A Semantic Web Service Based Middleware for the Tourism Industry

Yildiray Kabak, Mehmet Olduz, Gokce B. Laleci, Tuncay Namli, Veli Bicer, Nikola Radic and Asuman Dogac

***Abstract*— Currently in the travel domain most of the travel products are sold through Global Distribution Systems (GDSs). Since only major airline companies or hotel chains can afford to join GDSs, it is difficult for small and medium enterprises to market their travel products. In this paper, we describe a middleware, called SATINE, to address this problem.**

In the SATINE middleware, existing travel applications are wrapped as Web services. Web services, as such, is of limited use because the service consumer must know all the details of the Web service like the functionality of the Web service (what it does) and the content and the structure of input and output messages. Therefore we annotate both the service functionality and the service messages with Web Ontology Language (OWL) ontologies. Service functionality ontology is obtained from the "Open Travel Alliance (OTA)" specifications. Service message ontologies are automatically generated from the XML Schema definitions of the messages. These local message ontologies are mapped into one or more global message ontologies through an ontology mapping tool developed, called OWLmt. The mapping definitions thus obtained are used to automatically map heterogeneous message instances used by the Web service provider and the consumer using a global ontology as a common denominator. This architecture is complemented by a peer-to-peer network which uses the introduced semantics for the discovery of Web services.

Through the SATINE middleware, the travel parties can expose their existing applications as semantic Web services either to their Web site or to Web service registries they maintain. SATINE middleware facilitates the discovery and execution of these services seamlessly to the user.

I. Introduction

THE tourism industry today is the second largest economic sector, after manufacturing in the world. Tourism industry embarked on e-Business earlier than in other sectors as evident in several online travel e-Commerce sites.

Currently, travel information services are dominantly provided by Global Distribution Systems (GDSs) such as Galileo [12], Sabre [31] and Amadeus [1]. Major airline companies, many hotel chains and car rental companies list their inventories with major GDSs. A GDS gives its subscribers pricing and availability information for multiple travel products such as flights, hotel rooms and car rentals. Travel agents, corporate travel departments, and even Internet travel services, subscribe to one or more GDSs. However, small and medium-sized enterprises cannot participate to GDS-based e-Business activities since selling their products through GDSs is too expensive for them. Furthermore, GDSs are legacy systems that mostly rely on private networks. They are mainly for human use and have difficult to use cryptic interfaces, have limited search capabilities, and are difficult to inter-operate with other systems and data sources.

In order to facilitate eBusiness, the travel industry has formed a consortium called the Open Travel Alliance (OTA) [23], and OTA has produced XML schemas of message specifications to be exchanged between the trading partners, including availability checking, booking, rental, reservation, query services, and insurance. However, not all travel applications can be expected to produce and consume OTA compliant messages.

In this paper, we describe a middleware to facilitate eBusiness for all the involved parties in the travel domain which has been implemented within the scope of the SATINE Project [32]. The main idea is to expose existing travel applications as Web services and facilitate the discovery of and execution of Web services through semantic mediation and peer-to-peer (P2P) networks. The use of P2P technology facilitates the discovery of the services of small and medium size enterprises (SMEs) to enable them to easily sell their services over the Internet. Web service technology together with travel domain specific ontologies allow the parties consume heterogeneous messages.

The technology of the middleware is as follows:

- The existing travel domain applications are wrapped as Web services. Web services, as such, is of limited use because the service consumer must know all the details of the Web service like the functionality of the Web service (what it does) and the content and the structure of input and output messages. Although there are efforts to standardize the messages exchanged in the travel domain such as Open Travel Alliance, not every travel application can be OTA compliant. Furthermore, the GDSs must be a part of the system since today an important part of the travel information comes through them. Therefore, we handle the interoperability of Web service messages through semantic mediation.
- The main mechanism to discover Web services is the Web service registries. We show how to use the semantic annotation of Web services in the UDDI and ebXML registries to facilitate Web service discovery.
- There could be Web services not registered to any service registry but simply made available through a Web site. This may be preferable especially by SMEs which may not wish to have extra overhead of maintaining Web service registries. To facilitate the discovery of Web services, SATINE middleware uses peer-to-peer network

This work is supported by the European Commission, DG Information Society and Media, eBusiness Unit through IST-1-002104-STP SATINE project and in part by the Scientific and Technical Research Council of Turkey, Project No: EEEAG 102E035

which exploits the defined semantics. We show how to handle semantic routing for Web service discovery and how to invoke Web services over the P2P networks.
- In the SATINE middleware, the technical details are hidden behind user friendly graphical interfaces. This is essential to facilitate the usage of the system and to help with its take-up.

The paper is organized as follows: Section II gives an over all view of the system architecture. In Section III, the semantic infrastructure of SATINE is elaborated including the functionality ontologies and the message ontologies together with their utilization for semantic annotation of the Web services. The semantic mediation in SATINE is described in Section IV by providing the details of the ontology mapping and the normalization components. Section V covers the SATINE semantic P2P Network infrastructure. The role of service registries and how they are semantically enriched are given in Section VI and secure service invocation is summarized in Section VII. In Section VIII, the performance results of the system prototype are described. The related work is presented in Section IX. Finally, Section X concludes the paper.

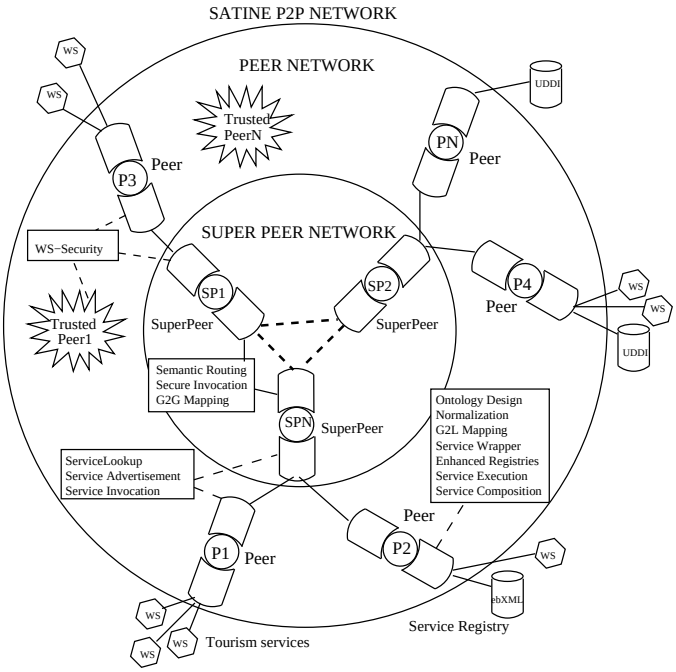

Fig. 1. The Architecture of the Satine Middleware

II. System Architecture Overview

The overall architecture of the SATINE middleware is presented in Figure 1. The Web service providers and service requesters are represented as “edge peers” in the “Peer Network”. The parties can advertise their existing tourism Web services to the “SATINE P2P network” as well as the new Web services which are created by wrapping existing tourism applications including the services of the GDSs. These services can either be hosted by the edge peers themselves, or in a service registry connected to the rest of the P2P network through the edge peers. In this way, the outmost layer of “P2P Network” contain the set of wrapped sources which are either the service registries or Web Services themselves.

SATINE uses super peer - peer architecture. It has been observed that P2P networks can lead to an efficient network architecture when a small subset of peers, called super-peers, takes over specific responsibilities for peer aggregation, query routing, and mediation. Super-peer based P2P infrastructures are usually based on a two-phase routing architecture, which routes queries first in the super-peer backbone, and then distributes them to the peers connected to the super-peers [18].

In the SATINE middleware, the super peers store the global semantic knowledge, that is, one or more global message ontologies and their mappings to local ontologies which are then used for semantic mediation of the messages exchanged. Peers interact with their super peers to advertise their services and also for Web service discovery and invocation. Furthermore, the super peers facilitate the semantic routing of messages between the edge peers by the use of indices.

