

Sangam: Universal Interop Protocols for E-Service Brokering Communities using Private UDDI Nodes

Arun Jagatheesan and Sumi Helal

Computer and Information Science and Engineering Department
University of Florida, Gainesville, FL 32611
helal@cise.ufl.edu

Abstract

This paper focuses on the development of protocols for brokering of interoperable e-services. The proliferation of e-services will have significant impact on e-business and various facets of daily life. This trend gives rise to the requirement of automated service discovery and binding. Applications have to discover the most relevant e-services that can satisfy the needs of the user. The discovered e-services have to be interoperable with the user application. These two problems of *relevancy* and *interoperability* among e-services have to be solved to make e-services truly pervasive. *Sangam* addresses these two problems by introducing a framework of brokering communities and a suite of related protocols. Unlike the emerging global registry for *Universal Description, Discovery and Integration* (UDDI), *Sangam communities* use private UDDI nodes to provide reliable, highly scalable and more relevant brokering of e-services. E-services are described by using WSDL documents. In this paper, we describe structure of the hierarchical framework and the protocols used for service discovery at each layer of a Sangam community. A prototype of the Sangam community and its protocols is presented.

1. Introduction

In today's world, a rapid evolution of Internet based technologies enable the deployment of day-to-day necessities and business processes as highly modular web services (also known as e-services). An E-service can be any service that can be accessed across the network using some standard protocol. Super-markets can provide a "*Grocery Order Service*" as an e-service, which can be used by a *House Management System* of customers to automatically order goods based on the customer buying patterns. The next stage of e-business - *dynamic e-business* involves the use of highly flexible e-services rather than using customized applications that are hard-wired to a few business partners in a business hub. In the near future when numerous e-services will be available, applications will not only have to discover a relevant e-service that satisfies the requirements of the user, but

will also have to make sure that the discovered e-service is interoperable with the application system from which the e-service is invoked. Technologies that facilitate software applications to automatically discover and bind with relevant and interoperable e-services are required.

Service discovery techniques currently available like the LDAP [15], UPnP [12], SLP [6], CORBA Trader Service and Jini Lookup service [4] are not the best possible solutions to discover e-services over the Internet. The emerging *Universal Discovery Description and Integration* (UDDI) [14] initiative creates an open framework for businesses to discover each other using a global repository of e-service descriptions. But, UDDI can not be used to search for more relevant e-services that satisfy the requirements of the user. The relationships between businesses to favor another business organization can not be captured using UDDI. The authenticity of entries in the public registry of UDDI can not be checked. A new framework that supports the emerging UDDI specifications and allows for search of relevant e-services is needed. This framework must be highly scalable and flexible, as it is required to support the plethora of e-services from different domains and geographical areas. The availability of most e-services are dynamic based on their load capacity and their *downtime*. This dynamic nature has to be supported in the new e-service discovery framework.

In this paper, we describe the *Sangam community* framework and its associated protocols as a solution for *large-scale service discovery* and interoperability of e-services. Sangam communities are hierarchical communities formed by e-service brokers along with service providers and service requestors. Each community specializes in a particular business sub-domain or a geographical locality of the service providers. Local brokers are used to represent businesses in various communities. The e-service requests (or service queries) are sent to the local brokers. If a local broker does not have knowledge about an e-service requested, it uses the *Sangam Protocols* to gain knowledge about the service from the communities. The *Sangam Protocols* used by the participants of Sangam communities are a step towards *Universal Interop Protocols* with regard to service discovery and binding. Instead of inventing its own protocol stack, Sangam relies upon standards-based technologies like HTTP, SOAP and UDDI to form e-service brokering communities.

The organization of this paper is as follows. In section 2, protocols and technologies related to e-service description and discovery are surveyed. Section 3 describes the general approach of Sangam and description of e-services. In section 4, the framework of Sangam communities is introduced. The Sangam protocols are explained in Section 5. Section 6 gives the implementation of Sangam broker. A sample scenario of usage is given in section 7. Section 8 summarizes our research and contributions.

2. Related Work

2.1. Related Protocols and Technologies

Various protocols have been used for advertising and discovering services over the network. *Light Weight Directory Access Protocol* (LDAP) is an IETF standard providing a scalable hierarchy of name spaces through which services can advertise and clients can locate services [15]. LDAP by itself does not specify any discovery protocols but can be used for service announcements and requests.

CORBA Trader Service (TS) provides an infrastructure for advertisement and request of services by attributes. This makes it possible for any device or service that has a CORBA implementation, proxy, or wrapper to use the Trader Service to advertise its attributes. Requests may be made as constraint expressions. CORBA TS supports federation of trader services into a single logical service. CORBA implementations on different platforms are needed and CORBA protocols may be difficult to implement and configure in small devices.

Service Location Protocol (SLP) is an IETF standard for “spontaneous” discovery of service [7]. SLP establishes a framework for resource discovery that uses a set of agents to operate. One of the drawbacks in SLP is that it relies on multicast discovery mechanisms. This prohibits the usage of SLP in Internet where very high scalability is required.

