

# Interacting the Edutella/JXTA Peer-to-Peer Network with Web Services

Changtao Qu

*Learning Lab Lower Saxony*

*University of Hannover*

*Expo Plaza 1, D-30539, Hannover, Germany*

*qu @learninglab.de*

Wolfgang Nejdl

*Learning Lab Lower Saxony*

*University of Hannover*

*Expo Plaza 1, D-30539, Hannover, Germany*

*nejdl @learninglab.de*

## Abstract

*Edutella/JXTA is a P2P (Peer-to-Peer) Semantic Web application, which is aimed to accommodate heterogeneous resource metadata repositories in a P2P manner and further facilitate the exchange of the resource metadata based on RDF. Whereas the initial use cases of Edutella/JXTA are defined and implemented in the context of a pure P2P setting, where all Edutella participants are JXTA peers interacting with each other via various P2P services, in this paper we investigate the interaction between Edutella/JXTA and Web services with the purpose of exchanging distributed functionalities between the two platforms. In terms of two typical interaction scenarios: (1) exposing existing Edutella/JXTA P2P services as Web services; and (2) integrating Web services enabled content providers into Edutella/JXTA, we propose an approach for achieving the service layer interaction between Edutella/JXTA and Web services through so-called Web services/Edutella proxies.*

## 1. Introduction

Edutella is a P2P (Peer-to-Peer) Semantic Web application [7], which is aimed to accommodate heterogeneous resource metadata repositories in a P2P manner and further facilitate the exchange of the resource metadata based on RDF (Resource Description Framework). In Edutella we make the essential assumption that all Edutella resources can be described in RDF and further all Edutella functionalities can be mediated through RDF statements and the queries against these statements. As we believe distributed P2P networks can nicely fit with the modular nature of RDF metadata, we employ Sun's open source P2P platform JXTA [4] as the technical basis of Edutella. As a result, the initial use cases of Edutella/JXTA are defined and implemented in the context of a pure P2P setting, where all Edutella participants are JXTA peers interacting with each other via

various P2P services. Recently, with the extension of the Edutella/JXTA user communities, some new use cases of Edutella/JXTA begin to go beyond a pure P2P setting, one of which is the need for interacting Edutella/JXTA with Web services. Some typical interaction scenarios include, e.g., exposing existing Edutella/JXTA P2P services as Web services in order to enable Edutella/JXTA functionalities to be integrated into other distributed computing paradigms built on Web services, such as OGSA (Open Grid Services Architecture) [2] and .Net [6], and integrating Web services enabled content providers such as a .Net driven E-Learning "portal" into Edutella/JXTA in order to extend the reach of Edutella/JXTA beyond a pure P2P domain. These new use cases motivate us to investigate the interaction between Edutella/JXTA and Web services.

## 2. Interacting Edutella/JXTA with Web Services: Challenges

P2P and Web services are originally proposed to address different problem domains. Whereas P2P usually has a broader definition, which is generally applied to a wide range of technologies that can greatly increase the utilization of information, bandwidth and computing resources at the edge of the Internet [4], Web services are more intended to promote interoperability and extensibility among various applications, platforms and frameworks by means of externalizing and modularizing application functionalities as sets of interoperable Internet services [1][5]. As far as the current application status of P2P and Web services is concerned, they belong to two isolated realms, adopting different wire protocols, using incompatible functionality descriptions, and applying different mechanisms to manage distributed functionalities in each platform. Taking Edutella/JXTA as an example P2P platform, we have to face four sides of challenges in order to achieve its interaction with Web services.

First, Edutella/JXTA and Web services adopt different entity identification systems and also use different

mechanisms for searching and locating entities and entity functionalities. This makes it rather difficult to mutually locate entities and entity functionalities in order to achieve the direct interaction.

As a typical example of a SOA (Service Oriented Architecture), the Web services architecture is composed of three major entities: the service provider, service registrar and service requestor. These entities are clearly distinguished and interact with each other through three types of operations: the publish, find and bind. These entities and operations work in concert to provide a loosely coupled computing paradigm, whose manifestation is described in a set of standards, most notably SOAP (Simple Object Access Protocol), UDDI (Universal Description, Discovery and Integration) and WSDL (Web Services Definition Language). SOAP provides the means for communication between Web services and client applications. UDDI is used to register and publish Web services and their characteristics so that they can be found by potential client applications. WSDL is used to describe the interface of Web services. Within the Web services world, service providers, service requestors and service registrars leverage the UDDI-based centralized discovery mechanism to locate each other and further achieve various functionality interactions.