In order to facilitate the discovery of the advertised Web services in the P2P network, we annotate the functionality of Web services through SATINE Tourism functionality ontologies. The SATINE Semantic Infrastructure is elaborated in detail in Section III.

Another characteristic of SATINE P2P architecture is the semantic mediation of exchanged messages between the parties involved. There are many different message formats and contents in the travel domain and if the sender and the consumer use different messages, they need to be semantically mediated to be of any use at the receiving end. For example, one may wish to invoke a Web service to reserve a flight by using Amadeus GDS [1] but the consumer of this Web service may not be aware of the content of Amadeus messages.

SATINE allows automatic creation of local message ontologies in Web Ontology Language (OWL) [25] from the XML message schemas that the parties are already using. These local message ontologies are manually mapped to the global ontology with the help of SATINE ontology mapping tool, namely, OWLmt [28]. Once this mapping is defined, message instances are automatically converted one into other by using the mapping definition produced. SATINE Semantic mediation component is detailed in Section IV.

Annotating the services through ontologies enables us to query the P2P network semantically to locate the services requested. The SATINE peer-to-peer network is enhanced with indices for semantically publishing and discovering Web services. In order to increase scalability, the queries are routed only to the peers that are hosting the requested services based on the semantics presented in the query message. The SATINE P2P architecture is elaborated in detail in Section V.

In SATINE, the tourism service provider peers may introduce service registries to the SATINE P2P network to publish their Web services. To enable semantic service discovery in SATINE P2P network, semantically enriched UDDI and ebXML service registries are integrated to the SATINE architecture as detailed in Section VI.

In the SATINE middleware the invocation of travel Web

services is performed based on WS-Security specification [44]. For this purpose "trusted peers" are introduced to the SATINE architecture to distribute the public keys of the peers and create session keys for each invocation transaction. SATINE secure Web service invocation is summarized in Section VII.

III. Semantic Infrastructure

In the SATINE middleware, the domain specific semantics is necessary for the Web services in the following respects:

- *For describing service functionality semantics*: In order to facilitate the discovery of the Web services, there is a need for semantics to describe what the service does. In other words, the service requestor should be able to locate a Web service which meets his needs in terms of its functionality. Note that, the naming of services performing the same kind of operation varies for different providers and languages.
  In SATINE, we have utilized OTA [23] specifications to construct a Service Functionality Ontology. OTA specifications expose significant domain knowledge in developing a global Web service functionality ontology since the OTA request/response pairs can be arranged into a class hierarchy to define operation semantics of travel Web services. An example OTA compliant travel functionality ontology is presented in Figure 2. Using this ontology, not only the functionality of a service (i.e. "AirAvailabilityServices") but also some of the service properties bounded to that functionality (i.e. "Quality of Service") can be specified.

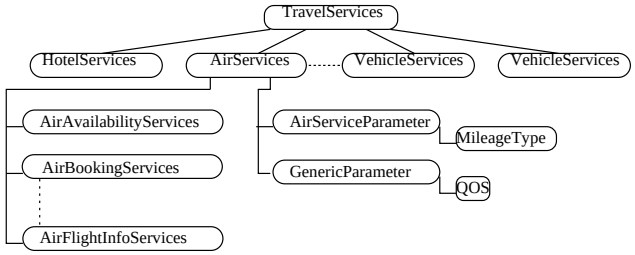

Fig. 2. An example OTA complaint Functionality Ontology

- Service functionality semantics enables us to discover the Web services based on their semantics. However in order to execute the discovered Web services, message level interoperability is necessary. Service functionality semantics may suffice to achieve interoperability only when all the Web services use the same message standards. However, it is not realistic to expect all the travel service providers to comply with the same message structure and content. Hence, there is a need to transform one message content and structure into another. In order to facilitate message transformation, SATINE utilizes semantic mediation.
  Currently the tourism application messages are usually in XML. In the SATINE middleware, the travel service providers and requesters express the semantics of the Web service messages through local ontologies in OWL. However it may not be straight forward for travel service providers to create Web services exchanging OWL instances. Hence there is a need for automatic bidirectional transformation of XML message instances to OWL message instances. To be able to realize this, automatic generation of OWL Schemas from XML Schema Definitions (XSDs) is needed. Such transformations, called Normalization, have been realized within the scope of the Harmonise project [13]. This normalization tool is enhanced to cover OWL concepts instead of RDFS in SATINE infrastructure.
  The local message ontologies generated from XSD schemas are mapped into the Global message ontology. Currently, SATINE uses more than one global message ontology one of which is based on the OTA specifications. It should be noted that our aim is not to propose ontologies for travel domain but to show how such ontologies, once developed, can be used for the semantic interoperability.

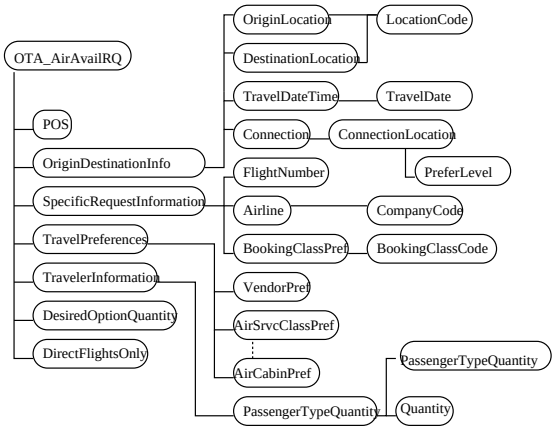

Fig. 3. An example OTA complaint Message Ontology part for AirAvailability Request

A. Annotating Web Service Functionality and Messages through Ontologies

Each Web service provider is expected to map their local semantics to a global ontology once at the conceptual level with the help of the semantic mapping tool SATINE provides. The automatic translation of the messages at the instance level is performed through the ontology mapping component presented in Section IV. A part of an example OTA message ontology for the input messages of Air Availability services is given in Figure 3.

In the SATINE middleware, for semantic description of Web services the "Web Ontology Language for Services (OWL-S)" [26] is used. OWL-S provides an upper ontology which defines a top level "Service" class with some generic properties common to most of the services. The "Service" class has the following three properties:

- *presents*: The range of this property is ServiceProfile class. That is, the class Service presents a ServiceProfile

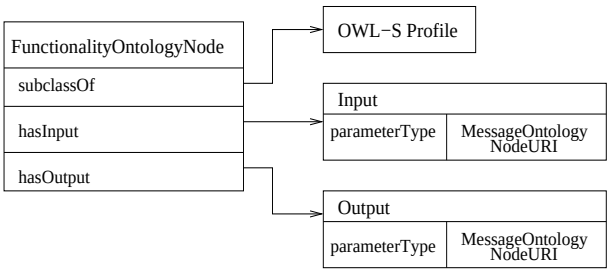

Fig. 4. Relating Service Functionality and Service Message Ontologies with OWL-S

to specify what the service provides for its users as well as what the service requires from its users.

- *describedBy*: The range of this property is ServiceModel class. That is, the class Service is describedBy a ServiceModel to specify how it works.
- *supports*: The range of this property is ServiceGrounding. That is, the class Service supports a ServiceGrounding to specify how it is used.

Based on the service functionality and service message ontologies, Web services are annotated in SATINE architecture through OWL-S as depicted in Figure 4:

- Each node in the functionality ontology is created as a subclass of OWL-S Profile class. For each Web service, an OWL-S definition is created as an instance of the related functionality ontology node. In this way, the functionality of the Web services is annotated through the service functionality ontology.
- In the OWL-S, Profile class has properties called "hasInput" and "hasOutput" whose ranges are "Input" and "Output" classes. These classes, in return have a property, namely, "parameterType". The value of this property is set to a node in the SATINE local message ontology. In this way the service's input and output parameters are annotated with the service message ontologies.

