the desired service is specified as an argument. Of course, the OWL-S description of the desired service will be much less complete than that given during the registration since only effects and outputs are used. Figure 5 shows an OWL-S description for a desired service.

```
<rdf:RDF>
  <!-- namespace declarations and OWL imports omitted -->
  <owl:Ontology rdf:about="">
    <owl:imports rdf:resource="&service;" />
    <owl:imports rdf:resource="&profile;" />
  </owl:Ontology>
  <service:Service rdf:ID="DesiredService">
    <service:presents rdf:resource="Profile_Desired_Service"/>
    <service:describedBy rdf:resource="DesiredProcessModel"/>
    <service:supports rdf:resource="DesiredGrounding"/>
  </service:Service>
  <profile:Profile
      rdf:ID="Profile_Desired_Service">
    <service:presentedBy rdf:resource="DesiredService"/>
    <profile:has_process rdf:resource="DesiredProcessModel"/>
    <!-- desired outputs and effects -->
    <profile:hasEffect rdf:resource="SetDuplexModeEffect"/>
    <profile:hasOutput rdf:resource="ErrorCodeOutput"/>
    <profile:hasOutput rdf:resource="ErrorMessageOutput"/>
</profile:Profile>
</rdf:RDF>
```

Fig. 5. OWL-S description of a desired service

D. The Overlay Network

One of the main issues that have to be solved when using peer-to-peer technology is the organization of the overlay network. The topology of the overlay network may differ completely from the underlying "real" network topology. Therefore, the peer-to-peer system has to provide its own routing methods, which should be robust enough to deal with changes in the underlying network structure as well as with link faults or peer faults.

Platform independence and XML-based communication are important characteristics of Web service environments. The peer-to-peer framework should fit well into this paradigm. Therefore, we chose JXTA [27]. JXTA supports both rendezvous peers and edge peers. Rendezvous peers are special peers that hold the shared resource distributed index (SRDI). The SRDI is a distributed algorithm for creating and maintaining a conceptual resource index of a network [28]. The rendezvous peers take part in a distributed hash table (DHT) to answer queries. Edge peers are normal, non-rendezvous peers. Devices with limited resources can participate as edge peers, whereas more powerful devices take more responsibility and act as rendezvous peers.

E. Internal Communication

For the internal communication, i.e. the communication inside the peer-to-peer overlay network, JXTA offers a large number of different pipes. `JXTABiDiPipes`, for example, offer a bidirectional communication channel which can be used in a reliable way optionally. The messages used for internal communication are the same as described in Section IV-C.

F. Important Algorithms

1) Finding Responsible Peers: Every peer participating in the overlay network of the peer-to-peer registry must assign the same peer to a Web service. The organization of the peers is done by the JXTA framework. It provides information about available peers. To assign concept to peers, we use a version of a content addressable network (CAN) [29] that has been adapted to JXTA IDs. In contrast to the original CAN, the responsible peer is not identified based on its peer ID but on the published pipe ID in order to achieve a better distribution of the peers. Figure 6 depicts how the algorithm works.

The algorithm starts by checking a trivial case: if there is only one known peer in the network, this peer is responsible. Otherwise, the decision will be based on hash values. Before calculating the hash values of the concepts, the reasoner is used to find compatible concepts for which the hash value calculation is also done. After that, the algorithm tries to find a peer for each hash value with a value minimally greater than the calculated one. If there is no peer with a minimally greater hash value, one with a value minimally smaller is searched. The resulting peer is the responsible peer. To calculate the hash value, well-established hash functions such as MD5 [30] or SHA1 [31] can be used.

2) Processing Requests: The first step of processing a request is to find the responsible peers (as described above). A request may include several concepts. Therefore, more than one peer may be responsible. Also, the reasoner checks for compatible concepts before hash values are calculated. The list of responsible peers will be iterated. In case the responsible peer is the local host, the request can be answered locally. Otherwise, the request has to be forwarded to the other peer. The forwarding peer is responsible for collecting the answers from the different peers and sending back an aggregated answer to the origin of the request. If an error occurs, this peer has to send back an error message. Figure 7 illustrates this algorithm.