Unlike in Web services, in Edutella/JXTA there is no clear differentiation between the service provider, service requestor and service registrar. An Edutella/JXTA peer typically implements one or more of the JXTA protocols, and may play multiple roles at the same time. As all Edutella/JXTA peers share the same JXTA core implementation module, they can directly interact with each other based on the JXTA platform facilities. This implies that there exists no direct functional entity mapping between a Web services node and an Edutella/JXTA peer, and also implies that it is impossible for a Web services node to achieve the direct interaction with Edutella/JXTA without conducting the JXTA bootstrapping process in advance.

With regard to the entity and entity functionality discovery, Edutella/JXTA mainly depends on decentralized discovery mechanisms for searching and locating peers and P2P services, although it does not essentially exclude other discovery mechanisms such as the centralized ones. However, as the current decentralized discovery mechanism of Edutella/JXTA is implemented based on the JXTA peer discovery protocol [10], which is not compatible with UDDI, Edutella/JXTA P2P services cannot directly be advertised or discovered in the Web services world. Likewise, the UDDI-based centralized Web services registry cannot directly be integrated into Edutella/JXTA and used by Edutella/JXTA peers for discovering and binding Web services [13].

Second, Edutella/JXTA and Web services adopt different transport and message protocols for the

communication within each realm. This wire layer incompatibility constitutes a big obstruction to the functionality interaction between the two platforms.

At the highest abstraction level, JXTA is nothing more than a set of protocols, each of which is defined by one or more messages exchanged among participants. With the focus on decentralization and ubiquitous computing, JXTA neither mandates the transport protocol nor mandates on how the messages are specified or propagated. In contrast, although theoretically Web services do not also mandate the transport protocol, since they principally depend on SOAP for XML (eXtensible Markup Language) based messaging, they practically use HTTP (Hypertext Transfer Protocol) for transport. This implies that there might exist the transport layer incompatibility between a Web services node and an Edutella/JXTA peer, which could lead to the communication blockage during the interaction. In addition, with regard to the message protocol, although the current JXTA specification adopts XML format to define messages, the definition is not compliant with SOAP. This makes it rather difficult to achieve the direct SOAP RPC (Remote Procedure Call) between Edutella/JXTA and Web services.

Third, Edutella/JXTA and Web services adopt different formats to describe distributed services. This makes it impossible to directly exchange service descriptions between the two platforms and further interact these services based on the service descriptions.

Web services use WSDL to describe services and their invocation details, whereas Edutella/JXTA uses XML structured service advertisements to describe P2P services. In Edutella/JXTA, a P2P service is generally described by three types of JXTA advertisements: the module class advertisement, module specification advertisement and module implementation advertisement, which jointly describe the same Edutella/JXTA service at different abstraction levels [10]. Although all these abstraction levels can find their conceptual mappings in WSDL, the description formats of both are not compatible. Moreover, as Edutella/JXTA and Web services adopt different entity identification systems and also use different protocols for the service invocation, the Edutella/JXTA service advertisements cannot provide a Web services node with necessary information for locating the access point of P2P services and further invoking these services. Likewise, the SOAP RPC information provided by WSDL also makes no sense for Edutella/JXTA peers that wish to invoke Web services, as the SOAP RPC is not yet supported in Edutella/JXTA.

Fourth, although both Web services and JXTA address some common security issues such as the encryption, hashing, and authentication [8][12], their security implementations are based on different security models.

This security model incompatibility further obstructs the direct interaction between the two platforms.

The security of the Edutella/JXTA platform is implemented based on the so-called “Web of trust” security model [12]. An Edutella/JXTA peer can loan out its credentials to another peer and a community is bootstrapped by linking trusted peers. As a Web services node does not conduct any JXTA configuration, it has no chance to participate in an Edutella/JXTA community and sequentially interact with Edutella/JXTA peers and P2P services.

