




Cooperative Distributed UDDI (dUDDI) Architecture for P2P Service Networks

P. Victor Paul¹ · Achyut Shankar² · L. Jayakumar³ · Shailesh Khapre⁴ 

Accepted: 3 September 2023 / Published online: 25 September 2023

© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2023

Abstract

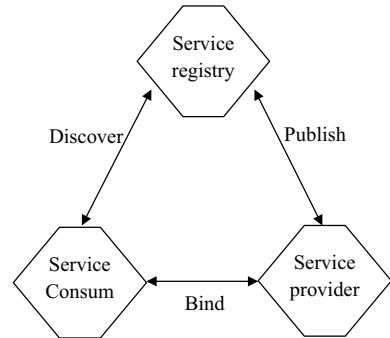
A web service is a software interface that describes a collection of operations that can be accessed over the Internet using standard protocols. Though web services have significant features, centralized UDDI architecture is one of the most challenging issues which attract researchers for an efficient solution. In this paper, a cooperative distributed UDDI (dUDDI) architecture for P2P service networks has been proposed. dUDDI system decentralizes the traditional UDDI using a collection of minimum traffic components which maintains the service provider discovery start list. Service providers act cooperatively on the service discovery operation by linking to other providers who offer similar services. A comprehensive description of the various elements in the dUDDI architecture and their internal component is presented. We also presented an effective algorithm for service publish and discovery operations using dUDDI architecture. The proposed model improves the efficiency of service resource retrieval and also applies different security measures. The proposed dUDDI model is evaluated with the best-in-class working decentralized UDDI models by considering different conditions like the registry size, QoS factors and discovery of the relevant services based on user request. A testbed has been generated consisting of 1000 web services of various domains and services are manually divided into 21 domains with different QoS requirement combinations. The experimentation results justify that the proposed model outperforms the existing decentralized UDDI models in terms of precision, recall and f-measure factors.

Keywords Distributed · UDDI · dUDDI · P2P · Web service · Registry

1 Introduction

Web service is an emerging technology that allows interaction between applications in a programming language and platform-independent manner. World Wide Web Consortium (W3C) describes web service as a software system designed to allow interoperable machine-to-machine communication over the internet using XML, SOAP, WSDL and UDDI open standards. The key characteristic features of web service are, XML-based which allows platform independence, Loosely coupled that improves the system manageability and integrity, Coarse-grained services allow a proper level of information

Fig. 1 Basic web service architecture



exchange, operate in both synchronous or asynchronous ways in the execution of services, handle transparent document exchange to support business incorporation, Interoperability allows the system to work on different technologies, Platform and language-independence keep no dependencies for programming language and operating system.

Web service architecture consists of three fundamental elements which communicate with each other as in Fig. 1. The basic three elements of web service computing architecture are the Service provider, service requestor and service registry. A service provider is responsible for creating a service, generating its service descriptions. A service consumer needs service and locates service descriptions that are published in one or more service registries. The service registry is responsible for promoting service descriptions published in it by service providers, and for making the service consumers find the required service from the collection of service descriptions published within it. Within the web service architecture, the service registry is a logically centralized directory of services. At present, there are three leading viewpoints on how a discovery service should be performed on the directory: as a *registry*, as an *index*, or as a *peer-to-peer* system.

Web service has three main components such as SOAP, WSDL and UDDI which have emerged as open standards for communication over the Internet. Simple Object Access Protocol (SOAP) is a platform-independent, XML-based communication protocol for information exchange between computers/applications over the internet. Web Services Description Language (WSDL) is an XML-based standard format for describing a web service. UDDI (Universal Description, Discovery, and Integration) is an XML-based specification designed for describing, publishing, and locating web services. Though web services have significant features, there are several technical challenges associated with them like centralized UDDI architecture, service availability, provider-consumer security, transaction integrity, and scalability. The issue taken into research in this work is the centralized UDDI architecture which means that the service registry is a single-point component for service publishing, selection [39, 41–43] and discovery operations [8, 9, 35, 40]. Thus the centralized UDDI (cUDDI) have the risk to be a single point of failure and network traffic, lack of efficient service description updation mechanism, performance degradation with an increase in service providers/consumers (scalability) and difficulty to maintain a large collection of service entries at a single place. To solve the various critical issues of cUDDI, a peer-to-peer structured UDDI will be an optimal solution. In this paper, we propose a distributed UDDI (dUDDI) framework which is a cooperative, loosely coupled P2P structured service provider model managed by a minimum loaded component called Service Provider Locator and Service Aggregator (SPLSA).

2 Related Works

There have been various strands of research in the literature to support a peer-to-peer-based framework for the UDDI registry. We present below an account of the various approaches for the decentralization of UDDI architecture performed by researchers.

Jo et al. [20] describe the various benefits of integrating web service and P2P technologies and design issues that arise in constructing the framework design. A concise design for three types of web service operation processes such as service publish, inquiry and invoke using sequence diagram [32]. Shadija et al. [18, 19] presented a technique to completely decentralize a service-oriented architecture using a self-organizing peer-to-peer network maintained by service providers and consumers [28]. In [2], a novel P2P-based UDDI (PDUS) approach is presented using distributed Register Center Nodes (RCNs) in which services are published. Libing et al. [31] claim that the centralized UDDI model leads to performance degradation because too many services are being registered and queried. They proposed an interoperable model for distributed UDDI with three server types namely root, super domain and normal server which are managed hierarchically. Though it discussed little on the service publish operation but service discovery, security, scalability and reliability issues are left untouched.

Wang et al. [3] proposed a mechanism for web service publication, discovery and invocation with the scheme of a P2P network using super/group peer without a dedicated “central server”. Zongxia et al. [4] proposed an active and distributed registry, Ad-UDDI, the service information is distributed among multiple registries to avoid traffic bottlenecks in one public UDDI. This Ad-UDDI implements a three-layered hierarchical model of distributed service registry which again has the risk of single point crash on top layers and also propagation of service update information within the system is more complicated. In [9], authors claim that decentralizing UDDI registries leads to another level of complexity on how to effectively discover the services across the distributed registries. In [16] Al-masri et al. depict that the ability to discover Web services of interest then across multiple UBRs becomes a major challenge. But the architecture presented lacks an effective provider-consumer communication structure and crawling all the available registries is virtually impossible to accomplish. In [33], authors proposed a new structured P2P overlay network infrastructure designed for web services discovery operation which supports two ranges of queries [17]. In [39], Wang et al. proposed QoS-Aware Service Discovery and Selection technique for a cloud environment based on an optimization algorithm.