Jini Lookup service provides a set of Java-based mechanisms and APIs for service lookup and spontaneous discovery [4]. Jini is similar to SLP in many ways. Jini requires service templates based on Java objects. Jini lacks the scalability required for massive e-business applications on the Internet.

We find that these existing protocols are not scalable to the requirements of brokerage of e-services on the Internet or they are tied up with some language. In the next

sub-section, we analyze some of the emerging standards and methodologies related to e-service brokering.

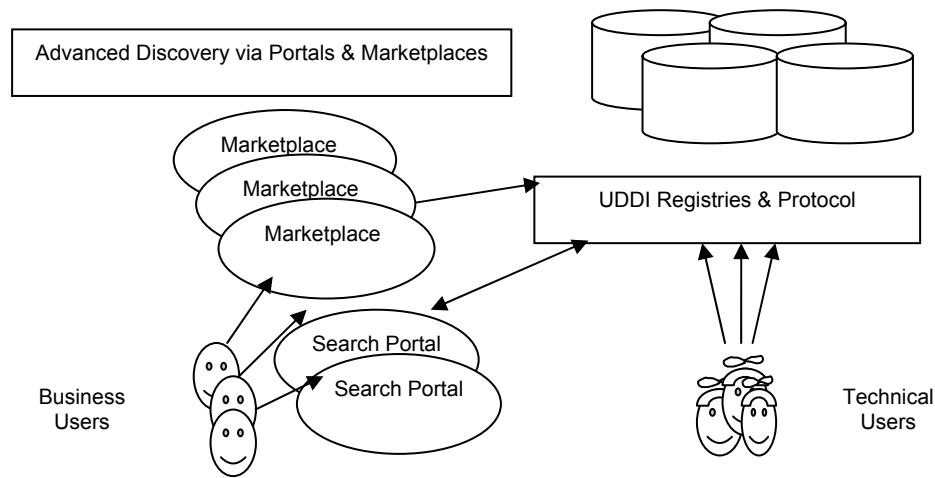
2.2. Emerging Technologies

E-speak [5] platform from HP consists of *Service Framework Specification* (SFS), *E-services Village* and the *e-speak Service Engine*. The broader goal of e-speak is to implement cross compatibility between different marketplace enablers using e-speak services.

The lookup abstraction in e-speak is separated out of the core functionality as a separate service. The e-speak service engine can take a service query and search all known repositories for required service descriptions. E-speak allows the formation of groups among different e-speak engines. The current e-speak advertising service depends mostly on LDAP for its service registration and lookup. E-speak requires every member to support the e-speak platform. Different groups within a network can share a LDAP and there by share the service information with each other. User can combine different groups into an abstract of a *community* (This is different from the Sangam communities discussed in this paper). E-speak does not have protocols using standard technologies for the formation and maintenance of its groups. The match making specification and the interfaces for e-speak is planned to be made UDDI compliant by HP.

The UDDI project is a comprehensive, open industry initiative. The project creates a platform-independent, open framework for e-businesses to interact with each other. UDDI enables businesses to discover each other; to define interfaces using which other businesses can interact over the Internet and to share information in a global repository of e-services. UDDI specification [14] defines a way to publish and discover information about Web services.

UDDI describes a conceptual cloud of Web services and a programmatic interface that defines a simple framework for describing any kind of Web service. The specification consists of several related documents and XML schema, which define SOAP based programming protocol for registering and discovering Web services.



Source:www.uddi.org

Fig 1 Usage of UDDI Registries for Service Discovery

Figure 1 taken from the UDDI technical specification shows the relationship between the UDDI specifications, the XML schema and the UDDI business registry cloud. As shown in the figure, Business Users can use brokers or marketplaces, which internally use the UDDI registries to find other businesses. Also technical users can have applications that either publish or discover data from UDDI. Information about e-services can be published to the UDDI business registry either via a UDDI operator web site or by using tools that can use the programmatic service interfaces described in the UDDI Programmer's API Specification [1].

UDDI does not form a full-featured discovery service [14]. UDDI is designed to complement the existing online marketplaces and search engines by providing them with standard formats for programmatic or automatic service discovery. Businesses need to use any advanced discovery portals or marketplaces which lookup UDDI. The data in the public UDDI registry is not authenticated or validated to make sure an e-service exists as advertised. The relationships among business partners are not captured using UDDI. The global registry is implemented using database approach and is replicated at various operator sites at regular intervals. This might lead to scalability issues in future.

2.3. Relevant Technologies

In this section we list some of the technologies that are relevant to service discovery like the WSDL used to describe web services and SOAP, used to invoke web services.

Web Service Description Language (WSDL) has been submitted as a suggestion to the World Wide Web Consortium for describing services for the W3C XML activity on XML Protocols by Ariba, IBM and Microsoft. “WSDL is an XML format for describing network services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information. The operations and messages are described abstractly, and then bound to a concrete network protocol and message format to define an endpoint” [3].

3. Sangam

This section introduces the Sangam’s Approach and Design Criteria before we describe the Sangam Framework and Protocols in the next sections.