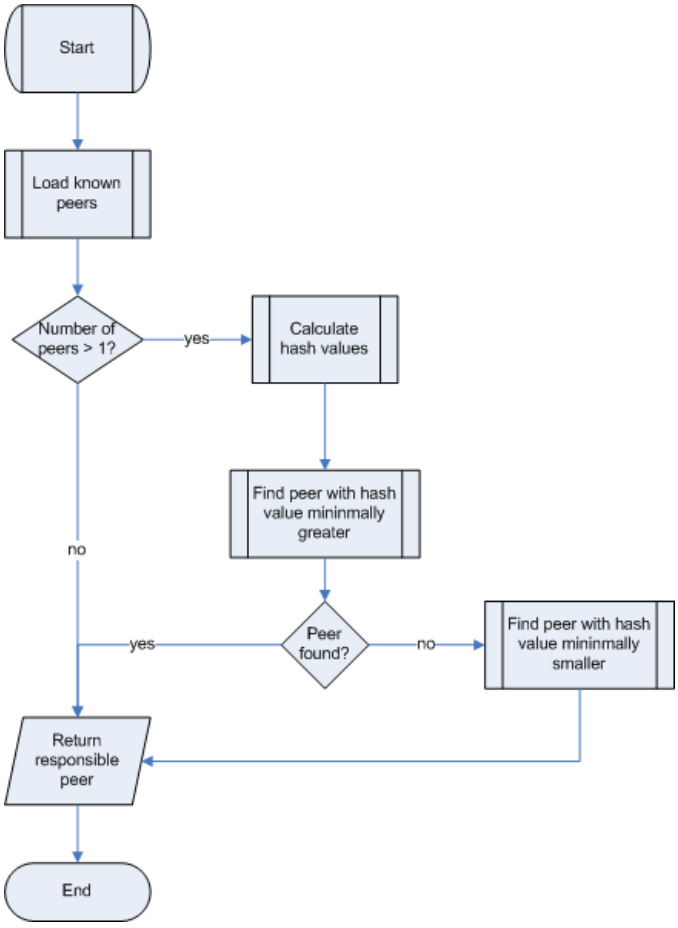

Fig. 6. Flow diagram showing the algorithm that is used to select the responsible peer for given attribute

3) Ensuring Data Consistency: Peer-to-peer networks may experience continuous change. Peers may be added and removed, either due to voluntary actions of administrators or because of failures. This is a hazard to the consistency of the data in the peer-to-peer network. If a peer leaves the network voluntarily, it has to transfer its data to another peer. Therefore, a new responsible peer will be selected (see above).

However, peers may also leave involuntarily due to faults or failure. In this case, data will be lost. To keep the stored data consistent with the actual situation, data should be reloaded from time to time. To lower the risk of data loss, redundancy can be added to the peer-to-peer network by assigning more than one responsible peer. However, a trade-off has to be made between protection against data loss and the added overhead. This trade-off depends on the planned usage environment of the system. In a stable environment with few failures and changes, redundancy should be kept low.

V. Implementation

A prototype of the registry has been developed in Java 1.6, using the OWL-S API [32] for OWL-S processing. The