In [2, 3] and [20], only a theoretical description of the proposed system has been discussed but experimentation and analysis, which must justify its effectiveness, are left for future research. There are other remarkable contributions to peer-to-peer based web service architecture [38] have been proposed such as pService [5, 6], semantic web service system [1, 14, 30] hash table-based super peer [10], broker-based design [11, 21], agent-based organization [12, 13, 22], Bio-Inspired models [34, 36, 37, 39, 41] and semantic-based peer classification [7, 15, 41]. Each technique has its pros and cons, but none of the work can be claimed as the most thriving architecture based on the critical performance criteria. In the work [23], a distributed UDDI framework has been developed to address the problems of the existing centralized UDDI model. The framework followed the recent release specification of UDDI v3 and focused on minimizing unequal access to services, poor processing performance and vulnerability of single point of failure. In addition, service publication and service discovery have been improved and the strong practicality of the distributed UDDI method was validated. In [24], the authors claim that the centralized

UDDI suffers from a single-point failure issue and high maintenance costs. To cope with these issues, the researcher proposed a novel framework based on a catalogue of mobile agents and metadata for web service discovery. The proposed model works based on the user profile to search and find a suitable web service, satisfying consumer requests, in less time and taking into account the QoS properties [25, 29].

Baresi et al. [26] claim that the service registry in SOA concepts is not very effective due to problems of safety and governance. To overcome the issue, a distributed implantation of registry DREAM (Distributed Registry by ExAMple) is proposed which is a public / subscribed solution to combine existing, separate registries, along with a corresponding approach to ease the publishing and discovery of services. The authors proposed a system of chord-based semantic service discovery framework with QoS consideration [25]. In the model, QoS-related criteria are placed in OWL-S to describe services and adopt chord-based distributed storage. Also proposed an OWL-QoS-based matching algorithm to discover services. The findings of the experiment demonstrated that the method proposed increased the reliability and accuracy of the discovery of web services. In [27], a novel architecture is proposed based on the mobile agent, and user profile for web services discovery to mitigate the search space and improve the relevant service retrieval.

Observations from the study: The registry type of directory holds fast retrieval requested service but it suffers single-point failure that affects the reliability of the system, inactive service references and complex ownership constraints. On the contrary, the peer-to-peer type of directory holds decentralization, reliability against single point failure and active & updated service pointers but suffers from problems such as high response delay, high performance-cost, path propagation of request (a peer may receive the same service request many times), inefficient service search, overload network with service request message and no guarantee that a request will spread across the entire network. Thus, these two types of directories should be merged to have the advantages and also limitations of one type will be mutually surmounted by the other. A novel P2P-based UDDI architecture is required which should have the following features, decentralized registry, no single point of failure, effective method for service publishing, distribution and discovery, retrieved service will be active and updated and improved system reliability and scalability.

3 Proposed System

3.1 The dUDDI Architecture

The proposed distributed UDDI (dUDDI) architecture is a combination of the registry and peer-to-peer types of service registries. The service providers and consumers are arranged in a P2P fashion and the dUDDI system contains a collection of minimum-loaded SPLSA which acts as a local service directory. Figure 2. shows the distributed structure for the web service paradigm using dUDDI architecture.

It is much essential to understand the relationships among various entities involved in the proposed web service paradigm using dUDDI architecture. Figure 3 depicts the complete web service-centred relationship among the various components in the dUDDI architecture. The distributed UDDI architecture proposed in the research consists of three elements, *Service provider*, *Service consumer* and *dUDDI system*. Figure 4 illustrates the architecture of dUDDI with its elements and the interaction among them.

The web service computing environment and operations can be represented as,

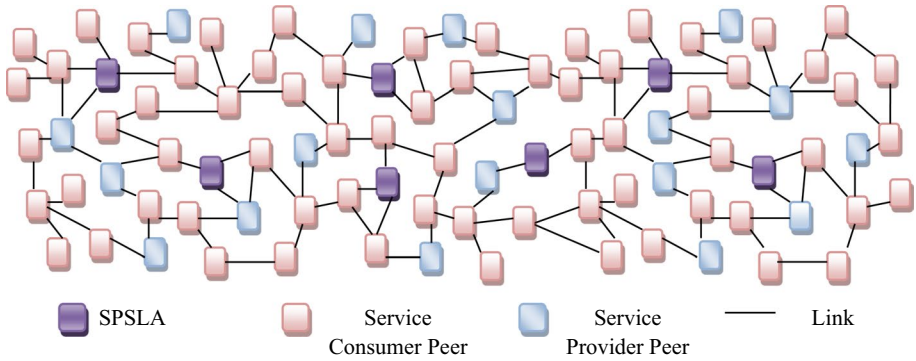


Fig. 2 Distributed structure for web service paradigm using DUDDI

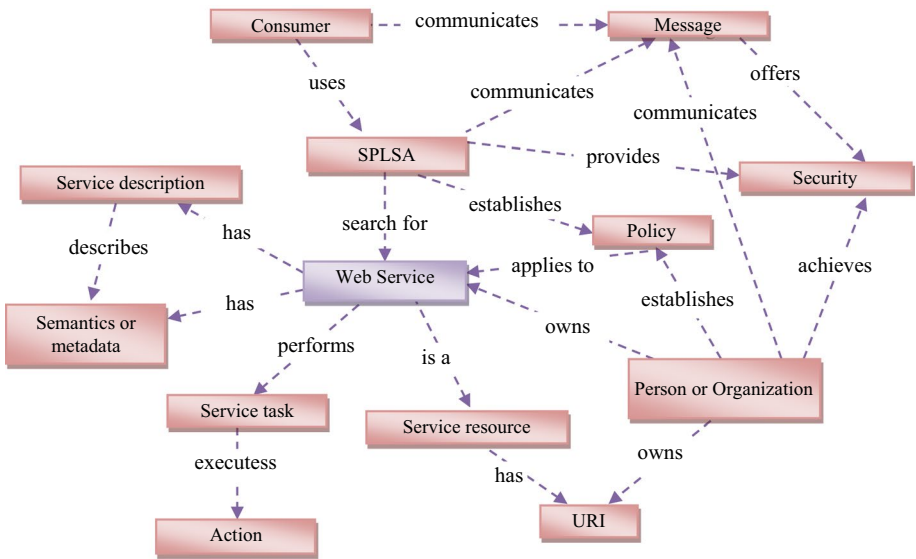


Fig. 3 Service centered relationship diagram for various components

$$WS_{opr} = \{P, D\}$$

where, WS_{opr} refers to the web service computing model operations, P refers to the publish operation and it can be represented as,