3.1. Sangam's Approach

Sangam aims to achieve more relevant and interoperable brokering of e-services in a highly scalable and dynamic environment of e-services. Sangam uses a combination hierarchical approach and concepts from existing technologies discussed earlier to achieve its goal. The hierarchical model provides very large scalability. Brokers act as intermediaries between the service providers and service requestors providing more relevant and interoperable brokering of e-services. Sangam uses both state based and stateless protocols in its different layers.

One of the criteria used during the design of the framework is to use standard technologies. UDDI is used for the broker registry as it is emerging to become a standard with lot of support from different companies. But, using the public UDDI registry does not solve the problem. All the e-services need not be published to the public UDDI cloud. Most of the e-services in the global registry are not relevant to the other users of the registry. Sangam uses private UDDI nodes in its brokers. Service providers and service requestors associate themselves with a local broker. The service provider registers its services with the local broker using UDDI standard document messages. The other technologies used by Sangam are SOAP, XML and

WSDL. SOAP uses HTTP. Hence Sangam really depends on HTTP and XML which today in the Internet Computing world are synonymous to internetworking and interoperability respectively. We anticipate WSDL to become a standard for describing web-services in the near future.

Sangam uses *service constraints* to find more relevant services for the service requests. Local brokers rank or select services based on the fulfillment of *service requestor constraints* and the *service provider constraints*.

3.2. Service Description

Discovery of an e-service involves the discovery of its service description that can be used to access the e-service. In this sub-section, we take a look at how Sangam uses WSDL to describe its e-services. Each service description must contain two parts: *Interface description* and *Binding description*. Apart from these two, constraint description might also be included in service description.

3.2.1. Interface Description

Interface (or Template) description of an e-service has information regarding the names of the operations that can be invoked in a service and their respective input and output parameters. For example consider a *calculator e-service* that has two input operations *calculate* and *generateRandom*. The *calculate* operation may accept three parameters, namely: operator, operand-1, operand-2. But the *generateRandom* operation does not accept any parameter. When a developer (service requestor) wants to use the *calculator e-service* in his/her application, he or she at build time has to know the name of the service, the name of the service operation and its operands. Also the developer needs information on the data type of the operands. All this information is provided in the Interface description. It must be noted that another service provider who wants to implement the same service can reuse the same template description. So, a given service interface could have many service implementations by different service providers. Hence service requestor applications once designed based on an interface (say “calculator_interface”) can use any service implementation (from any service provider) that is based on the same interface (“calculator_interface”). The only

difference in service description between two service implementations that follow the same interface is their binding description.

3.2.2. Binding Description

Binding Description for a service gives information on where the service is located and how it can be accessed. Each service implementation will have a binding description. The binding description gives information like the protocol type(s), port(s), URL etc that can be used to access the service. Hence if the service provider makes his implementation so that the service can be invoked using various protocols like HTTP, HTTPS, SMTP the description is specified here. To achieve interoperability the service requestor has to specify in the service query the protocol(s) he may use to invoke the service. The broker uses this information as a constraint in finding the services. This solves the problem of interoperability among the e-services by finding compatible e-services.

In automated service discovery, an application uses this information to find a desired protocol that can be used to invoke the e-service programmatically. When an application needs a service, it queries the local broker requesting for service implementations of a service interface. The broker returns the binding information required for accessing the service provider.

3.2.3. Constraints Description

It is often necessary to describe the constraints regarding the attributes (parameters) of an e-service. Service implementations having the same interface description might have different constraint descriptions. If we consider the previous example of `calculator_interface`, even though the interface may be the same for two implementations by different service providers (X and Y), X might output only positive numbers for the operation “*generateRandom*”. Where as, Y might be able to generate both positive and negative numbers. This information has to be present in the service description. The input parameters also might have some constraints. Apart from constraints on a single attribute, there can be inter-attribute constraints. Again in the previous example (`calculator_interface`), there can be an inter-attribute constraint requiring that the sum of `operand-1` and `operand-2` must be greater than

100. These are examples of service provider constraints, which can be described along with the binding description by each service provider.

Apart from the constraints from the service providers, the service requestor can also have some constraints or expectations on the requested service to be discovered. For example, the service requestor would like to have a service implementation of “calculator_interface” which can accept operands, which are negative. These service requestor constraints can be used in the service query.

4. Sangam Framework

The *Sangam Community* consists of *local brokers*, *domain brokers* and *Super brokers*. The local brokers are the e-service brokers, who represent a set of service providers and service requestors. It is also possible that a business uses a local broker to represent its internal systems, which may contain service providers and service requestors. The community as shown in Fig.2 is classified into abstract layers as: *Service Layer*, *Brokering Layer* and *Super Brokering Layer*.