In overall, all four sides of challenges stem from the wire layer incompatibility between Edutella/JXTA and Web services. By defining and implementing sets of JXTA core protocols, JXTA constructs a relatively independent realm, which is not yet SOAP-aware thus excludes any direct interaction with Web services nodes that are not “JXTAalized”. Although JXTA has promised to become SOAP-aware at the wire layer through the JXTA SOAP binding [11], the ongoing efforts show that the JXTA SOAP binding might likely impact the primary JXTA specification thus still has a long way to go when the issues concerning its compatibility with other JXTA bindings are considered. Before JXTA becomes SOAP-aware at the wire layer, we need a technical approach being able to overcome the wire layer incompatibility between the two platforms in order to achieve the interaction between Edutella/JXTA and Web services.

In this paper we propose an approach for interacting Edutella/JXTA with Web services through so-called Web services/Edutella proxies. A Web services/Edutella proxy is an Edutella/JXTA peer, which also possesses a SOAP implementation module responsible for mediating the SOAP RPC between Edutella/JXTA and Web services. In order to be identified outside of Edutella/JXTA, unlike usual Edutella/JXTA peers, the Web services/Edutella proxy has to expose its IP address besides its usual JXTA ID. It actually plays a double role, being able to directly address a few of aforementioned challenges such as the entity mapping and the security management. In the following we will further elaborate how the Web services/Edutella proxies can overcome the wire layer incompatibility between Edutella/JXTA and Web services thus address other challenges left.

### 3. Service Layer Interaction between Edutella/JXTA and Web Services

Both Edutella/JXTA and Web services adopt a layering architecture to implement a SOA. In figure 1 we first illustrate the layering network architecture of the JXTA platform [4].

The JXTA platform is composed of three layers. The JXTA core layer can be viewed as the wire layer, including several building blocks for managing P2P

messaging, routing, etc. These building blocks can be used by almost all P2P applications regardless of their intended users, platforms, devices and specific implementations. The JXTA service layer builds upon the JXTA core layer, dealing with some higher-level concepts such as indexing, searching and discovering through sets of P2P services. Various Edutella/JXTA P2P services designed for implementing specific Edutella functionalities, e.g., the Edutella query service, annotation service, etc., exist at the JXTA service layer. On the top of the JXTA service layer is the JXTA application layer. Generally, there is no absolute demarcation between the JXTA service layer and application layer. Edutella itself can usually be viewed as a JXTA application that exists at the JXTA application layer.

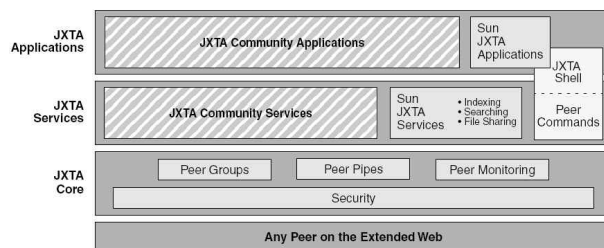


Figure 1. JXTA network architecture

As envisioned by the W3C (World Wide Web Consortium) Web services architecture working group, various interrelated technologies involved in Web services can also be investigated in terms of a layering architecture as illustrated in figure 2 [1].

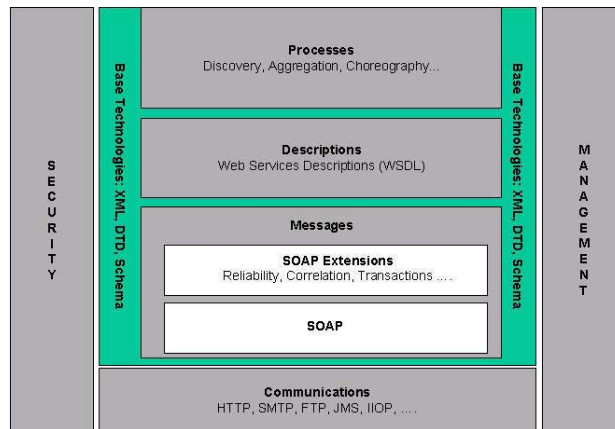


Figure 2. Web services architecture

The Web services architecture consists of four function layers. According to the SOA, these four layers can further be generalized into a wire-service-application layer model. Since a Web service can be defined as being network-accessible via SOAP and represented by a service description [1], the bottom three function layers of the Web services architecture: the communication layer,