$$P = \{T_s, I_s, W_s\}$$

where, T_s represents the title/name of the web service, I_s represents the service information/description, and W_s represents the WSDL representation of I_s .

D refers to the discovery operation and is represented as,

$$D = \{R_F, R_{NF}\}$$

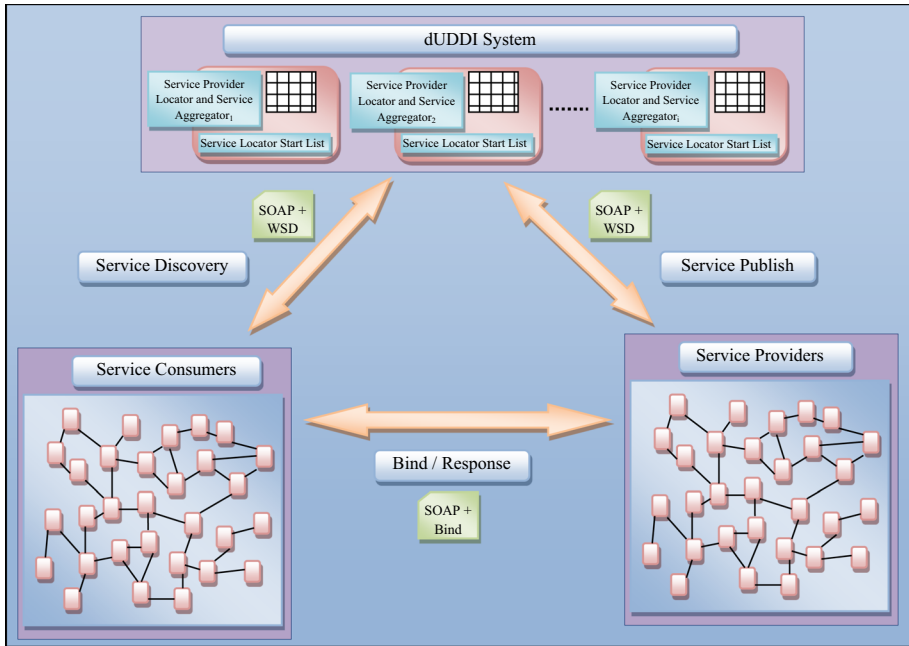


Fig. 4 Distributed UDDI architecture

where, R_F represents the functional requirements or keywords, and R_{NF} represents the non-functional requirements or QoS measures.

3.1.1 Service Consumer

A web service consumer is the actual user of a service. From the perspective of an application, a service consumer can be a service or application or other types of software module that requests a web service. The module triggers the process of discovering the service from the registry, binding to the service transport details, and executing the service. In the proposed model, consumers are the peers who, on demand, request its nearest SPLSA to retrieve the service based on the service requirements embedded in the SOAP message. As a response, the consumer receives a reply SOAP message which contains the graded services that are appropriate to the service requirement specifications. The response also comprises details required to communicate to the provider, such as the addressing of the provider corresponding to the service, service identification number, service versioning, service updation, binding and transport information. Using the details acquired from the SPLSA, the consumer communicates with the service provider which suits its requirements.

3.1.2 Service Provider

Service providers are the nodes that store the service code/ component they provide to their requested consumers. The service provider maintains a catalogue of various services categorized based on different versioning. Though the service providers are organized in a

P2P fashion, they are cooperatively structured in such a way that each provider has a linkage to other providers who provide a similar type of service. The linkage details among providers are managed by SPLSA at the time of service registration by the provider. Each provider has a Link to Providers (LTP) module which holds the details of the providers it is linked with. To publish a service, the provider generates a Web Service Description (WSD) details of the service and communicates the same as a SOAP message to its nearest SPLSA. As a response, the provider will receive a unique service identification number (S_{id}) for the service it has requested to publish. In future, any further operation on the service like updation, versioning or discovery is carried out using the S_{id} assigned to the service. The structure followed to preserve linkage among providers is a Doubly Linked List (DLL). DLL structure is preferred because of the following benefits,

- To traverse 'n' service providers, only 'n' messages are needed even in the worst case.
- The structure can fairly share the load between the peers and reduce bottlenecks.
- The structure has its dual-way organization which helps to improve the robustness and to know which peers have already visited or not.
- Each provider peer forward the service search requests to its linked providers so that all the similar service providers are made to respond to the request.

When a service provider publishes a service, SPLSA will search for a similar type of service in its Service Locator Start List (SLSL). If it found any similar service, it makes any of the existing service providers point to the current publishing provider with its DLL structure constraints. Otherwise, a new record is created in SLSL and the current provider's details are stored as the start node.

Services retrieved from the DUDDI using QoS matchmaking would be represented as a matrix (S_{ij}) of services as,

$$S_{ij} = \begin{bmatrix} q_{11} & q_{12} & q_{13} & \cdots & q_{1j} \\ q_{21} & q_{22} & q_{23} & \cdots & q_{2j} \\ \vdots & \vdots & & & \\ \vdots & \vdots & & & \\ q_{i1} & q_{i2} & q_{i3} & \cdots & q_{ij} \end{bmatrix}$$

where, 'i' refers to the total number of services retrieved from the DUDDI, 'ij' refers to the jth QoS parameter of the ith service and 'q_{ij}' refers to the value of the jth QoS parameter of the ith service.