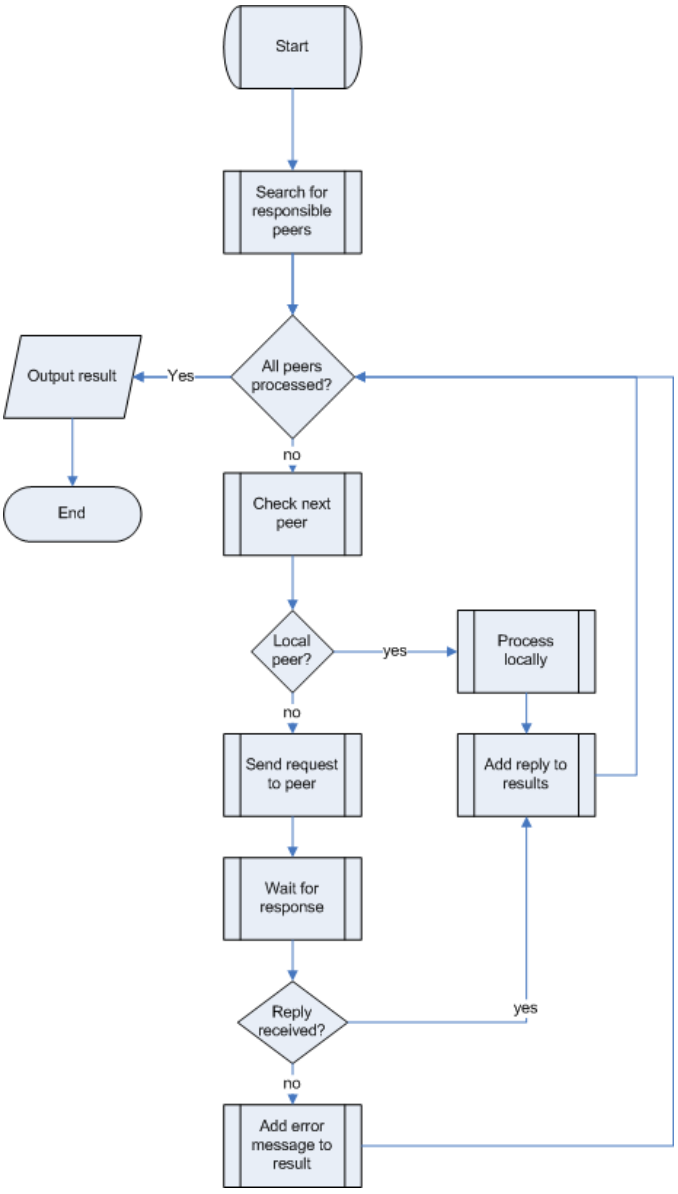

Fig. 7. Flow diagram for request processing

prototype consists of three main parts: Registry Server, XML Message, and Peer-to-Peer.

In addition to the main parts mentioned above, which are implemented as different packages, there is a test package that provides JUnit tests for the parts that are independent from the peer-to-peer overlay network.

A. Registry Server

This part contains the main class of the system (`RegistryServer`), which is responsible for controlling the system as well as the functionality of the local server. Its main functions are the local server and the peer-to-peer kernel, which are implemented as different threads.

The server listens to UDP packets and forwards them to the peer-to-peer kernel, if they have been identified as requests by the XML parser. The kernel determines the responsible peer(s) and forwards the request. The local server collects and aggregates replies.

The local data is managed by the class `DataStorage`. It uses a `TreeMap` with the URIs and the device IDs. A FIFO queue is used to manage the data validity efficiently, without frequent searching through the entire database.

B. XML Message

This package implements the object-oriented message representation. It provides the necessary scanner and parser for the messages, including OWL-S support. In order to ensure a modular design that facilitates future extensions, the messages are created using the visitor pattern [33].

C. Peer-to-Peer

This package contains the peer-to-peer kernel (`P2PCore`) that is started in the registy server (see Section V-A). The thread first initializes the JXTA framework. After that, the peer connects itself to the overlay network. The peer-to-peer kernel publishes the local resources, especially the contents of the `JXTAServerPipe` periodically. The `JXTAServerPipe` is the input channel for requests in the overlay network.

For efficiency reasons, an internal representation independent from the JXTA framework is used for the data storage. Advertisements are stored in `AdvertisementKeeper`, which is implemented in analogy to the `DataStore`, including the validity check using a FIFO queue.

Another important class in the peer-to-peer package is `SHA1DigestProducer`. It can be used to calculate hash values with SHA1, which is used for assigning information to individual peers according to the CAN principle.

For startup, a file named `knownhosts.dat` is used. Each entry in the file is contacted with a timeout, which defaults to 30 s. If no rendezvous connection can be established during this time, the next entry is probed.

VI. Evaluation

In order to evaluate our prototype, we have performed different experiments. In the following, we will discuss the time needed for query processing. Then, we will take a look at service registration and the required initial OWL-S processing.

A. Processing Queries

Scalability with a growing number of peers is an important criterion for the usablility of a peer-to-peer solution. To evaluate the implemented prototype, we measured the average duration of processing a request (until the reception of the answer). To ensure scalability, this duration should increase only moderately with the increase of the number of peers.

Table I shows our measurement results for the average processing time (in ms) of different requests. The peers have been initialized uniformly inside a Gigabit LAN. Afterwards, a dedicated host sent a series of 100 random requests to the peer-to-peer network. The experiment has been repeated 50 times. Different rendezvous peers have been used.

# Peers	# Rendezvous Peers			
	1	2	3	4
1	6.25	–	–	–
2	15.16	6.88	–	–
3	38.34	11.41	9.08	–
4	45.16	31.87	25.63	14.89
6	47.62	51.87	49.06	50.00
8	51.40	58.12	62.08	46.76
12	63.44	58.43	75.13	72.50
16	69.38	77.81	76.87	89.84
20	75.15	89.47	82.11	92.03

TABLE I
MEASUREMENT RESULTS FOR THE DURATION OF THE REQUEST PROCESSING (IN MS) FOR DIFFERENT NUMBERS OF PEERS

Fig. 8 illustrates the increase of the average processing time related to the number of peers in the system. The huge increase in the beginning of the curves results from the following effect. If the number of peers is equal to the number of rendezvous peers, queries can be answered locally, resulting in average processing times around 6.5 ms.