message layer and description layer are essentially required to provide or use any Web services thus can be viewed as the interoperable basis of the whole Web services world. Among them, the communication layer and message layer can jointly be viewed as the wire layer of the Web services architecture, taking SOAP as the layer cornerstone to manage the low-level message exchanging in the platform. The description layer can be viewed as the service layer, using WSDL as the layer cornerstone to describe and expose functionalities of Web services. Additionally, the process layer of the Web services architecture can be viewed as the application layer, depending on UDDI and some other advanced Web services standards, e.g., WSFL (Web Services Flow Language), Web Services Coordination, etc., to build practical Web services applications. It is worth noting that unlike some other Web services architectures proposed by leading Web services vendors [5], W3C intentionally categorizes some UDDI-centered Web services processing functionalities such as the service publication, service discovery, etc., into the application layer instead of service layer of the Web services architecture. This is because that according to the Web services definition from W3C [1], these service processing functionalities are not mandatory for every Web services applications, e.g., the Web services discovery can also be realized through other approaches instead of the UDDI-based centralized discovery mechanism. While achieving the service layer interaction between Edutella/JXTA and Web services, we mainly focus on the WSDL-centered service description issues. As a complement, in another publication we also discuss the interaction between Edutella/JXTA's decentralized discovery mechanism and Web services' UDDI-based centralized discovery mechanism [13].

Due to the considerable wire layer incompatibility, the direct wire layer interaction between Edutella/JXTA and Web services is rather difficult to achieve before JXTA becomes SOAP-aware. In contrast, the service layer and application layer interaction are expected to be able to overcome the wire layer incompatibility thus enable the exchange of distributed functionalities between the two platforms without shaking their respective basis. As the service layer interaction is the prerequisite to the implementation of the application layer interaction, in this paper we focus on achieving the service layer interaction between Edutella/JXTA and Web services through so-called Web services/Edutella proxies. Several critical issues to the realization of the service layer interaction, e.g., how to overcome the service description incompatibility between Edutella/JXTA P2P service advertisements and WSDL, in which way to expose existing Edutella/JXTA P2P services as Web services, as well as how to mediate the service invocation between Edutella/JXTA and Web services, etc., are addressed

during the design and implementation of the Web services/Edutella proxies.

#### 4. Service Layer Interaction between Edutella/JXTA and Web Services: Technical Implementations

The service layer interaction between Edutella/JXTA and Web services may have various interaction scenarios. In terms of typical use cases of the Edutella/JXTA P2P network, we figure out two example scenarios of the service layer interaction between the two platforms.

##### 4.1. Interaction scenario 1: exposing existing Edutella/JXTA P2P services as Web services

The first Edutella/JXTA prototype implements sets of P2P services [7] including the Edutella query service, annotation service, replication service, etc. In this section we take the Edutella query service as an example to present the interaction scenario of exposing existing Edutella/JXTA P2P services as Web services. The technical implementation of this interaction scenario is depicted in figure 3.

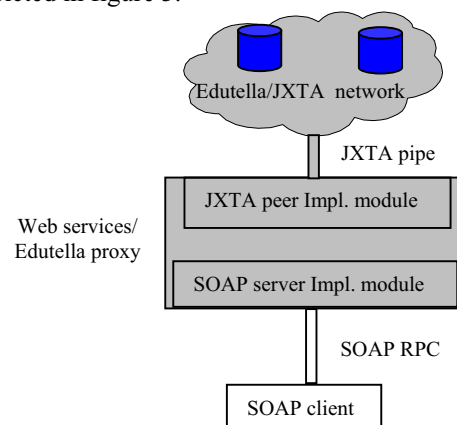


Figure 3. Exposing existing Edutella/JXTA P2P services as Web services

The Edutella query service is a major P2P service implemented for providing standardized query and retrieval of RDF metadata from all Edutella/JXTA content providers. As a standard Edutella/JXTA P2P service, it can be discovered by any Edutella/JXTA peers within the Edutella/JXTA P2P network by means of JXTA's decentralized discovery mechanism. Any Edutella/JXTA peers wishing to consume Edutella resources need to first bootstrap into Edutella/JXTA, search and discover the service advertisements, and then follow the service advertisements to open a JXTA output pipe to send a query to the Edutella query service, as well as open a JXTA input pipe to wait for the query result. In practice,

an Edutella/JXTA “consumer” peer can accomplish all these operations leveraging the Edutella consumer service initiated during its JXTA bootstrapping process. The Edutella query service advertisements can provide the Edutella consumer service with all necessary information for the service invocation.