SPLSA can either be public or private. A public SPLSA, which can be managed by an open group, can register and hold the service information of any providers who are agreeable to publish. Private SPLSA is maintained by any organization which is responsible for the services publishing and maintenance of the published services. Private SPLSA can be used as a service directory within an organization or as commercial purpose service access. A single commercial organization may maintain a collection of SPLSAs, which works mutually, and service can be published in more than one public SPLSA.

3.1.3 dUDDI System

The dUDDI system acts as an intermediate between the service provider and the consumer to publish and discover the service. The dUDDI system consists of a collection of dedicated peers called Service Provider Locator and Service Aggregator (SPLSAs) which

perform the task accomplished by the cUDDI component, with minimum traffic, in a distributed manner. Each SPLSA can operate as a standalone registry or a part of a group registry where coordination among the participant SPLSAs is a critical issue in service publishing and discovery. SLSL of a SPLSA maintains the essential information about the services that are published which is used to locate the service provider on service discovery operation. In SPLSA, published services are segregated using search keywords, and service segregation based on domain, ontology and platform.

4 Components of dUDDI Architecture

4.1 Service Consumer

In general, Service consumers are persons or software entities that discover and invoke the service published by the providers. In dUDDI architecture, important functions of service consumers are,

- perform user authentication and session management
- acquire service request specifications, wrap up into a SOAP message and forward to SPLSA
- display the service search response received from SPLSA
- Error management between the user and SPLSA

Service consumer building blocks, shown in Fig. 5, are,

4.1.1 Authentication Module

This module is responsible for conversion with SPLSA for user authentication and session management.

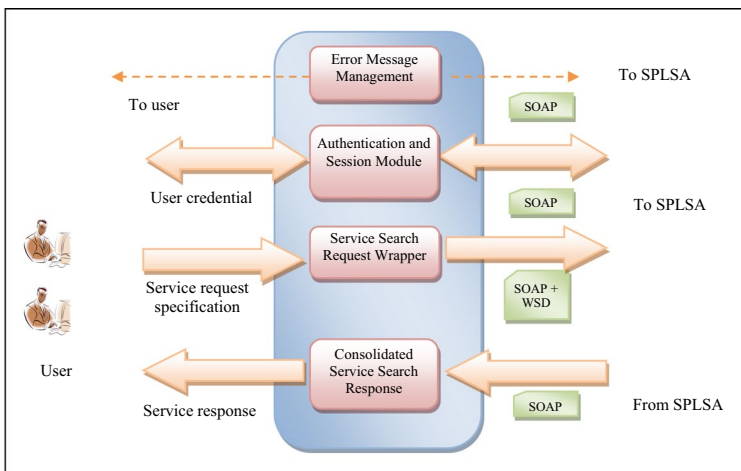


Fig. 5 Internal components of service consumer in dUDDI architecture

4.1.2 Service Request Wrapper

The request wrapper is responsible to retrieve the required service specification details from the user and forward the request in SOAP format to SPLSA.

4.1.3 Service Response

SPLSA will respond to the user request with complete details about the providers whose service specification matches the user's requirements. This module is responsible to interact with the user based on the response received.

4.2 SPLSA

SPLSA is an intermediate between the service requester and the provider. It performs the following tasks,

- Receive the service request from the consumer.
- Session and Authorization management.
- Assigning request ID and message encryption policies.
- Locate the providers who provide the service requested and multicast the request.
- Aggregate the service responses received from different providers based on request ID.
- Reply to the consumer with the consolidated service responses as a single message.
- Error management between the service requester and provider.

SPLSA building blocks, shown in Fig. 6, are,

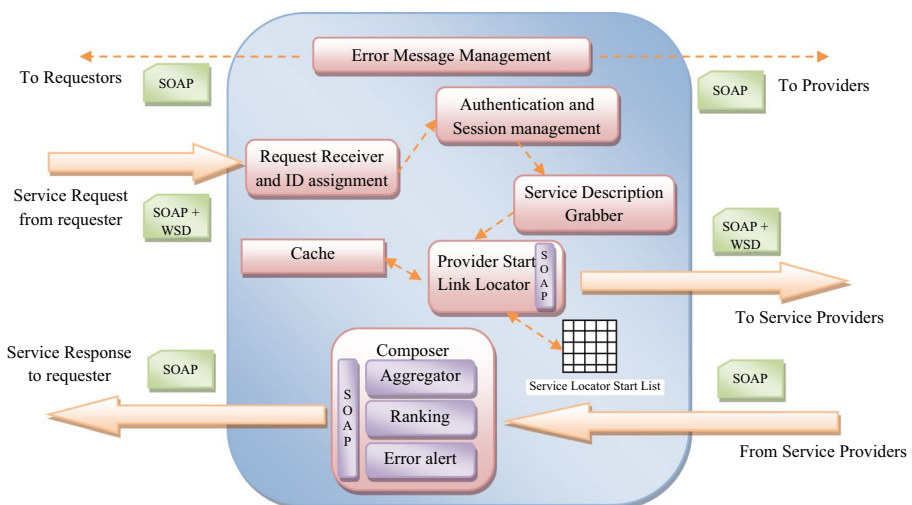


Fig. 6 Internal components of service SPLSA in dUDDI architecture

4.2.1 Service Description Grabber

This module performs pooling at a specific address at which the consumer sends the request either through the browser or application and assigns a service Identification (SID) number. The request is received in the SOAP message format extracts the service description or search keyword part of the message and forwards it to the next module.

4.2.2 Authentication and Session management

This module performs authentication of consumers with credentials (say username and password). Authentication is performed using the user credential database which is updated while the user registers on SPLSA. If the authentication fails then an Error message is sent to the service requested consumer otherwise a fresh session with unique Session identification (seID) is created and seID is sent to the user. This module plays an essential role in the security and operational integrity aspects of the model.

4.2.3 Provider Start Link Locator (PSLL)

This module performs the task to identify the service providers who offer the service that matches with service description or search keywords extracted from the request message. A sub-module, Service Locator Start List (SLSL) is a database used to link the providers offering similar types of services. SLSL consist of fields such as service ID, service description and Start Link List (SLL) as shown in Fig. 7. SLL contains the link (probably the IP and port addresses of providers) to the first few service providers matching the requested service and request message is forwarded to them. The cache module store the frequently accessed service details to speed up the repeated and related requests. This module also works with data consistency techniques to evade obsolete data responses.