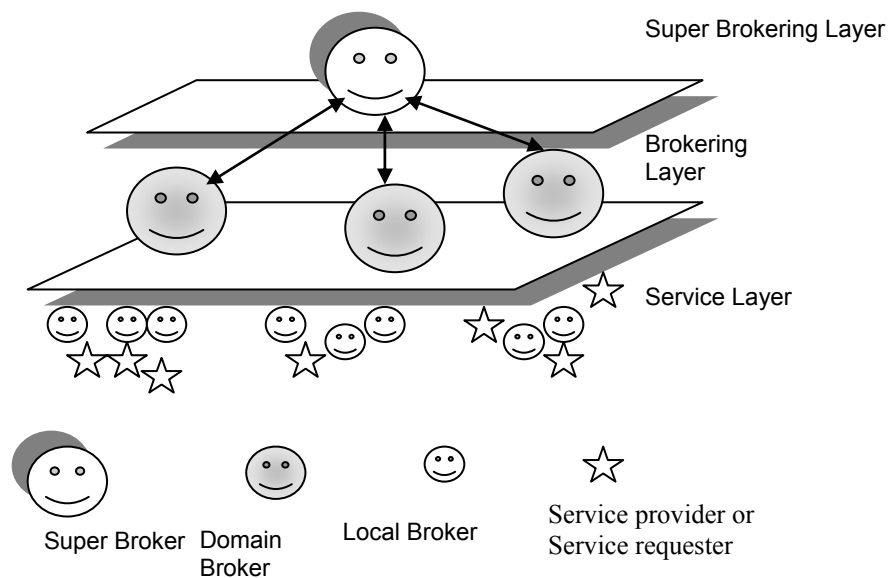


Fig 2. Hierarchical Brokering Architecture of Sangam

4.1. Service Layer

The *Service Layer* is the bottom layer in the community hierarchy and is composed of *Local Brokers*. The local brokers act as representatives for a local set of service providers and the service requestors, which are organized below the local broker. Service providers in this layer register their service descriptions with the local broker. The service descriptions contain information on the service interfaces the service provider has implemented and the end-points to access the services. A service provider might register the same e-service implementation with any number of local brokers at the service layer. The local brokers pass the description of the registered e-services to the corresponding *domain broker* in the next layer, which may advertise the e-service to the entire community.

Service requestors in the service layer make service queries to the local broker. The local broker together with the service providers and service requestor who are organized under it forms a brokered system and facilitates service discovery among the members. If the local broker is not able to find a service provider satisfying a service query, it contacts the domain broker to find the service.

4.2. Brokering Layer

The *Brokering Layer* consists of *domain brokers* who specialize in e-services related to a particular domain or sub-domain. The domain broker specializes in knowledge of service providers about a certain business domain. Every domain broker in the brokering layer registers with the domain *Super broker* of the community. The *Brokering Protocol* governs how brokers in the *Brokering Layer* advertise on behalf of the service providers; how service request are satisfied by the domain broker and how domain brokers self-organize themselves in the community. The protocol for knowledge discovery results in expansion of knowledge of e-service descriptions among the domain brokers.

4.3. Super Brokering Layer

The *Super Brokering Layer* is the top layer of the community. In this layer, the Superbroker helps in the self-organization of a domain based community using the *Super Brokering protocol*, which governs how super brokers can communicate

and coordinate over common service interface descriptions between their communities.

The Sangam Community is easily scalable to a very large size by accepting new brokers into the community. When a domain broker joins or leaves the community the Super Broker updates the knowledge among the members of the community. Super Brokers have control over load balancing within their community and enhance reliability among the community members by keeping only the domain brokers who can be trusted as members of the community.