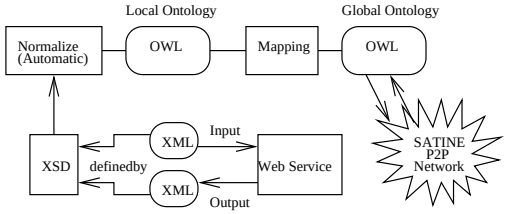

Fig. 5. Normalization Process in SATINE Architecture

SATINE provides graphical user interfaces (GUI) to guide a user to create semantic annotation of the Web services. The GUI tool produces semantic definitions in terms of OWL-S files automatically where the message semantics are annotated with the "local message ontologies".

IV. Semantic Mediation

SATINE architecture enables the semantic mediation through an OWL ontology mapping component, namely, OWLmt. OWLmt allows the design of mappings between two ontologies and the automatic translation of the OWL instances based on the mapping definitions. However, since most of the Web services exchange XML messages defined by an XML Schema Definition (XSD) [46], a normalization tool is integrated to the SATINE system. The normalization tool enables the conversion of OWL instances to XML messages and vice versa as well as the automatic creation of local message ontologies from XSD schemas.

In the following sections, the normalization and the ontology mapping processes are detailed emphasizing how they are utilized in the SATINE architecture.

A. Normalization

Normalization tool enables the creation of OWL ontologies from a set of XML Schema Definitions (XSDs) of the services. This is very suitable for the creation of the local ontologies of travel service providers. Once a travel service provider has the XSDs of its services, the only process needed in order to communicate through SATINE network is the creation of local OWL ontologies from these schemas automatically, and the definition of the mappings between this local ontology and one of the global ontologies available in the system. The integration of the normalization process in SATINE infrastructure is presented in Figure 5.

In this way, it becomes possible for tourism Web services to consume XML messages in SATINE architecture. Whenever a local ontology is automatically created from an XSD file, a normalization map is created automatically and stored in the peer. This normalization map is used by the Normalization tool to convert the OWL instance in local ontology provided as an input to the Web service to an XML message. The output of the Web service provided as an XML message is denormalized to OWL and continues its journey in the SATINE P2P network as presented in Section IV-B.

B. Ontology Mapping

Ontology Mapping is the process where two ontologies with an overlapping content are related at the conceptual level, and the source ontology instances are transformed into the target ontology instances according to those relations. In the SATINE architecture, an OWL mapping tool is developed, called OWLmt [28] which is used to handle ontology mediation. OWLmt provides two key components which are the mapping GUI and the mapping engine as shown in Figure 6.

The mapping GUI provides a user-friendly environment, which enables the designer to define the relations among the similar entities based on the overlapping content in the source and target ontologies. As a result of this "matching" process, the "mapping definition" is constructed which stores the relations between the two ontologies. The representation of the matching between the entities of the two ontologies and all the information related to it are defined as mapping patterns in the mapping definition. Mapping definition is also represented in OWL. Mapping patterns mainly involve the following:

- *Matching the source ontology classes to target ontology classes*: Four mapping patterns namely, "EquivalentTo", "SimilarTo", "IntersectionOf" and "UnionOf" are used to

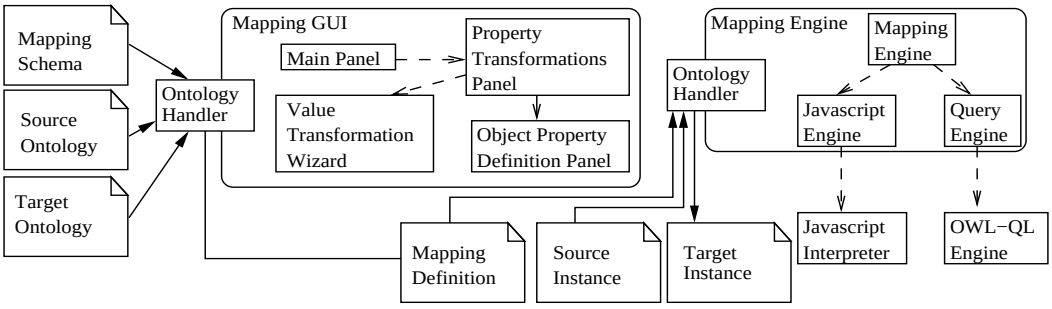

Fig. 6. Architecture of OWLmt

represent the matching between the classes of source and target ontologies.

- *Matching the source ontology Object Properties to target ontology Object Properties*: "ObjectPropertyTransform" pattern is used to represent the path of classes connected with object properties in order to map one or more object properties in the source ontology with one or more object properties in the target ontology.
- *Matching the source ontology Data Properties to target ontology Data Properties*: Datatype properties of an instance in the source ontology are transformed to the target ontology instance datatype properties by specifying a "DatatypePropertyTransform".

OWLmt provides a powerful translation mechanism for the datatype properties, because the datatype properties may be structurally different in source and target ontologies. As a result, more complex transformation operations may be necessary than copying the data in the source instance to the target instance. XPath specification [45] defines a set of basic operators and functions which are used by the OWLmt such as "copy", "concat", "split", "substring", "abs", and "floor". However, in some cases, there is a further need for a programmatic approach to specify complex functions. For example, the use of conditional branches (e.g. if-then-else, switch-case) or iterations (e.g. while, for-next) may be necessary in specifying the transformation functions. To handle this kind of data transformation, OWLmt is supported by JavaScript statements. By specifying the JavaScript to be used in the "DatatypePropertyTransform" pattern, the complex functions can also be applied to the data as well as the basic functions and the operators. Similarly, the data transformation can also be performed by invoking a Web service by providing its WSDL. The Web service support is important for the use of data dictionaries to lookup the corresponding data values.

In Figure 8, an example mapping definition between OTA global ontology (Figure 3) and a local ontology for AirAvailability request message is presented. In this example, the Javascript shown in Figure 7 is used to transform "AirCabinPref" Datatype property values of "First", "Business" or "Economy" to "ServiceClass" Datatype property values of "1", "2" and "3".

As an example to using Web services for DatatypeProperty transformation, a Web service may provide the code of the airport once the IATACode for a city is provided (Figure 8).

```
if(x==null){
    return "";}
var str=new java.lang.String(x);
if(str.equals("First")){
    return "1";
}else if(str.equals("Business")){
    return "2";
}else if(str.equals("Economy")){
    return "3";
}else{  return "";}}
```

Fig. 7. An Example Javascript for DatatypeProperty Transformation

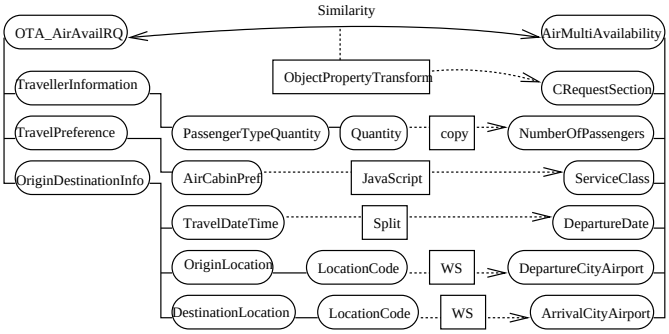

Fig. 8. Example Mapping for AirAvailability Request message

When a tourism organization wishes to register itself to the SATINE P2P Network, two mapping definitions need to be defined for its local message ontology: local-to-global and global-to-local. This is necessary because the communication language in the Super Peer Network (Figure 1) is the global ontologies. Whenever a peer sends a message to its super-peer it has to be translated to the global ontology from the local ontology, and whenever a super-peer sends a message to its peer, it is translated to the local ontology of the peer. Each edge peer provides the OWLmt mapping GUI to the user to define these mapping definitions and these are stored in the peer and its super peer.

The mapping engine is responsible for creating the target ontology instances using the mapping patterns stored in the mapping definitions and the instances of the source ontology. It uses OWL Query Language (OWL-QL) [24] to retrieve required data from the source ontology instances. The mapping engine first executes the class mapping patterns. Then, the property mapping patterns are executed. Similar to the class mapping patterns, OWL-QL queries are used to locate the data.