4.2.4 Aggregator and Ranking

This module receives the service reply from the various service providers and ranks the result services (on demand) based on the matching with the search criteria. It consolidates the received or ranked services into a single message and forwards it to the service requester.

Service ID	Description	Keyword or Criteria	Start Link List			
				SP ₁	SP ₂	SP ₃	SP _n

Fig. 7 Service locator start list structure

4.3 Service Provider

The service provider is any host that implements a web service and makes it available on the Internet. The service provider module, shown in Fig. 8, consists of the following blocks,

4.3.1 ID Repetition Check

When the provider receives a request from SPLSA, this module checks whether this service request is already received to avoid repeated processing of the same request.

4.3.2 Parse through Service Description

The request is received in the SOAP message format and this module extracts the service description or search keyword part of the message. The service description is parsed to extract the keywords and send it to the service selection module of the provider.

4.3.3 Service Selection

This module performs the task to prefer the service that suits the service requested based on the parsed keywords.

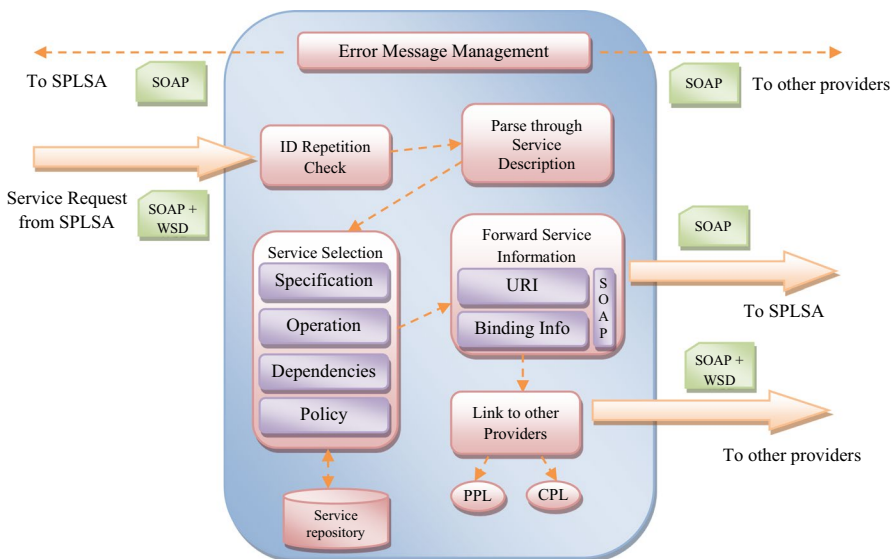


Fig. 8 Internal components of service provider in dUDDI architecture

4.3.4 Forwarding Service Information

This module creates a SOAP message which is embedded with details of providers and the service chosen for requested criteria. This SOAP message is forwarded to SPLSA to promote it to the consumer.

4.3.5 Link to other Providers

This module has links for other service providers who offer a service similar to that of the current service request.

4.3.6 Error message management

This module takes control of all the error notifications during the communication between the SPLSA and other providers to offer the integrity of the search procedure and service selection.

5 Operations of dUDDI Architecture

The process of working on the two important UDDI operations namely service publishing and discovery are discussed in this section.

5.1 Web Service Publishing

Publishing a Web service means enabling a Web service consumer to locate the service description and instructing the consumer on how they should interact with the Web service. The service publish process takes place, as shown in Fig. 9, with the following steps,

Step 1 The publisher who wants to publish a service need to provide the admin credentials which should be registered with the SPLSA.

Step 2 SPLSA receives the request SOAP message and authenticates it for a genuine user. If authentication fails an error message is replied to the requestor otherwise a new

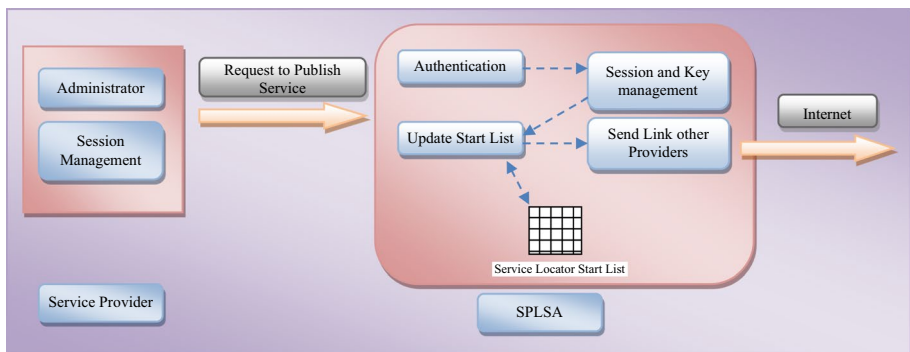


Fig. 9 Web service publish operation in dUDDI architecture

session is created with a unique ID. This session is maintained till the whole service publish process completes.

Step 3 SPLSA assigns an ID to the request and extracts the service description from the request SOAP message received. The extracted service description is parsed to extract the keywords.

Step 4 PSLL module of SPLSA use the extracted keywords to search against the keywords column of SLSL. If no match is found, then the extracted keywords are generalized and the search operation is repeated.

Step 5 If a keyword match is found in SLSL, then the service providers' details (IP and port addresses) are retrieved from the SLL column corresponding to the matched keyword column.

Step 6 SPLSA create SOAP message by replicating the received request message embed service ID and message hop count (based on several providers index since only one hop is required for one provider), then forward it to the service providers retrieved.

5.2 Web Service Discovery

Service Discovery is the process of identifying the service offered by the providers that fulfil the requirements of the consumer. Service discovery takes place, as shown in Fig. 10, with the following steps,

Step 1 The service consumer creates a SOAP message with the required service description and forwards it to SPLSA through a browser or application.

Step 2 SPLSA receives the request SOAP message and authenticates it for the true user. If authentication fails an error message is replied to the requestor otherwise a new session is created with a unique ID. This session is maintained till the whole discovery process completes.