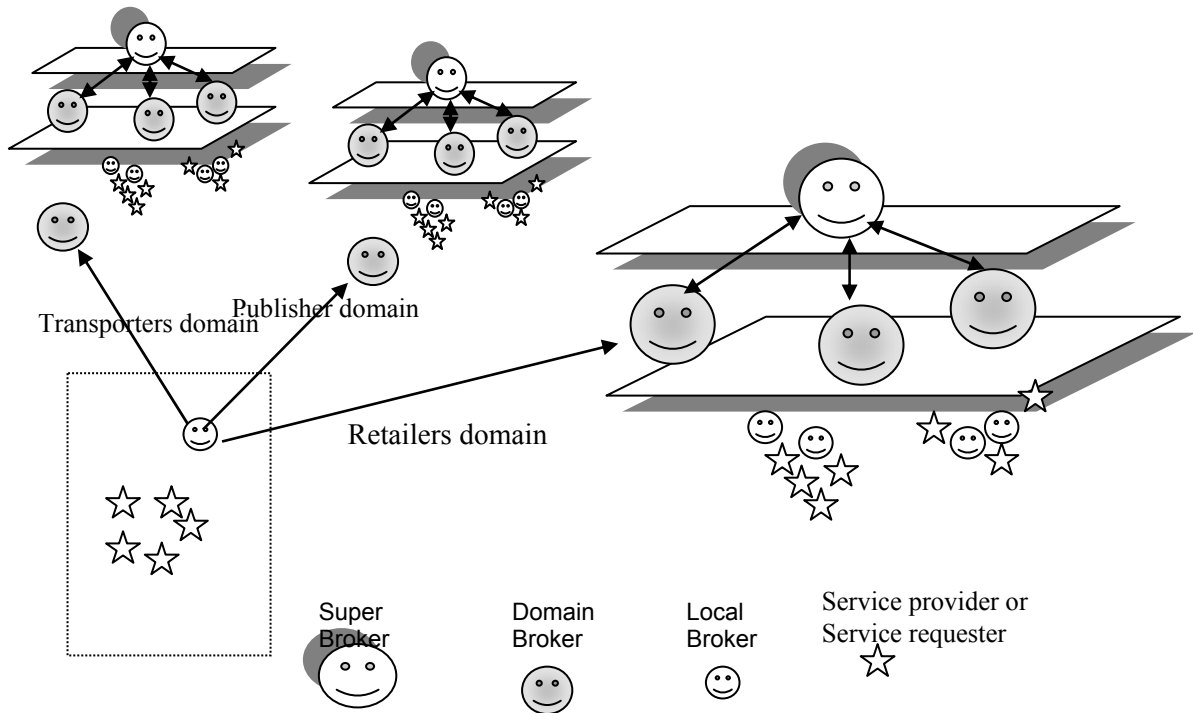


Fig 3. Local Broker along with various domain brokering communities

4.4. Advantages of using a Local Broker

A business may use a Local broker to represent itself in the desired domain-specific communities. In this case, as shown in Fig.3, the local broker becomes the single source of information about any e-service for all applications used by the business. One of the advantages of using a local broker to represent entire business is that a single point of contact with the external world is formed which is easy to

maintain. Another major advantage is the ability to do both “*arms-length*” transactions and “*relationship*” transactions which analysts consider very important in B2B environment [16].

Researchers have designed different organizational structures for brokering systems, such as peer-to-peer multi-brokering architecture of *InfloSleuth* [13], the partitioned repository design in *Blackboard Systems* [2], centralized message routing design in *JAT Lite* [9], and the multi-facilitator mediation in OAA [10]. The hierarchical Sangam Community is designed to meet the current requirements of scalability, interoperability and highly customized brokering. This hierarchical layer approach and protocols of Sangam for e-service discovery are similar to the framework used in *Brokering Based Agent Community Protocols* (BBACP) [8]. Unlike BBACP, Sangam uses standards-based technologies and focuses on e-services discovery. The Service layer and protocols are modified to use private UDDI nodes for the brokers. Leader election protocol may be used in the event of failure of the Superbroker to elect a new Superbroker in the community. Each broker only needs to have the information about the Superbroker it is registered with, and communicates with other brokers only upon necessity. This reduces the amount of messages flowing among the brokers within the community. Also, the community becomes self-organizing, adapting itself based on the unsatisfied requests and the availability of brokers.

In the next section we take a look at the protocols of Sangam and how they allow the brokers in the different layers to interact with each other.

5. Sangam Protocols

Brokers in each layer interact with the other layer or among its peers in the same layer using the *Sangam Protocols*, which consist of *Service Protocols*, *Brokering Protocols* and the *Super Brokering Protocols*. Sangam protocols are based on document exchange using SOAP messages.

5.1. Service Protocols

The Service Protocols illustrate the base level commitment and responsibility of the local brokers to the community. The local brokers interact with the domain

broker using the service protocol. The service protocols provide the basic brokering functionality such as register/unregister e-services with the domain broker, Browse the categories and service descriptions supported by the domain broker.

The local broker uses the Sangam service protocols to publish its public e-services (which can be accessed by any one) to the corresponding domain brokers based on the category information. Whenever a public e-service is unpublished or updated from the local broker, the local broker has to update the information in the community by contacting the corresponding domain broker with whom the service would have been previously registered. A local broker must implement all basic Sangam messages using SOAP conforming to Sangam Protocol Specification. Apart from these basic messages, the domain broker may provide some additional SOAP messages, which may be viewed as value added services like “getBestService”.

It must be mentioned that service provider and service requestor can use the UDDI client protocol messages to communicate with the local broker, which is a UDDI private node. The service protocols are stateless, less expensive and easy to implement. They are suited for in pervasive environments like *House Management Systems* to discover services offered by other devices in the house or e-services offered by other systems on the Internet.

5.2. Brokering Protocols

The *Brokering protocols* describe a cooperative multi-brokering system, which provides the solution for interoperation among brokers in a dynamic and environment of local brokers, service providers and requestors. Each domain broker performs brokering functionality, such as service discovery, dynamic service composition and knowledge sharing with the community. Domain brokers representing a set of local brokers can advertise the e-service descriptions and send capability queries to other domain brokers. The brokering protocols regulate the joining and leaving of domain brokers from a community. It also governs e-service knowledge discovery and sharing of acquired knowledge.

Unlike the Service Protocol messages, which are stateless, the messages of the brokering protocol involve necessary state information. The reason for having state based protocol is to be more selective in admitting other domain brokers into

the community. The sub-protocols of the Brokering Protocol include Joining Protocol, Departure Protocol, Knowledge Discovery and Exchange Protocol and Capability Advertisement Protocol.

5.3. Joining Protocol.

The domain broker uses the *Joining Protocol* to join a community; it includes a logic conversation between the domain broker and the Superbroker. The conversation policy is a sequence of rule-based communication mapped into a finite state machine as illustrated in Fig.4. Each domain broker, in its application to join the community, specifies the service interfaces and categories it deals with.