For a Web services client outside of Edutella/JXTA, the Edutella query service is neither discoverable nor directly invocable. The key to exposing the Edutella query service to the Web services world is the use of the Web services/Edutella proxy, which, as a usual JXTA peer, initiates the Edutella consumer service that is responsible for searching, locating and interacting the Edutella query service during its JXTA bootstrapping process. Besides this JXTA implementation module, the Web services/Edutella proxy also runs a SOAP server implementation module responsible for exposing the functionality of the Edutella query service.

In general, there are two approaches for exposing the functionality of the Edutella query service to the Web services world. The first is the so-called direct wrapping approach, in which the Edutella query service has to be re-implemented as a pure Web service hosted by the SOAP server implementation module. This SOAP version of the Edutella query service would have to directly communicate with Web services clients through the SOAP RPC, manage the message exchanging between the SOAP RPC and JXTA pipes, as well as complete its original duty assigned in Edutella/JXTA, namely, interacting with all Edutella/JXTA content providers to distribute the query and gather the query result using JXTA pipes. Due to the considerable wire layer incompatibility between Edutella/JXTA and Web services, the realization of the Web services/Edutella proxy in this direct wrapping approach would have to deal with many technical complexities.

The second is the so-called broker wrapping approach, in which the Web service to expose is not the re-implementation of the Edutella query service, but instead a broker Web service that invokes the Edutella query service. This broker Web service does not need to directly interact with JXTA pipes, instead it is merely responsible for managing the message exchanging with Web services clients via the SOAP RPC, and directly leverages the Edutella consumer service to handle the interaction with the Edutella query service. In our implementation, we employ the broker wrapping approach to realize the broker Web service for exposing the functionality of the Edutella query service. This broker Web service is described by a WSDL version of the Edutella query service advertisements, as illustrated in figure 4. Hosted by the SOAP server implementation module on the Web services/Edutella proxy, this broker Web service can exist in the Web services world and undergo the publish-find-bind lifecycle just like other usual Web services. From the

perspective of Web services clients, the Web services/Edutella proxy acts as a usual Web services provider identified by its IP address. In Edutella/JXTA, however, this proxy acts as an Edutella/JXTA “consumer” peer identified by its JXTA ID.