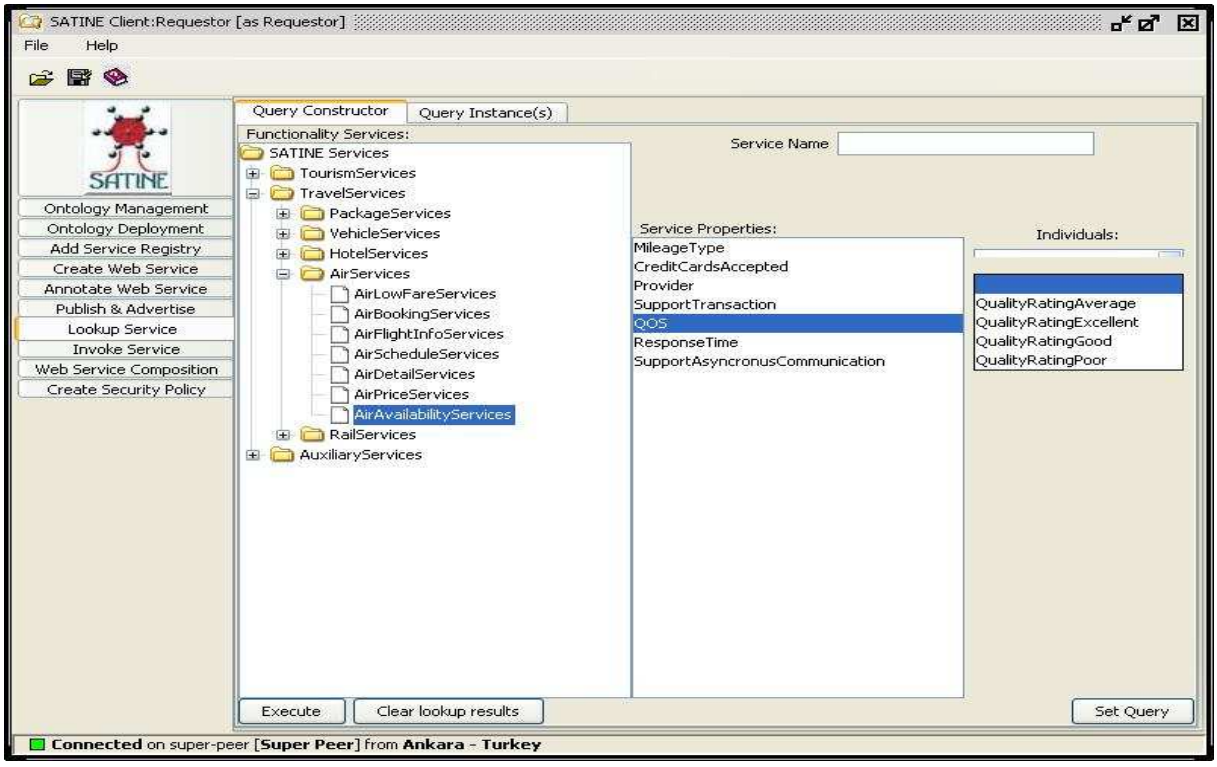

Fig. 9. SATINE Service Lookup Interface

The use of the OWL-QL enables OWLmt to have reasoning capabilities. In order to perform value transformations, the mapping engine uses the JavaScripts or Web services in the "DatatypePropertyTransform" pattern.

Once the mapping definitions are prepared and stored in peers and its super-peer; the mapping engine works seamlessly to the user at the service invocation phase. In order to enable semantic interoperability, the travel services are not invoked directly by the peer; the service requester sends a service invocation request to its super peer. The service requestor does not have to know the local semantics of the services provided by different parties. Hence, the input message is translated to the global semantics automatically by the peer using the mapping engine and sent to the super-peer of the service requester. Once the service provider gets the input message from its super-peer, it is translated to the local semantics of the service provider automatically and given as an input to the service to be invoked. The invocation result is also mediated to the service requestor in the same manner.

V. SATINE P2P NETWORK

One of the major aims of developing a semantically enriched P2P network infrastructure for SATINE is to provide a scalable distributed architecture for the currently centralized service discovery and invocation mechanisms in the tourism domain.

SATINE P2P architecture is implemented based on the JXTA platform [21]. JXTA is an Open Source project supported and managed by Sun Microsystems. JXTA provides discovery mechanisms for the resources owned by the peers. SATINE Peer-to-Peer architecture has enhanced JXTA capabilities for publishing, discovering and invoking semantically enriched Web services. It facilitates the discovery of Web services both from the individual peers in the SATINE network and also from the public service registries that are a part of the P2P Network. In SATINE, the edge peers can wrap UDDI and ebXML service registries and provide functionalities for publishing semantically enriched web services to service registries and semantic querying of the service registries as presented in Section VI. This facilitates the semantic discovery of web services from several service registries in a P2P environment.

A. Semantic Service Advertisement in SATINE P2P Network

SATINE provides user interfaces which guide the user to create semantic annotation of the Web services. As presented in Section III, in SATINE, Web service's semantic definition is represented as an OWL-S file where the functionality ontology nodes are referred for annotating service functionality, and message ontology nodes are referred for annotating the semantics of service parameters.

Each SATINE edge peer provides a graphical interface to the user for advertising its services to the SATINE P2P network. In order to advertise a Web service to the system, the technical description (WSDL), semantic description (OWL-S) and the security policy of a service is provided to the system through the "Service Advertisement" component. While advertising the Web service, the user also indicates whether he wishes the Web service to be published to a service registry. In this case, the access information for the Web service registries is also provided. The edge peer processes this information, and

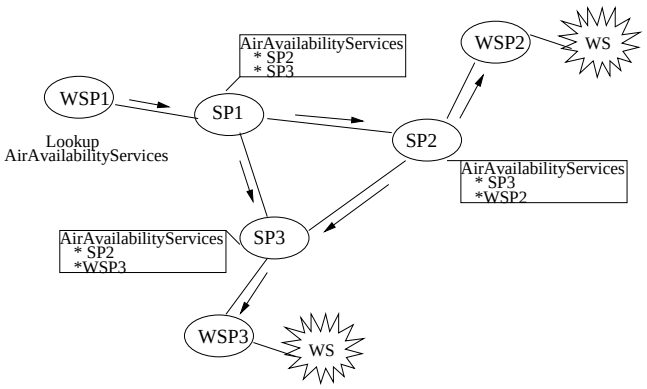

Fig. 10. Super Peer Semantic Routing indexes

if a registry is selected, the service is semantically published to the selected registry seamlessly to the user using the methods described in Section VI.

To enable semantic message routing and semantic query mechanisms based on service advertisements, the service semantics are processed by the P2P network accordingly. Hence a service advertisement triggers the interaction of the edge peer with the SATINE super peer network: the peer sends the OWL-S file of the service to its super peer. SATINE super peer network handles the semantic routing of the queries to the appropriate peers by establishing routing indices based on the semantics of the services advertised. First, for a given peer its super peer processes the semantic definition of the service, and updates its indices to indicate that the peer has a service with the specified semantics. Then the super peer informs other super peers about the semantics of the web service advertised, so that the queries can be correctly routed.

B. Semantic Service Discovery in SATINE P2P Network

In order to handle the semantic querying, SATINE provides each peer a query formulation and a query interpretation component, namely "Service Lookup" component as presented in Figure 9. The query formulation tool parses the travel functionality ontology and presents it through a graphical interface to the user for specifying the query parameters so that a user can easily search for the services he is looking for. As presented in Figure 9, on the left hand side panel, the functionality ontology nodes are presented. Whenever the user selects one of the nodes in this panel, the properties of this functionality ontology node and its possible values are presented on the right hand side panel. Based on the user input, the OWL-QL query is constructed at the backend. For example, if the user selects "AirAvailabilityServices" as a functionality ontology node, and "QualityRatingExcellent" as the value of "QOS" property, the OWL-QL query presented in Figure 11 is generated automatically by the SATINE Lookup interface.