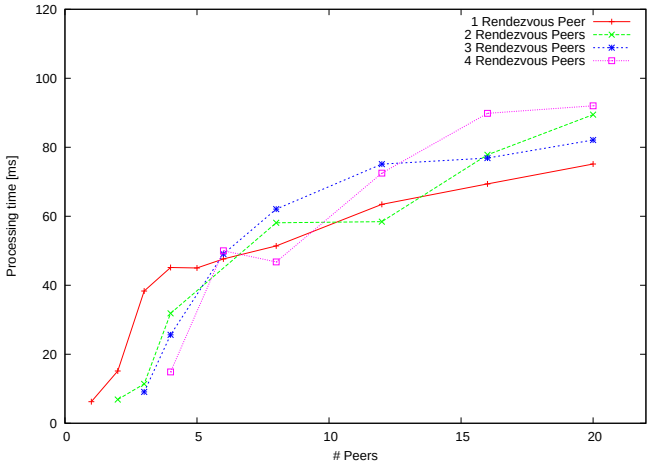

Fig. 8. Comparison of the average processing time (in ms) with different numbers of rendezvous peers.

If we compare the results for 8 and 16 peers, we see an increase of processing time of ca. 34% when the number of peers is doubled (with 1 - 3 rendezvous peers). This increase results from a decreasing probability that the contacted peer is already the responsible peer. Extrapolating these results for 1280 peers, we should be around 0.5 s. for the average processing time. This is consistent with the evaluation results for the underlying JXTA framework [34], [35].

B. Service Registration

The majority of the Web services is registered when the peer-to-peer registry is started. Later, new services will be added when new devices with additional management Web services are integrated into the network.

In the following, we present measurement results for the start-up phase (see Table II). We measured the time that it took to register all available Web services in the peer-to-peer registry using different sets of Web services with different numbers of services. We used a peer-to-peer network of 16 peers, where 4 peers act as rendezvous peers.

# Web services	Registration time (ms)
1	10446
10	10910
20	12159
50	15163
100	23700

TABLE II
TIME NEEDED TO REGISTER WEB SERVICES (WITH 12 EDGE PEERS AND 4 RENDEZVOUS PEERS)

The time needed for the registration rises with a growing number of Web services, as expected. However, besides a relatively slow registration, we can observe a large plateau in the beginning. Even the registration of a single Web service takes more than 10 s. Therefore, we analyze the OWL(-S) initialization separately from the registration request processing. Table III shows only the time needed for processing the registration request, without the ontology preparation.

# Web services	Registration request precessing time (ms)
1	64.85
10	641.45
20	1296.15
50	3471.15
100	7963.15

TABLE III
TIME NEEDED TO PROCESS A REQUEST (IN MS), WITHOUT INITIAL OWL-S PROCESSING

Figure 9 depicts the results graphically. We see that especially the OWL-S processing is responsible for the long registration time. The actual request processing, i.e. the processing done by the peer-to-peer registry, shown by the line at the bottom, is much faster. The reason for the weak performance of the OWL-S processing is the slow ontology set-up of the used libraries (Jena and OWL-S API).

VII. CONCLUSION

In this paper, we have shown the concept of a peer-to-peer-based registry for Web services. This registry is part of PoMAS, a policy-based management system for Autonomic Communications. The registry is realized as an overlay formed by the devices that offer management functionality as Web services, because in this way a single point of failure can be avoided and the registry fits better into the self-management paradigm. The contribution of the paper is to show how a distributed Web service registry that stores management Web services according to their IOPEs using OWL-S can support PoMAS.

Initial evaluation results demonstrate that the registry scales linear to the number of Web services being registered. Although linear is sufficient here, optimizations are still needed, especially for the ontology processing.

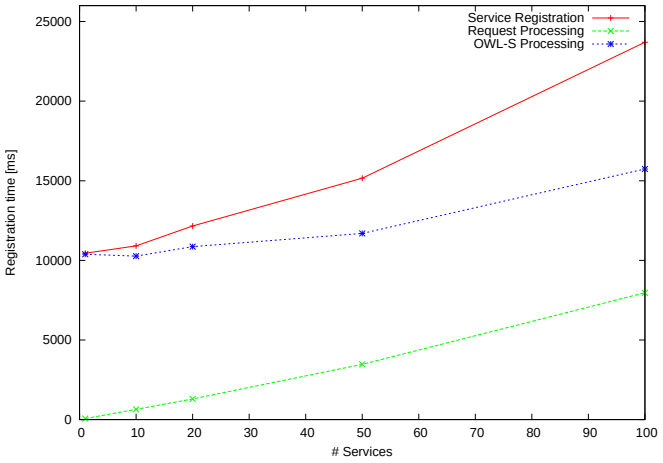

Fig. 9. Time needed to register services in the registry (with 12 edge peers and 4 rendezvous peers) depending on the number of Web services.