Step 3 SPLSA assigns an ID to the request and extracts the service description from the request SOAP message received. The extracted service description is parsed to extort the keywords.

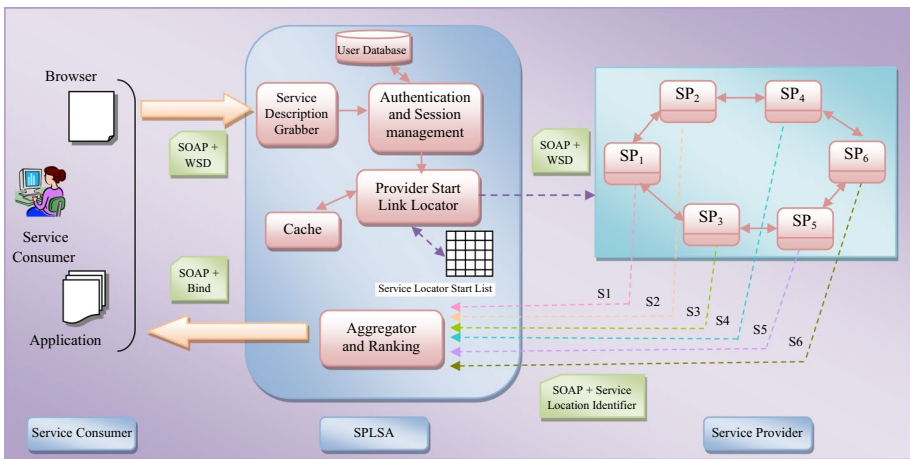


Fig. 10 Web service discovery operation in dUDDI architecture

Step 4 PSL module of SPLSA use the extracted keywords to search against the keywords column of SLSL. If no match is found, then the extracted keywords are generalized and the search operation is repeated.

Step 5 If a keyword match is found in SLSL, then the service providers' details (IP and port addresses) are retrieved from the SLL column corresponding to the matched keyword column.

Step 6 SPLSA create SOAP message by replicating the received request message embed service ID and message hop count (based on the number of providers index), then forward it to the service providers retrieved.

Step 7 The service provider receives the SOAP formatted service request message from SPLSA and checks whether the same service request ID is already received or not to avoid repeated processing of the same request.

Step 8 The provider extracts the keyword from the request message and selects a service that matches the consumer's requirement. Then, the provider creates a response SOAP message and includes the details of matched service (like URL) and provider constraints. This response message is forwarded to SPLSA from where it received the request message.

Step 9 After sending the response message, the Link to Providers (LtP) module identifies the other providers which linked with the current provider for the requested service and forwards the request message to them. This message is forwarded to all the linked providers (both Parent Provider Link and Child Provider Links) except where it received the request message.

Step 10 From Step 7 to Step 9 is repeated till the provider's index or hop count embedded with the message expires.

Step 11 SPLSA receives the reply from the different service providers and ranks them based on the keywords on the request and the response. These ranked replies are consolidated into a single SOAP-formatted response message and forwarded to the service requester.

In the process of service discovery, conversations among the various entities are carried out in SOAP message format.

6 Experimentation and Result Analyses

6.1 Experimental Setup

To evaluate the proposed model, the web service selection operation has been accomplished on the services which are retrieved from the repository on the user-preferred QoS parameter ranges. The system used for implementation purposes is Intel i3 processor with a 16 GB RAM machine connected to a 100 MB/s Ethernet, Windows 10 and IDE Eclipse to retrieve the services. And the optimization process was done in MATLAB for the effective use of the web services.

In the service oriented architecture, a web service composed of a collection of measures corresponds to the QoS element of the service. The service set S be the collection of 'u' services from the UDDI and each service consists of a 'q' number of QoS factors,

$$S = \{s_1, s_2, \dots, s_u\}$$

$$S_k = \{Q_1, Q_2, Q_3, Q_4 \dots, Q_q\}$$

where, u is the total number of services in the UDDI, q is the total number of QoS parameters for the service k in S .

In the current scenario, there is no standard web service discovery testing methodology and so, a list of real-time web service provider are gathered and registered with the UDDI. A testbed has been generated consisting of 1000 web services of various domains and services are manually divided into domains such as Airlines, Tourist, Automobile, Postal, Banking, Bioinformatics, News, Conversion, Search, Weather, Dictionary, Education, Employment, Entertainment, Financial and Social Networking. Each domain will have 21 QoS parameters like Penalty (P), Incentives (I), Service Reputation (SR), Service Provider Reputation (SPR), Response Time (RT), Throughput (T), Controllability (CN), Availability (A), Accessibility (AC), Non-Repudiation (NR), Successability (SU), Standard Adoptability (SA), Standard Conformability (SC), Reliability (R), Transaction Integrity (TI), Informability (IN), Authentication (AT), Authorization (AZ), Collaborability (C) and Privacy (PR).

To improve the precision of the experiments, the scenario considered for the experiments consists of a collection of different classes as in Table 1. Each class contains a different combination of QoS factors deliberated for the performance evaluation. Based on the type of class used for experiments, service requestors are allowed to specify the range of QoS factors corresponding to the class used for the particular scenario.

6.2 Assessment Criteria

The following are some of the performance factors (adapted from [24]) used to evaluate the performance of the proposed web service model, as the evaluation metrics of web service retrieval are similar to the information retrieval scheme. The relevance of a retrieved web service can be measured based on the factors like relatedness, topicality, beneficiality and utility. In the proposed model, we have considered relevance in two directions topical and user utility relevance. The relevance of the service based on the topical relevance is determined using the keywords provided by the consumer in the service search query and the

Table 1 List of various classes and their QoS factors combination

Class	Combination of QoS factors
Class-I	A
Class-II	P
Class-III	R,RT
Class-IV	A,R,P
Class-V	A,R,SR,SPR,AC,RT
Class-VI	R,T,I,SA,SC,TI,P
Class-VII	R,I,SR,TI,C,IN,CN,P
Class-VIII	AC,TI,C,IN,CN,AT,AZ,NR,RT,P
Class-IX	I,SR,SPR,AC,SU,SA,SC,TI,C,IN,CN,AT,AZ,NR,PR,P
Class-X	A,R,T,I,SR,SPR,AC,SU,SA,SC,TI,C,IN,CN,AT,AZ,NR,R,T,PR,P

service domain. Relevance measurement based on the user utility is performed based on different QoS parameters considered in the experiments.