Afterwards, the query is sent to the SATINE P2P network. The super-peers semantically route the query only to the related peers by checking their indices. While the query is being routed in the SATINE P2P network, the query is mapped from one global ontology to another whenever necessary

```
(and (|http://www.w3.org/1999/02/22-rdf-syntax-ns#|:type ?x
|http://144.122.230.177:8080/satine/TravelFunctionality
Ontology.owl#|:|AirAvailabilityServices|)
(|http://www.w3.org/1999/02/22-rdf-syntax-ns#|:type ?p
|http://144.122.230.177:8080/satine/TravelFunctionality
Ontology.owl#|:|QOS|)
(|http://144.122.230.177:8080/satine/Profile.owl#
|:|sParameter| ?p
|http://144.122.230.177:8080/satine/TravelFunctionality
Ontology.owl#|:|QualityRatingExcellent|))
```

Fig. 11. An example OWL-QL query generated by SATINE Lookup interface

based on the mapping definitions available at the super peers. For example, in Figure 10, Web service Peer 1 (WSP1) issues a query to find a service with the functionality of "AirAvailabilityServices". The service lookup query is sent to the super peer of WSP1, which is Super Peer 1 (SP1). SP1 resolves the query and checks its indices. As a result, it discovers that there exists a path to a service provider with "AirAvalabilityServices" from SP2 and SP3 routes. Then, the query is forwarded to the corresponding super peers. Note that by such semantic routing, the flooding of the messages is highly reduced. Additionally, in the topology presented in Figure 10, the lookup messages may go in a loop between SP2 and SP3 since their indices points at each other. This looping is prevented in the SATINE architecture by attaching the route that the query passed to the message.

When the lookup message arrives at the service provider, the OWL-QL query is executed on the semantic definitions of its services. If it is an edge peer wrapping a service registry, the mechanisms for semantically querying the UDDI and ebXML registries, which are elaborated in Section VI, are exploited automatically. All the available "AirAvailabilityServices" found in the SATINE network are accumulated in the service requestor and displayed to the user according to the service name, service provider and service description so that the user can continue with invoking the preferred ones.

VI. Enhanced Service Registries

The SATINE System supports both ebXML [7] and the UDDI [38] registries, to store and discover semantically enriched travel Web services. In order to exploit the service registries in SATINE middleware, a number of mechanisms are developed to relate the semantics of the Web services with the registries.

The Web services are discovered in the SATINE network based on their functional properties which are specified according to a functionality ontology. Hence, the functionality ontologies have to be stored in the service registries to query the services published to the registry. After storing the functionality ontologies to service registries, it is necessary to relate the services advertised in the registry with the semantics defined through the ontology.

UDDI registries do not provide a mechanism to store an ontology internal to the registry. The mechanism to relate semantics with services advertised in the UDDI registries are the tModel keys and the category bags of registry entries. tModel keys provide the ability to describe compliance with taxonomies, ontologies or controlled vocabularies. Therefore

if tModel keys are assigned to the nodes of the ontology and if the corresponding tModel keys are put in the category bags of the services, it is possible to locate services conforming to the semantic given in a particular node of the ontology. This issue is elaborated in [2] and [3], where the mechanisms are introduced to store and relate DAML-S (earlier version of OWL-S) ontologies with services advertised in the UDDI registries.

An ebXML registry, on the other hand, allows to define semantics basically through two mechanisms: first, it allows properties of registry objects to be defined through "slots" and, secondly, metadata can be stored in the registry through a "ClassificationScheme". Furthermore, "Classification" objects explicitly link the services advertised with the nodes of a "ClassificationScheme". This information can then be used to discover the services by exploiting the ebXML query mechanisms. How to store OWL ontologies into ebXML registries and how to associate these ontologies with Web services are described in [5] and [4], respectively.

Within the SATINE P2P Network, ebXML and UDDI registries are connected to the "Registry Peer". "Registry Peers" can be thought of as wrappers for registries: they facilitate the communication between the service registries and the rest of the P2P network. The requests for storing ontologies, and storing or searching for Web services are realized by the registry peer in terms of JXTA messages. The registry peer processes these JXTA messages and creates the necessary communication with the registries based on the types of the registries. In SATINE, the registry peer communicates with the UDDI registry through the UDDI4J API [39]. For ebXML registries, we adapted the freebXML which is the OASIS ebXML Registry Reference Implementation Project (ebxmlrr 2.1) [11]. With this adapted ebXML registry, SATINE System has the capability of storing and discovering the technical (WSDL), semantic (OWL-S) and security (Security policy) information of the semantically enriched travel Web Services. The registry peer converts the JXTA messages to the related ebXML SOAP Messages, which are then translated to the SQL statements before being sent to the ebXML Registry as presented in Figure 12.

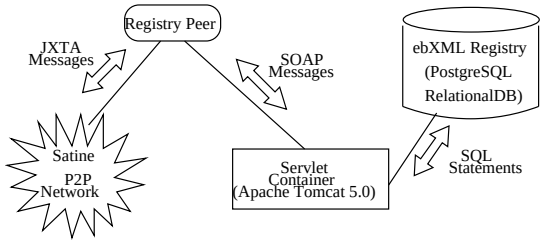

Fig. 12. SATINE ebXML Integration

The ebXML registry implementation consists basically of three parts; the client, the servlet container and the relational database. The relational database constitutes the main ebXML registry; in other words, it holds tables to store all the registry data. The freebXML implementation allows various relational database products such as Oracle, PostgreSQL, and MSSQL to be used as the database. In the SATINE middleware, the PostgreSQL Relational Database [27], which is an open source database, is used as the registry part of the SATINE ebXML Wrapper.

The general overview of the SATINE ebXML Registry structure is presented in Figure 12. As a middle layer, a servlet container is used to interconnect the client and the registry. The servlet container is required for the SATINE ebXML Wrapper because all the messages sent to and received from the client are the SOAP Messages. The servlet container makes the necessary conversions from SOAP messages to SQL statements which are consumable by the registry in the relational database. freebXML implementation essentially supports the Apache Tomcat 4.0.4 or above. For the SATINE middleware, we set up the version of 5.0 for the Apache Tomcat servlet container. The client side, which is the registry peer in the SATINE middleware, constitutes the other end point. The registry peer submits its request to the ebXML Registry with SOAP Messages through the servlet container. These requests can be: deploying an ontology schema, publishing a Web Service with its semantics or querying the registry. For all these requests, client prepares a SOAP message with the core request in it. The classes and packages of freebXML implementation are used and modified to prepare and submit the SOAP messages.

VII. Secure Service Execution

As the complexity and sophistication of application and business logic within Web services increases, the need to provide security becomes more important. To address this need, security standards for Web Services have emerged based on the WS-Security roadmap [42]. Two of the standards introduced to ensure the security of Web services are WS-Policy [41] and WS-Security [44]. These standards allow encrypting the selected parts of a Web service message and to automatically share security token information regardless of platform or protocol, as opposed to the more commonly used SSL, which is simply a way to encrypt the entire channel of communication. By allowing encryption of the selected parts of the messages, these standards facilitate data sharing.

In SATINE network, the security implementation becomes more complex since the Web service messages travel a number of hops in the P2P network before reaching the service provider while the necessary semantic mediations are performed. There are two main factors which increase the complexity of the security implementation: One stems from the SATINE P2P network architecture in which more than two actors are involved in the invocation process while routing of the message. The other factor is the use of different ontologies by the Web service provider and the requestor. The WS policy files are defined on the elements of the message schemas of WS parameters. Since the message schemas may be different, the selected message parts specified to be encrypted in the policy file of the Web service may need to be translated to the requestor's message ontology through semantic mediation.