In our future work, we will investigate and integrate distributed ontology processing techniques. This includes substitution of the used OWL(-S) libraries with less resource consuming alternatives. Also, we want to reduce memory consumption of our software to make it more lightweight and better suited for network devices with limited resources.

REFERENCES

[1] D. Harrington, R. Presuhn, and B. Wijnen, "An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks," RFC 3411, Dec. 2002.

[2] D. Booth, H. Haas, and F. McCabe, "Web Services Architecture, W3C Working Group Note," Feb. 2004. [Online]. Available: http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/

[3] J. Schönwälder, A. Pras, and J.-P. Martin-Flatin, "On the Future of Internet Management Technologies," *IEEE Communications Magazine*, vol. 41, no. 10, Oct. 2003.

[4] J.-P. Martin-Flatin, *Web-Based Management of IP Networks and Systems*. Wiley, 2002.

[5] D. Berardi, G. D. Giacomo, M. Mecella, and D. Calvanese, "Automatic Web Service Composition: Service-tailored vs. Client-tailored Approaches," in *Proc. 8th Int'l Conference on Artificial Intelligence and Symbolic Computation*, Beijing, China, Sept. 2006.

[6] T. Klie, F. Gebhard, and S. Fischer, "Towards Automatic Composition of Network Management Web Services," in *Proc. of 10th IFIP/IEEE International Symposium on Integrated Management (IM)*, Munich, Germany, May 2007.

[7] L. Clement, A. Hately, C. von Riegen, and T. Rogers, "UDDI Version 3.0.2," OASIS Technical Commitee Draft <uddi_v3>, Oct. 2004. [Online]. Available: http://uddi.org/pubs/uddi_v3.htm

[8] M. Smirnov, "Autonomic Communication: Research Agenda for a New Communication Paradigm," Fraunhofer FOKUS," White Paper, Nov. 2004.

[9] X. Gu, J. Strassner, J. Xie, L. Wolf, and T. Suda, "Autonomic Multimedia Communications: Where Are We Now?" *Proceedings of the IEEE*, vol. 96, no. 1, pp. 143–154, Jan. 2008.

[10] C. Prehofer and C. Bettstetter, "Self-Organization in Communication Networks: Principles and Design Paradigms," *IEEE Communications Magazine*, vol. 43, no. 7, pp. 78 – 85, July 2005.

[11] U. Thaden, W. Siberski, and W. Nejdl, "A Semantic Web based Peer-to-Peer Service Registry Network," L3S Research Center, Tech. Rep., 2003.

[12] W. Nejdl, B. Wolf, C. Qu, S. Decker, M. Sintek, A. Naeve, M. Nilsson, M. Palmér, and T. Risch, "EDUTELLA: A P2P Networking Infrastructure Based on RDF," in *Proc. 11th Int'l. World Wide Web Conference (WWW)*, Honululu, Hawaii, USA, July 2002.

[13] M. Paolucci, T. Kawamura, T. Payne, and K. Sycara, "Semantic Matching of Web Services Capabilities," in *1st Int'l. Semantic Web Conference (ISCW)*, Sardinia, Italy, June 2002.

[14] G. Glass, *Web Services*. Prentice Hall, Nov. 2001.

[15] A. Maedche and S. Staab, "Services on the Move – Towards P2P-Enabled Semantic Web Services," in *Proc. 10th Int'l. Conference on Information Technology and Travel & Tourism (ENTER)*, A. J. Frew, M. Hitz, and P. 'Connor, Eds. Helsinki, Finland: Springer, Jan. 2003.

[16] M. Schlosser, M. Sintek, S. Decker, and W. Nejdl, "HyperCuP – Hypercubes, Ontologies and Efficient Search on P2P Networks," in *Proc. 1st Int'l Workshop on Agents and Peer-to-Peer Computing (AP2PC)*, Bologna, Italy, July 2002.