After receiving a registration attempt (State 1), the Superbroker conducts multifaceted membership evaluation of the applicant for credibility, possible value addition to the existing community, knowledge and capability relevancy among others. This evaluation results in *acknowledge* (State 1 \rightarrow State 3) or *reject* (State 1 \rightarrow State 6).

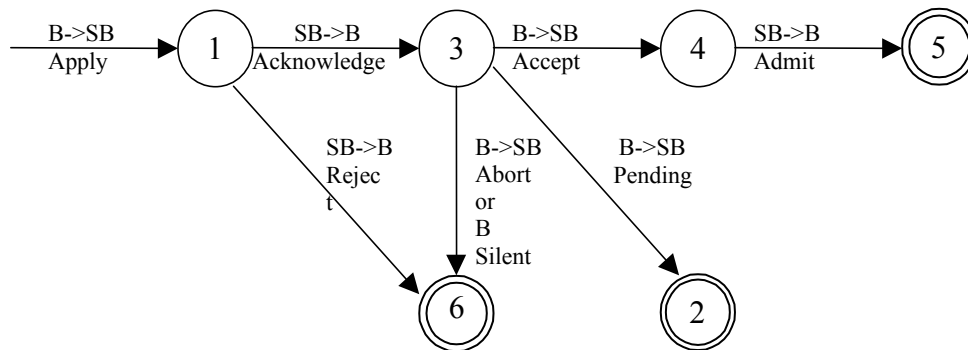


Fig 4. Finite state machine of Joining Protocol

The acknowledgement message from Superbroker to the applicant domain broker contains the necessary information used by the domain broker to join the community and also the default or basic service interfaces the domain broker might need in the community.

When the domain broker receives the acknowledgement from the Superbroker (State 3), it automates its behavior, if needed to fit into the community.

The domain broker now autonomously makes a final decision to join the community or not based on the knowledge it can gain from the community. If the domain broker makes a positive decision, an acceptance message is sent to the Superbroker (State 3 → State 4). The finite state 5 is reached, when the Superbroker confirms the grant of membership upon receiving acceptance message. If the domain broker decided not to join the community, an abort message is sent to the Superbroker (State 3 → State 6).

If a decision cannot be made by the domain broker whether to join the community or not, a pending message is sent to the Superbroker. In order to allow the applicant to continue the member application in the future, the Superbroker maintains the knowledge of access history. The state information in the conversation policy is maintained at the Superbroker side. During application processing in future, the Superbroker first checks the access history, following the conversation policy based on the state information. If the Superbroker does not receive any replies within a specified period, the synchronous conversation will move to the finite state of application failure (State 6).

Upon approval of a successful registration, the Superbroker updates information about the new broker in its repository. Failure of registration could encourage the domain broker to engage in additional learning activities about the community. The domain broker could also be modeled as an autonomous agent that can make autonomous decisions in the community.

5.4. Departure Protocol.

The departure protocol is used to inform the Superbroker that the domain broker won't be able to provide the advertised e-service(s) any more. The leaving broker initiates the leaving procedure by sending a SOAP message "*LeavingCommunity*" to the Superbroker. The knowledge and services that would become unavailable because of the departure of this domain broker is updated in the community by a SOAP message "*UnpublishBroker*" by the SuperBroker to the member domain brokers in the community. Upon receiving the "*UnpublishBroker*" message, the brokers remove the services advertised by the leaving broker from

their repositories. The Superbroker makes the necessary changes in its member repository, member credit history and service quality about the leaving broker.

5.5. Knowledge Discovery and Exchange Protocol

The individual domain brokers interoperate in the community to gain knowledge to serve more service queries from the local brokers. When a local broker can not satisfy a service request, it contacts the domain broker of that service, which tries to find the requested service within its own brokered system. If the domain broker is not able to satisfy a service request within its brokered system, the request along with the constraints is sent to the community (using the superbroker) as an “*Ask*” message to other domain brokers (of the same domain) within the community. The capable domain brokers in the community, who can satisfy that service, provide the requester with the information about the service requested. This process leads to discovery and expansion of knowledge within the requesting domain brokers.

A domain broker can also request to be informed if some specific service with certain constraints becomes available in the community. This form of knowledge discovery is called *subscription*. A broker may broadcast a “*Subscription*” to the community, which is stored by each member broker. If a broker (or a service provider it represents) is capable of serving the subscribed service, the broker will reply to the subscription and start a conversation with the subscriber. A broker can also unsubscribe its subscriptions.

Knowledge discovery protocol follows a finite state machine. If a request is satisfied and a discovery is made, the broker updates it self about the acquired capabilities. In case of many unsatisfied requests for the same service query, the broker may send a subscription for that service to the Superbroker.

5.6. Capability advertisement protocol

The domain broker can advertise the e-services that can be provided by its local brokers. When the Superbroker receives an advertisement, it checks for any subscriptions that can be satisfied and informs the subscribers. When a domain broker needs to make a change in the service advertised, it sends an unadvertise

message to Superbroker and makes an update. Advertisements are basically the means of getting business for the e-service providers.