6.2.1 Precision (P)

Precision can be referred to as the part of discovered web services that are appropriate to the user's requirements. The formula for the same can be expressed as,

$$Precision = \frac{|S_{Relevant} \cap S_{Retrieved}|}{|S_{Retrieved}|} \quad (6.1)$$

where, $|S_{Relevant}|$ is the total number of services that are relevant to the request. $|S_{Retrieved}|$ is the total number of services that are retrieved.

6.2.2 Recall (R)

The recall is the number of relevant services that are fetched to the user's requirements. The formula for the same can be expressed as,

$$Recall = \frac{|S_{Relevant} \cap S_{Retrieved}|}{|S_{Relevant}|} \quad (6.2)$$

where, $|S_{Relevant}|$ is the total number of services that are relevant to the request. $|S_{Retrieved}|$ is the total number of services that are retrieved.

6.2.3 F-Measure

F-measure can be defined as the harmonic mean of precision and recall. The formula for the same can be expressed as,

$$F - Measure = \frac{2 * Precision * Recall}{Precision + Recall} \quad (6.3)$$

6.3 Result Analysis

In this section, the efficacy of the proposed dUDDI architecture along with the recent and best working decentralized UDDI models referred to in [23, 24] and [27] under similar experimental setups is discussed concerning the appropriate set of performance criteria as discussed in Sect. 6.2. Table 2 illustrates the performance of different decentralized UDDI models on the capability of service retrieval from the UDDIs under various scenarios considered for the experiments. This experimental result is rudimentary based on which the results for various performance factors are measured and justified.

6.3.1 Precision

The evaluation based on the factor precision reveals the ability of the decentralized UDDI models on retrieving the collection of services that are exactly requested by the service requestors to the total number of services retrieved from the UDDI considering the demand

Table 2 Performance in service retrieval of different decentralized UDDI models

S. no	Class of QoS factors	No. of services	No. of feasible services	No. of services retrieved			No. of relevant services retrieved				
				Existing decentralized UDDI models		Proposed model	Existing decentralized UDDI Models		Proposed model		
				[23]	[27]	[24]	[23]	[27]	[24]		
1	Class-I	100	78	98	86	85	79	73	78	78	
		500	421	432	432	423	423	423	414	417	419
		1000	795	833	804	807	800	800	785	789	792
2	Class-II	100	78	108	92	81	81	81	73	74	76
		500	442	472	470	456	452	452	436	437	438
		1000	836	841	842	827	836	836	807	817	822
3	Class-III	100	71	99	79	75	72	72	63	65	71
		500	411	441	421	416	414	414	401	405	407
		1000	761	803	781	775	766	766	749	755	759
4	Class-IV	100	70	99	80	77	75	75	59	63	67
		500	398	428	418	408	408	408	382	389	396
		1000	792	824	817	816	809	809	743	763	776
5	Class-V	100	62	97	75	74	63	63	47	55	58
		500	346	352	348	353	347	347	315	321	332
		1000	713	740	734	717	716	716	654	689	707
6	Class-VI	100	68	92	78	79	74	74	54	58	63
		500	317	352	341	332	327	327	301	305	311
		1000	746	768	763	748	747	747	691	724	735
7	Class-VII	100	63	79	74	71	68	68	51	56	58
		500	361	389	381	376	370	370	327	339	355
		1000	718	735	736	731	720	720	641	679	691

Table 2 (continued)

S. no	Class of QoS factors	No. of services	No. of feasible services	No. of services retrieved			No. of relevant services retrieved				
				Existing decentralized UDDI models		Proposed model	Existing decentralized UDDI Models		Proposed model		
				[23]	[27]	[24]	dUDDI	[23]	[27]	[24]	dUDDI
8	Class-VIII	100	51	77	71	65	58	41	44	47	51
		500	312	342	334	323	319	298	302	307	312
		1000	669	697	681	676	673	614	651	659	668
9	Class-IX	100	44	71	63	57	51	36	39	40	43
		500	303	345	331	320	312	252	271	292	301
		1000	601	651	639	624	614	537	561	590	598
10	Class-X	100	39	68	60	49	45	21	28	35	37
		500	243	278	266	261	252	211	221	236	240
		1000	578	621	601	591	583	520	543	568	573

QoS factors. The experimental results for the various techniques w.r.t the precision factor and the improvement rate is shown in Table 3.

In the perception of the precision assessment factor, the proposed dUDDI model performs better than the various existing decentralized UDDI models considered for evaluation. It can be observed that the proposed dUDDI model obtained more than 90% precision achieved up to class VII of QoS factors regardless of the number of services considered in the UDDI. On the other hand, the precision value of [24, 27], and [23] drops below the 90% mark at the class IV, II and I of QoS combinations respectively. In the case of the proposed dUDDI model, the least precision value obtained is 82.22% for 100 services

Table 3 Experimental results for the various decentralized UDDI models w.r.t the Precision factor