[17] M. Castro, P. Druschel, A.-M. Kermarrec, and A. Rowstron, "One Ring to Rule them All: Service Discovery and Binding in Structured Peer-to-Peer Overlay Networks," in *Proc. 10th ACM SIGOPS European Workshop*, Saint-Emillion, France, July 2002.

[18] D. Martin, M. Burstein, J. Hobbs, O. Lassila, D. McDermott, S. McIlraith, S. Narayanan, M. Paolucci, B. Parsia, T. Payne, E. Sirin, N. Srinivasan, and K. Sycara, "OWL-S: Semantic Markup for Web Services," DAML White Paper Release 1.1, Nov. 2004. [Online]. Available: http://www.daml.org/services/owl-s/1.1/overview/

[19] P. Patel-Schneider, P. Hayes, and I. Horrocks, "OWL Web Ontology Language Semantics and Abstract Syntax, W3C Recommendation," Feb. 2004. [Online]. Available: http://www.w3.org/TR/owl-semantics/

[20] P. V. Biron and A. Malhotra, "XML Schema Part 2: Datatypes Second Edition, W3C Recommendation," Oct. 2004. [Online]. Available: http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/

[21] R. Chinnici, J.-J. Moreau, A. Ryman, and S. Weerawarana, "Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language," W3C Recommendation REC-wsdl20-20070626, June 2007. [Online]. Available: http://www.w3.org/TR/2007/REC-wsdl20-20070626/

[22] T. Klie and L. Wolf, "Autonomic Policy-based Management using Web Services," in *Proc. 2nd CoNext Conference*, Lisbon, Portugal, Dec. 2006.

[23] T. Klie, "Policy Refinement Using Automatic Composition of Management Web Services in a a Policy-based Autonomic Communications Environment," Ph.D. dissertation, TU Braunschweig, Nov. 2008.

[24] T. Klie, B. Ernst, and L. Wolf, "Automatic Policy Refinement Using OWL-S and Semantic Infrastr ucture Information," in *Proc. 2nd IEEE Int. Workshop on Modelling Autonomic Communications Environments (MACE)*, San José, CA., USA, Oct. 2007.

[25] J. Strassner, "DEN-ng: Achieving Business-driven Network Management," in *Proc. 2002 IEEE/IFIP Network Operations and Management Symposium*, Florence, Italy, Apr. 2002, pp. 753–766.

[26] T. Bray, J. Paol, M. Sperberg-McQueen, E. Maler, F. Yergeau, and J. Cowan, "Extensible Markup Language (XML) 1.1 (Second Edition)," W3C Recommendation REC-xml11-20060816, Sept. 2006. [Online]. Available: http://www.w3.org/TR/2006/REC-xml11-20060816/

[27] M. Duigou *et al.*, "JXTA v2.0 Protocols Specification, JXTA Specification," Aug. 2007. [Online]. Available: https://jxta-spec.dev.java.net/nonav/JXTAProtocols.html

[28] S. Li, "JXTA 2: A high-performance, massively scalable P2P network," IBM DeveloperWorks, Nov. 2003. [Online]. Available: http://www.ibm.com/developerworks/java/library/j-jxta2/

[29] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker., "A Scalable Content-Addressable Network," in *Proc. 7th ACM SIGCOMM Conference*, San Diego, CA, USA, Aug. 2001.

[30] R. L. Rivest, "MD5 Message-Digest Algorithm," RFC 1321, Apr. 1992. [Online]. Available: http://tools.ietf.org/html/rfc1321

[31] D. E. Eastlake and P. E. Jones, "US Secure Hash Algorithm 1 (SHA1)," RFC 3174, Sept. 2001. [Online]. Available: http://tools.ietf.org/html/rfc3174

[32] E. Sirin, "OWL-S API," Project WWW Page, Maryland Information and Network Dynamics Lab Semantic Web Agents Project (MINDSWAP), 2004. [Online]. Available: http://www.mindswap.org/2004/owl-s/api/

[33] B. Meyer and K. Arnout, "Componentization: the Visitor example," *IEEE Computer*, vol. 39, no. 7, pp. 23–30, July 2006.

[34] E. Halepovic, R. Deters, and B. Traversat, "Performance Evaluation of JXTA Rendezvous," in *Proc. Int'l Sympoium on Distributed Objects and Applications*, Larnaca, Cyprus, Oct. 2004.

[35] G. Antoniu, P. Hatcher, M. Jan, and D. A. Noblet, "Performance evaluation of JXTA communication layers," in *Proc. 5th IEEE International Symposium on Cluster Computing and the Grid (CCGrid)*, Cardiff, UK, May 2005.