5.7. Super Brokering Protocols

The *Super Brokering Protocol* governs how a Superbroker communicates and coordinates in an *Internet Enterprise* of communities, each specializing in a particular business domain. This protocol covers the community to community interaction and interoperability. The Superbroker Protocol is also responsible for Superbroker election and maintenance of the community. This set of protocols is under future work of this research.

Now that we have seen the architecture and protocols of Sangam, we will take a look at the architecture of a local broker in the Services Layer.

6. Implementation

As shown in Fig,5, we have a UDDI Client that is extended to handle Service Protocols. It communicates with the CSR(Constraint Satisfaction Ranker) and the CSP(Constraint Processor). The CSR is used to rank the degree of satisfaction by a constraint object and the CSP is used to compare two constraints and confirm if they satisfy each other. Community Interface and Community Broker Kernel are currently under development. We have used the IBM implementation of the UDDI registry.

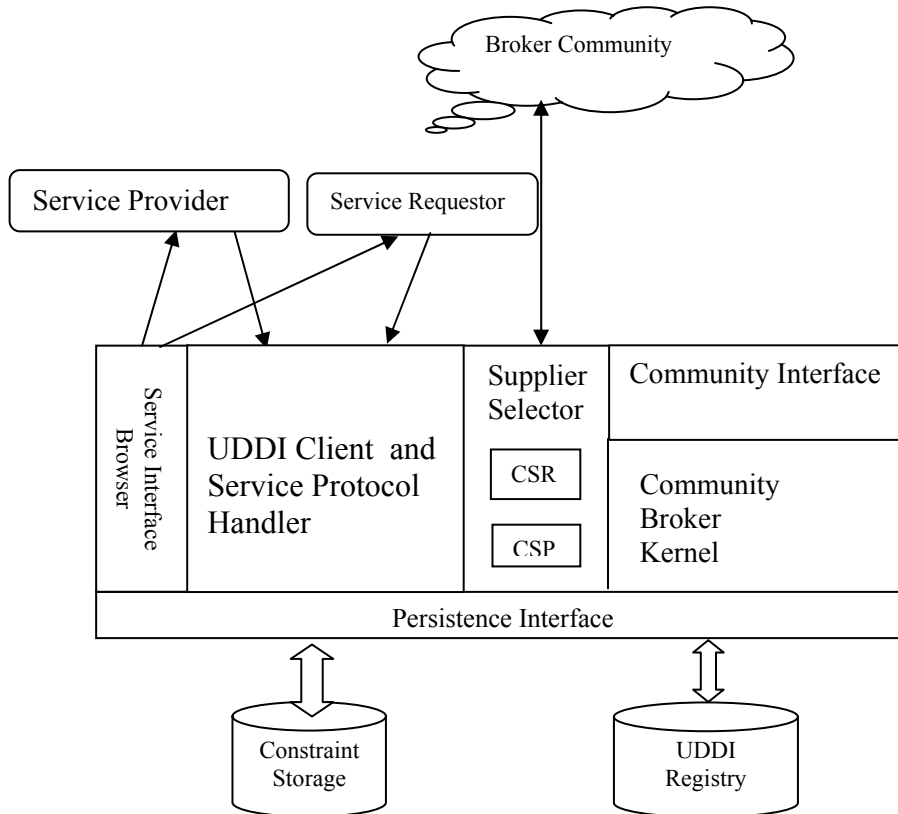


Fig 5. System Diagram for a local broker

7. Application Scenario

E-services can be composed together to form a dynamic Internet-based workflow. A Workflow engine dynamically determines the individual tasks that will be used to accomplish an activity [11]. Workflow systems can benefit by using e-services broker to dynamically find and bind to appropriate service providers for the individual tasks to be completed. Thus, the service broker enables late binding of services and critical in determining the dynamic workflow. The Workflow engine is a service requestor looking for appropriate service providers.

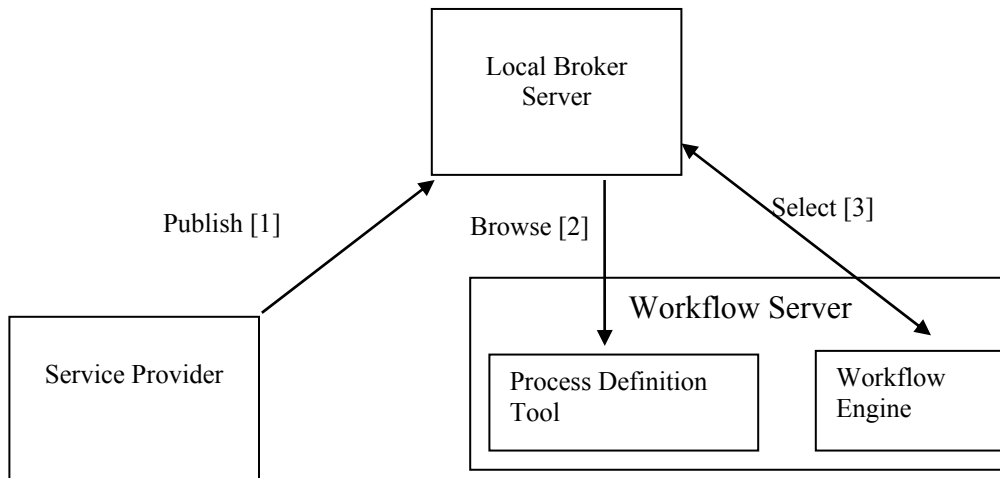


Fig 6. Service Discovery and Binding in Workflow system

As shown in Fig.6, the major operations performed in the *Internet Enterprise System* are:

- *Publish/Unpublish.* Service providers advertise (publish) their e-services to one or more registries.
- *Browse.* The designer of the workflow browses service interfaces available in the local broker that can be used in the design of the workflow model. This is *design time discovery*.
- *Select.* At run time the Workflow engine sends e-service requests to the Broker server to get desired service provider(s) who satisfy these e-service request (s). The e-service request may also have constraints. The local broker uses its knowledge and Sangam protocols to find the possible service provider binding information for the Workflow. This *run-time discovery* is automatic without any human intervention.

8. Conclusions and Future Work

Automation of e-service discovery and binding is required in future distributed applications scenarios. A framework and protocols that facilitate more relevant, interoperable and large scale e-service discovery are required. Sangam's hierarchical framework of brokers using private UDDI nodes and protocols more

suitable for real world situations have been proposed. Brokering of e-service constraints along with their WSDL description is suggested to provide more relevant services. Sangam prepares for the proliferation ubiquitous eservices [17] and workflow where devices and application systems can discover and bind with services automatically.

The algorithms used to match the constraints of the service providers and requestors may be improved to handle more data types and efficiency can also be improved upon. The Superbroker protocols that can be used for inter-community interactions have to be investigated further. Once more applications have been built and tested, Sangam can be made as a standard for community based large scale service discovery.

9. Acknowledgement

We would like to acknowledge Professor Stanley Su and Jie Meng for the valuable discussions on Dynamic Workflow concepts.

10. References

- [1] T. Boubez, M. Hondo, C. Kurt, J. Rodriguez and D. Rogers. "UDDI Programmer's API 1.0". <http://www.uddi.org/specification.html>
- [2] N. Caver, and V. Lesser. "The Evolution of Blackboard Control Architectures," CMPSCI Technical Report 92-71, October 1992.
- [3] E. Christensen, F. Curbera, G. Meredith and S. Weerawarana. "Web Services Description Language (WSDL) 1.1" W3C Note March, 2001. <http://www.w3.org/TR/wsdl>
- [4] K. W. Edwards. *Core Jini*. Prentice Hall, 1999.
- [5] HP E-Speak Web site. <http://www.e-speak.hp.com/>
- [6] E. Guttman, "*Service Location Protocol: Automatic Discovery of IP Network Services*" IEEE Internet Computing, vol. 3, no.4, pp 71-80, 1999.
- [7] E. Guttman, C. Perkins, J. Veizades., and M. Day. "*Service Location Protocol, Version 2,*" IETF, RFC 2608, June 1999. <http://www.rfc-editor.org/rfc/rfc2608.txt>
- [8] A. Helal, M. Wang, A. Jagatheesan. "Service-Centric Brokering in Dynamic E-Business Agent Communities," Journal of Electronic Commerce Research (JEER), Baltzer Science Publishers.
- [9] JATLite. <http://java.stanford.edu/>, May 1999

- [10] G. L. Martin, A. Unruh and S. D. Urban. "An Agent Infrastructure for Knowledge Discovery and Event Detection," Oct 1999. <http://www.mcc.com/projects/infosleuth/publications/>
- [11] J. Meng, A. Helal, and S. Su. "An Ad-Hoc Workflow Architecture Based on Mobile Agent and Rule-Based Processing," The special session on Software Agent-Oriented Workflows, Proceedings of the International Conference on Parallel and Distributed Computing Techniques and Applications, Las Vegas, Nevada, June 2000. pp. 245-251. <http://www.harris.cise.ufl.edu/projects/publications/adhocWF.pdf>
- [12] Microsoft Corporation, "*Universal Plug and Play Device Architecture*", White Paper, Version 1.0, June 6, 2000. http://www.upnp.org/UpnPDevice_Architecture_1.0.htm
- [13] M. H. Nodine, W. Bohrer, and A. Ngu. "*Semantic Brokering over Dynamic Heterogeneous Data Source in InfoSleuth*," ICDE 1999: 358-365.
- [14] UDDI website. <http://www.uddi.org/>
- [15] M. Wahl, T. Howes, and S. Kille. "*Lightweight Directory Access Protocol (v3)*," IETF RFC 2251, December 1997. <http://rfc-editor.org/rfc/rfc2251.txt>
- [16] A. Siegev, C. Beam and J. Gebauer. *Impact of the Internet on Purchasing Practices. Preliminary results from a Field Study.*
- [17] A. Sahai and V. Machiraju. "*Enabling Ubiquitous E-services Vision on the Internet*" E-Services Software Research Department, HP Laboratories. <http://www.hpl.hp.com/org/stl/emd/pubs.html>G