In the SATINE middleware, every service provider must produce a policy file for each Web service it provides. This

policy file complies with the WS-SecurityPolicy specification [43] and specifies the message parts to be processed by the encryption and signature algorithms. Figure 13 illustrates an example policy file for a Web service that uses a local message ontology. The conformed ontology is specified as “namespace” in the policy file as shown in the figure. This policy file is composed of two assertions: “Integrity” and “Confidentiality”. Integrity mechanism is used to verify that specific parts of the message have not been altered. For each part of the message specified in the “Integrity” section, a security token is associated with it. The “wsp:Usage” attribute specifies that the integrity assertion must be performed. The policy file includes an “algorithm” tag that is defined in the security extension specifying the type of the algorithm to be used for signing the message parts. The XPath expression in the “MessageParts” tag gives the part of the message that will be digitally signed. “Confidentiality” mechanism in the WS-SecurityPolicy specification is used to ensure that specific portions of a message are encrypted using a specific algorithm. As in the “Integrity” part, it specifies the message parts by XPath expressions for encryption and the algorithm to be used to encrypt those message parts. There can be more than one message part as it is shown in the “Confidentiality” tag in Figure 13.

```
<wsp:Policy xmlns:wsp="..." xmlns:wsse="..."
xmlns:ontology="http://bostanci.srdc.metu.edu.tr:8080/
ontology/owl/LocalAirMessageOntology.owl">
    <wsse:Integrity wsp:Usage="wsp:Required">
        <wsse:Algorithm Type="wsse:AlgSignature"/>
        <MessageParts>/InvokeMessage</MessageParts>
    </wsse:Integrity>
    <wsse:Confidentiality wsp:Usage="wsp:Required">
        <wsse:Algorithm Type="wsse:AlgEncryption"/>
        <MessageParts>/InvokeMessage/Instance/rdf:RDF/
        j.0:PoweredAir_MultiAvailability_class
        </MessageParts>
        <MessageParts>/InvokeMessage/Instance/rdf:RDF/
        j.0:PoweredPNR_AddMultiElements_class
        </MessageParts>
        <MessageParts>/InvokeMessage/Instance/rdf:RDF/
        j.0:PoweredAir_FlightInfo_class
        </MessageParts>
    </wsse:Confidentiality>
</wsp:Policy>
```

Fig. 13. Example Security Policy File

SATINE provides a user interface to enable service providers to construct policy files automatically for their Web services. In this interface, service providers specify the message parts to be digitally signed (Integrity) and encrypted (Confidentiality). After creating the Web service policy file, this file is used with the technical and semantic descriptions to publish the service in the SATINE middleware so that the requestors of the Web service can use the file to comply with the policy file of the provider. The important problem here is that the policy file is not directly usable because the ontologies of requester and provider may differ. SATINE solves this problem by transforming the terminologies in the policy file into the global semantics where it is further transformed to the requestor’s message ontology, as it is done for the semantic properties of the service while responding to a service query. This mediation is established based on the mapping definitions which are created and deployed to the SATINE system for once before joining the SATINE network.

In SATINE, if the requestor and the provider peers do not conform to the same local ontology, the super peers will need to decrypt the invocation messages sent between the peers in order to handle semantic mediation as described in Section IV. In this sense, the super peers in the network are “trusted entities”. In addition to this, there is a “trusted peer” in the system whose main functionality is key management including storing the public keys of the peers and the session keys as well as distributing them to the peers securely. The session keys are dynamically generated for every invocation of a Web service, however public keys of all peers currently in the system are stored permanently. In the SATINE middleware, both symmetric and asymmetric encryptions are used. While encrypting messages, the session key generated by symmetric key generation algorithm is used. In order to distribute this session key to the partners, asymmetric encryption keys (public key, private key) are used. For symmetric encryption, “3DES” algorithm [37] is utilized. The “RSA” algorithm [30] is used for asymmetric encryption and generation of public-private key pairs.

As a result of this architecture, the peers in the invocation process need public keys of other partners and the trusted peer shares these keys with others. The peers and super peers send their public keys to the trusted peer when they join the SATINE system. Also when a new “trusted peer” is added to the system, it requests and receives the public keys of the existing peers. Through these key exchanges, the trusted peer obtains the public keys of all super and edge peers so that it can respond to any public key request.

The secure Web service invocation process in the SATINE middleware is depicted in Figure 14. The requestor peer constructs the invocation message which includes the instance of the service message from its own message ontology. After constructing the message, the requestor peer requests the session key for this transaction from the trusted peer. The trusted peer creates a symmetric key, encrypts the key with the public key of requestor and sends it to the requestor. The trusted peer saves the session key into a hash table to serve it to the other peers involved in invocation. The requestor peer decrypts the key with its own private key and obtains original session key. By using this session key and its own private key, the message parts obtained from the policy file are encrypted and digitally signed and the invocation message is sent to the super peer of the requester peer.

The route of invocation message includes the service requestor, provider and their super peers. The super peers’ role in the invocation is the semantic mediation in the existence of more than one global message ontology as described in Section IV. Therefore, the super peers need the original message to perform the necessary mappings. In order to obtain the original message and verify that the message is coming from the true source, the super peers request the public key of the peer that message comes from and the session key for that transaction from the trusted peer. The super peer sends the peer IDs of both edge peers to specify the session while requesting the session key. Every time the trusted peer gets a request for the session key, it checks the ID of the peer that sends the

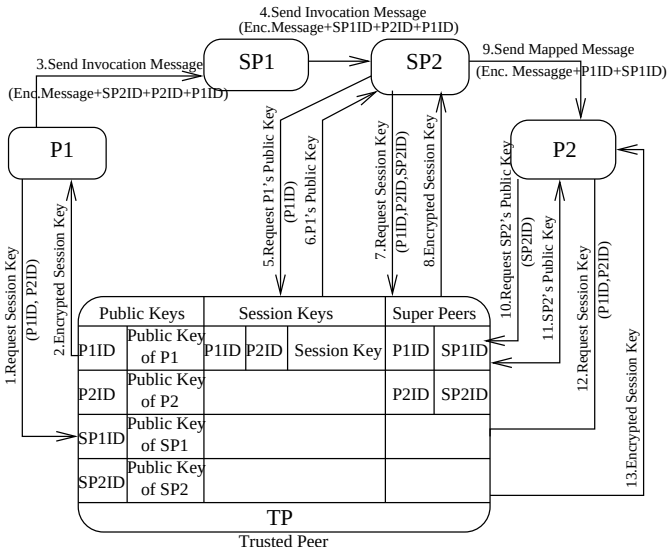

Fig. 14. Secure Service Invocation Process

request to be sure that the peer is authorized to get the key. Checking the edge peers are easy, since the session keys are saved in terms of their peer IDs. For super peers, the edge peers send their super peer IDs with their public keys to the trusted peer as already described. Thus, when a super peer requests a session key by sending the edge peer IDs, the trusted peer checks if the super peer is really the super peer of one of these edge peers. After these security checks, the trusted peer sends the session key by encrypting it with the public key of the super peer. Then the super peer decrypts the session key by its public key in order to use the session key for decrypting the invocation message. After performing the possible global-to-global mappings, the super peer gets the message parts from the policy file and encrypts and signs the specified message parts.

When the message comes to the service provider, the provider peer invokes the service with the input message issued by the service requester that is mapped to the provider's message ontology. The process of returning the result message securely to the service requester is the same as the invocation process.

VIII. Experimental Results

There are two metrics that will effect SATINE system deployment: Ease of set up and use and the performance of the system.

To facilitate system set up for the end users, we have developed an automatic installer as presented in Figure 16. Also we have developed viewlets explaining the set up, discovery, and invocation steps in SATINE environment. These viewlets are available at [33].

In order to evaluate SATINE system's performance in real life settings, we identified sample usage scenarios and the factors that may affect the system performance in these scenarios.

We believe that SATINE system functionality can better be exploited as a middleware running behind a tourism portal.