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- generated by Glue standard 4.0b2 on Fri Apr 11 10:58:33 CEST 2003 -->
<wscdl:definitions name="EdutellaQueryWS" targetNamespace="http://www.learninglab.de/wsdl/"
xmlns:tns="http://www.learninglab.de/wsdl/" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:wscdl="http://schemas.xmlsoap.org/wsdl/" xmlns:tme="http://www.themindelectric.com/">
<wscdl:message name="getEdutellaQueryResultIn">
<wscdl:part name="Query" type="xsd:string">
<wscdl:documentation>The RDF-QEL query sent to Edutella</wscdl:documentation>
</wscdl:part>
</wscdl:message>
<wscdl:message name="getEdutellaQueryResultOut">
<wscdl:part name="Result" type="xsd:string">
<wscdl:documentation>The query results returned from Edutella</wscdl:documentation>
</wscdl:part>
</wscdl:message>
<wscdl:port type name="EdutellaQueryWS">
<wscdl:operation name="getEdutellaQueryResult" parameterOrder="Query">
<wscdl:input name="getEdutellaQueryResultIn" message="tns:getEdutellaQueryResultIn"/>
<wscdl:output name="getEdutellaQueryResultOut" message="tns:getEdutellaQueryResultOut"/>
</wscdl:operation>
</wscdl:portType>
<wscdl:binding name="EdutellaQueryWS" type="tns:EdutellaQueryWS">
<soap:binding style="rpc" transports="http://schemas.xmlsoap.org/soap/http"/>
<wscdl:operation name="getEdutellaQueryResult">
<soap:operation soap:action="getEdutellaQueryResult" style="rpc"/>
<wscdl:input name="getEdutellaQueryResultIn">
<soap:body use="encoded" namespace="http://www.learninglab.de/"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding"/>
</wscdl:input>
<wscdl:output name="getEdutellaQueryResultOut">
<soap:body use="encoded" namespace="http://www.learninglab.de/"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding"/>
</wscdl:output>
</wscdl:operation>
</wscdl:binding>
</wscdl:service name="EdutellaQueryWS">
<wscdl:documentation>The Edutella Query Web service accepts Edutella RDF-QEL query in the format of
xsd:string and returns query results as flat RDF document</wscdl:documentation>
<wscdl:port name="EdutellaQueryWS" binding="tns:EdutellaQueryWS">
<soap:address location="http://130.75.152.216:8004/glue/EdutellaQueryWS"/>
</wscdl:port>
</wscdl:service>
</wscdl:definitions>
```

Figure 4. WSDL description of the broker Web service

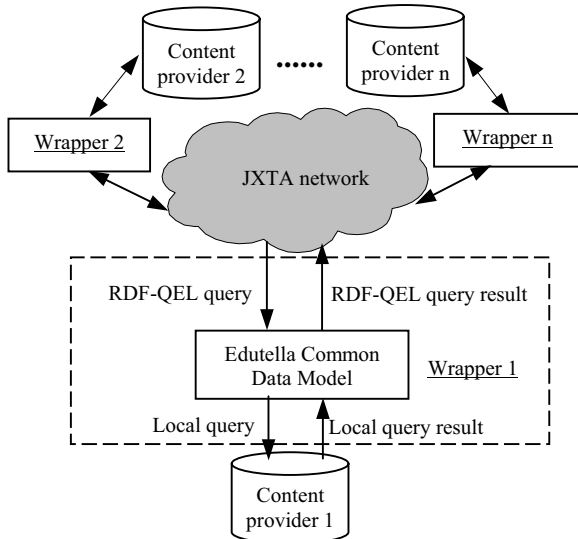
Besides the Edutella query service, all other Edutella/JXTA P2P services can also be exposed to the Web services world through the same broker Web service. In the WSDL description, each Edutella/JXTA P2P service corresponds to a WSDL “operation” element, having its own input and output message types, and possessing individual SOAP interaction styles. According to these “operation” requirements, a Web services client can easily generate code to invoke the broker Web service, which sequentially conducts corresponding operations to expose Edutella/JXTA functionalities to the Web services world.

#### 4.2. Interaction scenario 2: integrating Web services enabled content providers into Edutella/JXTA

The original Edutella/JXTA P2P network can only accommodate pure JXTA peers as content providers. As illustrated in figure 5, Edutella adopts a wrapper-like architecture to integrate heterogeneous content provider peers [7][9]. In order to be integrated into Edutella/JXTA, the Web services enabled content providers must strictly follow the Edutella content provider integration protocols to develop their Web services.

The kernel of the Edutella content provider integration architecture is the Edutella Common Data Model (ECDM)[7], which is shared by all content providers and provides the common data view of heterogeneous

metadata repositories. At its basis, ECDM is very similar to a binary relational data model defined in full compliance with the RDF data model. It uses Datalog [3] as the internal query language. Externally, Edutella/JXTA defines a common query language: RDF-QEL (RDF Query Exchange Language)[7] for the whole Edutella/JXTA network to represent the query and query result in the RDF syntax. The RDF-QEL query and query result are transferred within Edutella/JXTA in the form of serialized RDF/XML documents using JXTA pipes.



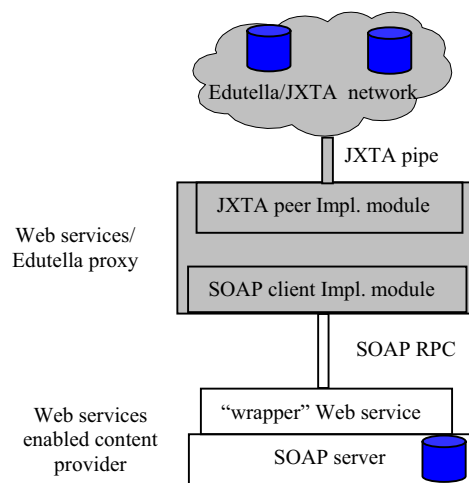
**Figure 5. Edutella content provider integration architecture**

In order to be integrated into Edutella/JXTA, a Web services enabled content provider must develop its Web service in the way that the Web service can accomplish the Edutella wrapper functionality. This Web service, which can also be viewed as the SOAP-based wrapper program implementation, must take the RDF-QEL query as the SOAP RPC message input and return the RDF-QEL query result as the SOAP RPC message output. In addition, this wrapper Web service has also to accomplish the concrete Edutella wrapper functionality. It has first to translate the RDF-QEL query into the local query of the underlying content repository, and then transform the local query result into the RDF-QEL query result. Corresponding to different types of content repositories, although the functional implementations of each wrapper Web services may be somewhat distinct, they use rather similar WSDL descriptions sharing almost all major description elements except the element of the SOAP server address. This feature can be seen from an example WSDL description of a Web services enabled content provider, as illustrated in figure 6. During the implementation of the Web services/Edutella proxy, this feature enables us to use a unique SOAP client implementation module to bind to different Web services enabled content providers.

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- generated by IBM Professional 4.1.2 on Sat Jul 05 14:11:05 CEST 2003 -->
<wsi:definitions name="EdutellaProviderWS" targetNamespace="http://www.learninglab.de/wsdl/"
xmlns:tns="http://www.learninglab.de/wsdl/" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/" xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/" xmlns:wsi="http://schemas.xmlsoap.org/wsdl/"
xmlns:tms="http://www.thomsoninfotelec.com/" />
<wsi:message name="getEduResultSetIn">
<wsi:part name="qel" type="xsd:base64Binary" />
</wsi:message>
<wsi:message name="getEduResultSetOut">
<wsi:part name="Result" type="xsd:base64Binary" />
</wsi:message>
</wsi:portType>
<wsi:binding name="EdutellaProviderWS" />
<wsi:operation name="getEduResultSet" parameterOrder="qel">
<wsi:input name="getEduResultSetIn" message="tns:getEduResultSetIn" />
<wsi:output name="getEduResultSetOut" message="tns:getEduResultSetOut" />
</wsi:operation>
</wsi:portType>
<wsi:binding name="EdutellaProviderWS" type="tns:EdutellaProviderWS">
<soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http" />
<ctm:optimizations tags="1" href="1" env="1" />
<wsi:operation name="getEduResultSet">
<soap:operation soapAction="getEduResultSet" style="rpc" />
<wsi:input name="getEduResultSetIn">
<soap:body use="encoded" namespace="http://www.learninglab.de/"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
</wsi:input>
<wsi:output name="getEduResultSetOut">
<soap:body use="encoded" namespace="http://www.learninglab.de/"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
</wsi:output>
</wsi:operation>
</wsi:binding>
<wsi:service name="EdutellaProviderWS">
<wsi:port name="EdutellaProviderWS" binding="tns:EdutellaProviderWS">
<soap:address location="http://130.75.152.216:8004/glue/EdutellaProviderWS" />
</wsi:port>
</wsi:service>
</wsi:definitions>
```

**Figure 6. Example WSDL description of a Web services enabled content provider**

For the Web services enabled content providers, developing the wrapper Web service is only the prerequisite to the integration. Although the wrapper Web service is described in WSDL and can also be discovered within Edutella/JXTA through our P2P-based Web services registry network [13], it cannot directly be invoked within Edutella/JXTA, since the current Edutella/JXTA is not yet SOAP-aware thus cannot deliver the RDF-QEL query to the Web services enabled content providers and sequentially gather the query result using the SOAP RPC. The key to invoking the wrapper Web service is the use of the Web services/Edutella proxy, which, as a usual JXTA peer, also runs a SOAP client implementation module responsible for finding and dynamically binding the wrapper Web service. In figure 7 we illustrate the technical implementation of this interaction scenario.



**Figure 7. Integrating Web services enabled content providers into Edutella/JXTA**

The Web services/Edutella proxy actually acts as the representative of the Web services enabled content provider in the Edutella/JXTA P2P network. Like usual Edutella/JXTA content provider peers, it initiates the Edutella provider service and publicizes corresponding service advertisements during its JXTA bootstrapping process. These service advertisements declare the existence of the Web services/Edutella proxy (actually the existence of the Web services enabled content provider), and also describe how the proxy can be accessed by the Edutella query service through JXTA pipes.