Fig. 15. A tourism Portal using SATINE middleware

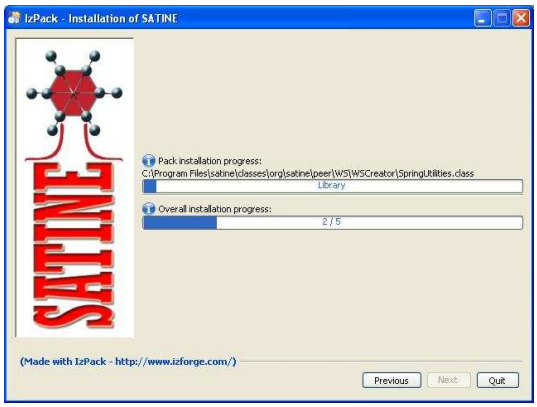

Fig. 16. SATINE Installer

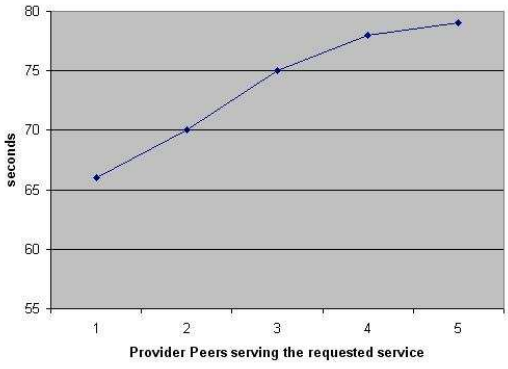

Fig. 17. Results for one service requestor, multiple provider peers

Currently most of the tourism portals communicate with GDSs in the backend to accomplish the requests of the end users. We have integrated SATINE functionalities to the prototype version of a tourism portal that is currently being used in Turkey as presented in Figure 15.

A typical transaction in this portal can be summarized as follows: the end user issues a search request to the portal for example to find the flights available from Istanbul to Rome. Under normal circumstances, the portal communicates with the GDSs it has alliance with and as a response receives the

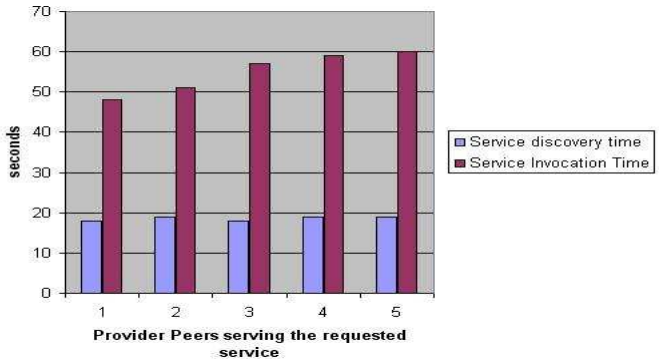

Fig. 18. Discovery and Invocation times for one requestor, multiple provider peers

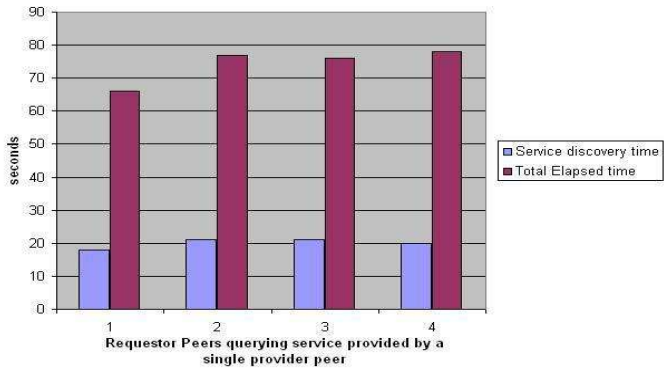

Fig. 19. Results for one service provider, multiple requestor peers

available flight information along with the prices. In SATINE architecture this transaction is performed in two steps: first the available "FlightAvailability" services in the P2P network are discovered through a semantic query, then the discovered services are invoked, and the results of the web service invocations are presented to the user when the invocation of all services are finalized. Based on this scenario, we have identified a typical transaction as (a query service plus the invoke service) operation pair. We set the primary performance metric as the total time elapsed from issuing the query until the results are returned.

In SATINE system, the queries are semantically routed only to the relevant peers in the network, hence the number of peers providing the requested service affect the total elapsed time, rather than the total number of peers in the SATINE P2P network.

SATINE system is implemented in Java and includes around 127,000 lines of code. The testbed contained eight Pentium 4 PCs running at 1.8 GHz with 2GB of memory. The experimental set up can be summarized as follows:

- One super peer, one trusted peer and one ontology manager peer running in one of the PCs,
- In each five PCs, a different service provider peer was running separately, all of them serving "FlightAvailability" services,
- Finally on the remaining two PCs, four service requesters were running, two peers in each machine.

In the first experiment, one service requestor peer is made avaible, and the number of provider peers are increased from one to five. As presented in Figure 17, each provider peer increases the total elapsed time around 2 to 5 seconds. However, the increase has a decreasing slope. In Figure 18, the discovery time and the service invocation times are depicted separately. As it can be seen from the figure the main reason for the increase in the total elapsed time is the increase in the invocation time. However, on the average a service lookup plus invocation transaction takes not more that one and a half minute which is an acceptable response time given that this includes all heterogeneous message translations plus semantic Web service discovery.

Finally in Figure 19, we have presented the results where we have a single service provider, and the number of requestor peers querying this service are incremented. From the results it can be seen that increasing the number of concurrent lookup and invocations do not dramatically increase the total elapsed time.

It should be noted that these experiments are performed in a laboratory environment with limited amount of hardware and the industrial performance of the system is yet to be tested.

IX. Related Work

Providing the interoperability of heterogeneous information systems through ontology mediation has been an active research area recently.

The Harmonise project [13] developed a harmonization network for the tourism industry to allow participating tourism organisations to keep their proprietary data format and use ontology mediation while exchanging information in a seamless manner. For this purpose they have defined a Interoperability Minimum Harmonization Ontology (IMHO) and an interchange format for the tourism industry. The MAFRA tool is used for ontology mediation [16]. MAFRA uses a component that defines the relations and transformations between RDF ontologies. For representing the similarities in a formal way, MAFRA provides a meta-ontology called Semantic Bridge Ontology (SBO).

SATINE middleware has both benefited and extended the Harmonise framework as follows:

- Existing applications are wrapped as semantically enriched Web services to facilitate service discovery and to provide semantic interoperability.
- Ontology mediation is based on OWL ontologies and an OWL mapping tool is developed for this purpose.
- To facilitate the discovery of Web services and to provide scalability a peer-to-peer architecture is developed.

[34] also focuses on integration of heterogeneous data sources in the Semantic Web context using a semantic mediation approach based on ontologies. They use OWL to formalize ontologies of different resources and to describe their relations and correspondences to allow the semantic interoperability between them. The relationships between local ontologies is defined in OWL, i.e., OWL is used as a mapping definition language exploiting native OWL constructs such as

equivalantClass and equivalentProperty. The mediator queries the local ontologies wrapping the backend information systems by using the mapping definition defined in OWL to mediate between heterogeneous local data representations.

This approach is limited to the mapping definition capabilities of native OWL constructs. SATINE middleware extends this mapping especially with various different tools for Datatype property transformation mechanisms which proves to be essential for the tourism application domain since the parties in years have developed different formats for expressing similar information such as hotel rates and flight classes.

Edutella project [17], [19] provides an RDF-based metadata infrastructure for P2P applications, building on the JXTA Framework for sharing educational resources. In [18], a super peer based routing strategy is described for RDF based peer-to-peer networks.

SATINE middleware semantic routing framework benefited from this work and has extended it for semantic Web services.

In [20] a vision on Semantic Retrieval in a P2P network is presented. The authors identified the requirements of the system based on a scenario in tourism domain. Avoiding query flooding in P2P network based on semantics, addressing semantic mismatches between the nodes in the P2P network, and using local repositories enriched with semantics for publishing information are among the identified requirements. Being a vision paper, [20] presents the needs, however the solutions addressing these requirements have not been elaborated.

SATINE middleware complements this work by providing concrete solutions to the requirements identified such as demonstrating how semantic routing of query messages is achieved in JXTA P2P network, showing how semantic mismatches are resolved through an ontology mapping tool, and how Web service registries enriched with semantics can be exploited within a P2P network.

[22] provides a vision on building "fully-enabled service driven systems" by the use of the following enabling technologies: Semantic Web, Web services and P2P architectures. The authors highlight the importance of semantic mediation, ontology based service discovery, and semantically enriched P2P networks for achieving this vision. Scenarios involving knowledge exchange between tourism enterprises and on the fly discovery and invocation of tourism services are presented as motivating scenarios. As in [20], this paper is also a visionary paper: it identifies the requirements and enabling technologies.

SATINE middleware complements these visions by providing a prototype implementation demonstrating how these technologies can be exploited together to achieve a semantic interoperability platform in the tourism domain.

Finally, in [6], the initial ideas on the use of semantically enriched Web services in the travel domain are presented.

X. Conclusions

Web services have become a prominent technology for providing syntactic interoperability. However, in order to fully exploit their potential, it is necessary to introduce semantics. Semantics is domain specific knowledge. Within the scope of the SATINE Project, we have demonstrated how semantically enriched Web services can be used to enhance eBusiness in the travel domain.

In the SATINE middleware, the tourism organizations do not have to advertise their services through Global Distribution Systems. SATINE allows tourism organizations to advertise their services by themselves to the rest of the P2P network. Therefore especially for SMEs, it is advantageous to be a part of the SATINE network which enables their services to be discovered by a wider community without the cost and overhead of connecting to Global Distribution Systems.

Furthermore, in the SATINE middleware, to provide semantic mediation of the exchanged messages, ontologies are derived from the existing local message schemas. These local ontologies are mapped into one another through global ontologies and mapping definitions thus created are used in mapping message instances automatically. Through this mechanism tourism parties exchange messages conforming to their own message schema with the rest of the peers.

SATINE platform provides the necessary mechanisms supporting secure service invocation over P2P network to properly handle commercial transactions.

The performance evaluation of the prototype system in laboratory environment with limited amount of hardware is promising: the results indicate that the response time is within acceptable levels although the Web services invoked are first discovered by using semantic information and their messages are translated back and forth.

References

[1] Amadeus, http://www.amadeus.com/index.jsp

[2] A. Dogac, G. B. Laleci, Y. Kabak, I. Cingil, "Exploiting Web Service Semantics: Taxonomies vs. Ontologies", IEEE Data Engineering Bulletin, Vol. 25, No. 4, December 2002.

[3] A. Dogac, I. Cingil, G. B. Laleci, Y. Kabak, "Improving the Functionality of UDDI Registries through Web Service Semantics", 3rd VLDB Workshop on Technologies for E-Services (TES-02), Hong Kong, China, August 23-24, 2002.

[4] A. Dogac, Y. Kabak, G. B. Laleci, C. Mattocks, F. Najmi, J. Pollock, "Enhancing ebXML Registries to Make them OWL Aware", Distributed and Parallel Databases Journal, Springer, Vol. 18, No. 1, July 2005, pp. 9-36.

[5] A. Dogac, Y. Kabak, G. Laleci, "Enriching ebXML Registries with OWL Ontologies for Efficient Service Discovery", in Proc. of RIDE'04, Boston, March 2004.

[6] A. Dogac, Y. Kabak, G. Laleci, S. Sinir, A. Yildiz, S. Kirbas, Y. Gurcan, "Semantically Enriched Web Services for Travel Industry", ACM Sigmod Record, Vol. 33, No. 3, September 2004.

[7] ebXML, http://www.ebxml.org/

[8] ebXML Registry Information Model v2.5, http://www.oasis-open.org/committees/regrep/documents/2.5/specs/ebRIM.pdf

[9] ebXML Registry Services Specification v2.5, http://www.oasis-open.org/committees/regrep/documents/2.5/specs/ebRIM.pdf

[10] Thomas Erl, " An Overview of the WS-Security Framework", http://www.ws-standards.com/WS-Security.asp

[11] freebXML Registry Open Source Project http://ebxmlrr.sourceforge.net

[12] Galileo, http://www.cendanttds.com/galileo/

[13] Harmonise, IST200029329, *Tourism Harmonisation Network*, Deliverable 3.2 Semantic mapping and Reconciliation Engine subsystems.

[14] IBM UDDI registry, http://www-3.ibm.com/services/uddi/find

[15] Jena2 Semantic Web Toolkit, http://www.hpl.hp.com/semweb/jena2.htm

[16] A. Maedche, D. Motik, N. Silva, R. Volz, "MAFRA-A MApping FRAmework for Distributed Ontologies", In Proc. of the 13th European Conf. on Knowledge Engineering and Knowledge Management EKAW-2002, Madrid, Spain, 2002.

[17] W. Nejdl, B.Wolf, C. Qu, S. Decker, M. Sintek, A. Naeve, M. Nilsson, M. Palmr, and T. Risch. EDUTELLA: a P2P Networking Infrastructure based on RDF. In Proc. of the 11th Intl. World Wide Web Conf., 2002.
[18] W. Nejdl, M. Wolpers, W. Siberski, C. Schmitz, M. Schlosser, I. Brunkhorst, and A. Loser. Super-peer-based routing and clustering strategies for RDF-based peer-to-peer networks. In Proc. of the Intl. World Wide Web Conf., 2003.
[19] W. Nejdl, B. Wolf, S. Staab, and J. Tane. Edutella: Searching and annotating resources within an RDF-based P2P network. In Proc. of the Semantic Web Workshop, 11th Intl. World Wide Web Conf., 2002.
[20] Hao Ding, Ingeborg Solvberg, Yun Lin, "A Vision on Semantic Retrieval in P2P Network", In the International Conference on Advanced Information Networking and Applications (AINA 2004), March 2004, Fukuoka, Japan.
[21] Project JXTA, http://www.jxta.org/
[22] A.Maedche and S.Staab, "Services on the Move - Towards P2P-Enabled Semantic Web Services", Proceedings of the Tenth International Conference on Information Technology and Travel and Tourism, ENTER 2003, Helsinki 2003/01/31
[23] Open Travel Alliance, http://www.opentravel.org/
[24] OWL Query Language, http://ksl.stanford.edu/projects/owl-ql/
[25] OWL Web Ontology Language 1.0 Reference http://www.w3.org/TR/-2002/WD-owl-ref-20020729/ref-daml
[26] OWL-S, http://www.daml.org/services/daml-s/0.9/
[27] PostgreSQL: An Open-Source Object Relational Database Management System (ORDBMS), http://www.paragoncorporation.com-/ArticleDetail.aspx?ArticleID=11
[28] OWL Mapping Tool. http://www.srdc.metu.edu.tr/artemis/owlmt/
[29] RDF Schema: Resource Description Framework Schema Specification, W3C Proposed Recommendation, 1999, http://www.w3.org/TR/PR-rdf-schema.
[30] RSA algorithm, http://en.wikipedia.org/wiki/RSA
[31] Sabre, http://www.sabre.com/
[32] SATINE Project, http://www.srdc.metu.edu.tr/webpage/projects/satine
[33] SATINE viewlets, http://www.srdc.metu.edu.tr/webpage/projects/satine/-viewlet/
[34] S. Suwanmannee, D. Benslimane and Ph. Thiran, "OWL-based Approach for Semantic Interoperability", in the Proceedings of AINA, IEEE Computer Science Press, March 2005.
[35] Simple Object Access Protocol (SOAP), http://www.w3.org/TR/SOAP/
[36] Tomcat, http://jakarta.apache.org/.
[37] Triple DES Algorithm, http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf
[38] Universal Description, Discovery and Integration (UDDI), www.uddi.org
[39] UDDI Java API, http://uddi4j.sourceforge.net/
[40] Web Service Description Language (WSDL), http://www.w3.org/TR-/wsdl
[41] Web Services Policy Framework (WS-Policy), http://www-128.ibm.com/developerworks/library/specification/ws-polfram/
[42] Web Service Security Roadmap, http//www-106.ibm.com/developerworks/library/ws-secroad/
[43] Web Services Security Policy (WS-SecurityPolicy) Web Services Specification: http://www-128.ibm.com/developerworks/library/ws-secpol/
[44] Security (WS-Security) Specifications v1.0, http://www-106.ibm.com-/developerworks/webservices/library/ws-secure/
[45] XML Path Language, http://www.w3.org/TR/xpath
[46] XML Schema Part 2: Datatypes, Paul V. Biron