The Edutella provider service is responsible for opening a JXTA input pipe for accepting the RDF-QEL query from Edutella/JXTA, as well as opening a JXTA output pipe for sending back the query result. In contrast to the usual Edutella provider service, which directly forwards the RDF-QEL query coming from the JXTA input pipe to the underlying content repositories and directly returns the RDF-QEL query result to the JXTA output pipe, the Edutella provider service running on the Web services/Edutella proxy needs a little bit of improvement to deal with the SOAP RPC based message exchanging between Edutella/JXTA and the Web services enabled content providers. In the new implementation of the Edutella provider service, we interpose a SOAP client implementation module into the usual message transferring process. The RDF-QEL query coming from the JXTA input pipe is first transferred to the SOAP client implementation module, which then dynamically binds the wrapper Web service and forwards the RDF-QEL query to the Web services enabled content providers through the SOAP RPC. After the query result is returned from the Web services enabled content providers, this SOAP client implementation module is responsible for unmarshalling the query result from the SOAP RPC and then transferring the result back to the JXTA output pipe. From the perspective of the Web services enabled content providers, they are not aware that their resources are being utilized in the context of a P2P setting. They exist in the Web services world and do not need to conduct any P2P configurations.

Via the Web services/Edutella proxy, the Web services enabled content providers are “virtually” included into Edutella/JXTA’s searching scope. Their resources can then be accessed by any Edutella/JXTA peers. In addition, as described in the interaction scenario 1, these resources can also be accessed by usual Web services clients mediated by the Edutella/JXTA P2P network. Note should be paid that in this accessing scenario, the resources provided by the Web services enabled content providers are delivered as part of the Edutella/JXTA output, following the common delivery protocols defined by Edutella/JXTA. In contrast, if these resources are directly

accessed in the Web services world, they are individually delivered, not being embedded in the Edutella/JXTA resource network.

## 5. Conclusions

From the basis, P2P and Web services are complementary technologies. While Web services provide the universal information architecture to solve the functional integration problem, P2P provides the distributed network architecture that takes us directly and securely to the sources of information at the edge of the Internet. The interaction between the two platforms may potentially combine advantages of both, leading to a promising distributed computing architecture converging P2P and Web services.

## 6. References

- [1] Booth, D., M. Champion, C. Ferris, F. McCabe, E. Newcomer, and D. Orchard, Web Services Architecture, <http://www.w3.org/TR/ws-arch/>
- [2] Foster, I., C. Kesselman, J. M. Nick, and S. Tuecke1, The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration, <http://www.globus.org/ogsa/>
- [3] Garcia-Molina, H., J. D. Ullman, and J. Widom, Database Systems: The Complete Book, Prentice Hall, USA, 2001.
- [4] Gong, L., Project JXTA: A Technology Overview, <http://www.jxta.org/>
- [5] Kreger, H., Web Services Conceptual Architecture (WSCA 1.0), <http://www-3.ibm.com/software/solutions/webservices/pdf/WSCA.pdf>
- [6] Microsoft Corp., Microsoft .NET, <http://www.microsoft.com/net/>
- [7] Nejd1, W., B. Wolf, C. Qu, S. Decker, M. Stintek, A. Naeve, M. Nilsson, M. Palmer, and T. Risch, Edutella: A P2P Networking Infrastructure Based on RDF, in Proc. of the 11<sup>th</sup> International World Wide Web Conference (WWW 2002), Hawaii, USA, May 2002.
- [8] O’Neill, M., Web Services Security, McGraw-Hill Osborne Media, USA, 2003.
- [9] Qu, C., W. Nejd1, and H. Schinzel, Integrating Schema-specific Native XML Repositories into a RDF-based E-Learning P2P Network, in Proc. of the 2<sup>nd</sup> International Conference on Dublin Core and Metadata Applications (DC 2002), Florence, Italy, Oct. 2002.
- [10] Sun Microsystems Inc., JXTA v2.0 Protocols Specification, <http://www.jxta.org/>
- [11] Sun Microsystems Inc., JXTA SOAP Binding, <http://soap.jxta.org/>
- [12] Sun Microsystems Inc., Security and Project JXTA, <http://www.jxta.org/>
- [13] Thaden, U., W. Siberski, and W. Nejd1, A Semantic Web based Peer-to-Peer Service Registry Network, Technical Report, Learning Lab Lower Saxony, 2003.