S. no	Class of QoS factors	Services considered	Precision (%)				Improvement rate (%)		
			Existing decentralized UDDI models			Proposed model dUDDI	Over [23]	Over [27]	Over [24]
			[23]	[27]	[24]				
1	Class-I	100	74.49	90.70	91.76	98.73	21.76	1.18	7.59
		500	95.83	96.53	99.05	99.53	0.72	2.62	0.48
		1000	94.24	98.13	98.14	99.38	4.13	0.01	1.26
2	Class-II	100	67.59	80.43	93.83	96.30	19.00	16.65	2.63
		500	92.37	92.98	96.05	97.79	0.66	3.31	1.81
		1000	95.96	97.03	98.88	99.09	1.12	2.94	1.21
3	Class-III	100	63.64	82.28	94.67	98.61	29.29	15.06	4.17
		500	90.93	96.20	97.84	99.28	5.80	1.70	1.47
		1000	93.28	96.67	97.94	99.22	3.64	1.31	1.31
4	Class-IV	100	59.60	78.75	87.01	92.00	32.14	10.49	5.73
		500	89.25	93.06	97.06	97.55	4.27	4.29	0.51
		1000	90.17	93.39	95.10	97.90	3.57	1.83	2.94
5	Class-V	100	48.45	73.33	78.38	98.41	51.35	6.88	25.56
		500	89.49	92.24	94.05	99.71	3.08	1.96	6.02
		1000	88.38	93.87	98.61	99.58	6.21	5.05	0.99
6	Class-VI	100	58.70	74.36	79.75	91.89	26.69	7.25	15.23
		500	85.51	89.44	93.67	96.94	4.60	4.73	3.49
		1000	89.97	94.89	98.26	99.73	5.46	3.56	1.50
7	Class-VII	100	64.56	75.68	81.69	92.65	17.22	7.95	13.41
		500	84.06	88.98	94.41	97.57	5.85	6.11	3.34
		1000	87.21	92.26	94.53	99.72	5.78	2.46	5.49
8	Class-VIII	100	53.25	61.97	72.31	87.93	16.39	16.68	21.61
		500	87.13	90.42	95.05	97.81	3.77	5.12	2.90
		1000	88.09	95.59	97.49	99.26	8.52	1.98	1.82
9	Class-IX	100	50.70	61.90	70.18	84.31	22.09	13.36	20.15
		500	73.04	81.87	91.25	96.47	12.09	11.45	5.73
		1000	82.49	87.79	94.55	97.39	6.43	7.70	3.01
10	Class-X	100	30.88	46.67	71.43	82.22	51.11	53.06	15.11
		500	75.90	83.08	90.42	95.24	9.46	8.83	5.33
		1000	83.74	90.35	96.11	98.28	7.90	6.37	2.26

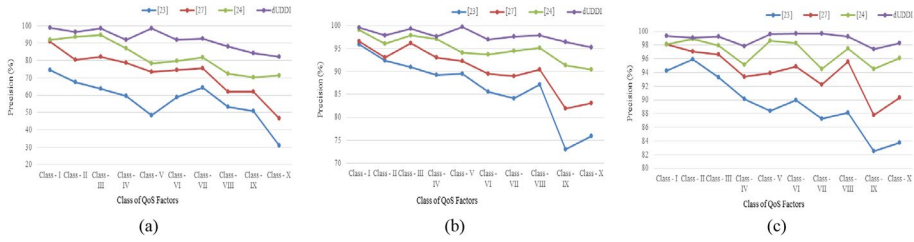


Fig. 11 Precision based performance analyses on different decentralized UDDI Models

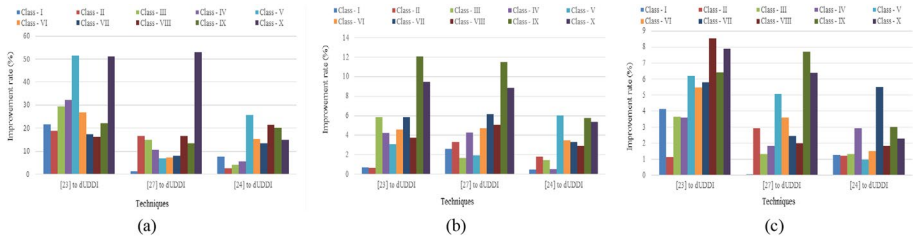


Fig. 12 Precision based performance improvement rate on different decentralized UDDI Models

with Class X of QoS factors combination which stands much superior to the next better 71.43% for the same scenario setup. The precision-based performance analyses on different UDDI models are shown in Fig. 11 which illustrates that the proposed model outperforms the recent and the best working decentralized UDDI models regardless of the type of QoS combination class and the number of services in the concerned UDDI. The three parts of Fig. 11a, b and c represents the precision value of the models with 100, 500 and 1000 services in the UDDI respectively.

To demonstrate the significance of the proposed model, the improvement in the performance of the proposed model over the existing techniques has been tabulated in Table 3. From the table, it can be evidenced that the proposed model shows significant improvement w.r.t the existing models. At maximum, the proposed model obtained 51.35% over the model in [23] for Class V with 100 services, 53.06% over the model in [27] for Class X with 100 services, and 25.56% over the model in [24] for the Class V with 100 services. This justifies that the proposed model shows a substantial improvement in the performance of the web service retrieval over the existing models despite the size of the UDDI and the complex combination of the QoS factors provided for the service requestors. Precision-based performance improvement rate on different models is shown in Fig. 12. The three parts of the Fig. 12a, b and c represent the precision performance improvement rate of the model with 100, 500 and 1000 services in the decentralized UDDI respectively.

6.3.2 Recall (R)

The experimental results for the various techniques w.r.t the Recall factor and the improvement rate are shown in Table 4. From Table 4, it can be perceived that the proposed model achieved 100% of recall for 20 sets of experiments whereas the existing models in [24, 27] and [23] were able to obtain only 2, 1, and 0 sets of experiments respectively. The least recall value obtained for the proposed model is 94.87% for 100

Table 4 Experimental results for the various decentralized UDDI techniques w.r.t the Recall factor

S. no	Class of QoS factors	Services considered	Recall (%)				Improvement Rate (%)		
			Existing decentralized UDDI models			Proposed model dUDDI	Over [23]	Over [27]	Over [24]
			[23]	[27]	[24]				
1	Class-I	100	93.59	100.0	100.00	100.00	6.85	0.00	0.00
		500	98.34	99.05	99.52	100.00	0.72	0.48	0.48
		1000	98.74	99.25	99.62	100.00	0.51	0.38	0.38
2	Class-II	100	93.59	94.87	97.44	100.00	1.37	2.70	2.63
		500	98.64	98.87	99.10	100.00	0.23	0.23	0.91
		1000	96.53	97.73	98.80	99.88	1.24	1.10	1.09
3	Class-III	100	88.73	91.55	100.00	100.00	3.17	9.23	0.00
		500	97.57	98.54	99.03	100.00	1.00	0.49	0.98
		1000	98.42	99.21	99.74	99.87	0.80	0.53	0.13
4	Class-IV	100	84.29	90.00	95.71	98.57	6.78	6.35	2.99
		500	95.98	97.74	99.50	100.00	1.83	1.80	0.51
		1000	93.81	96.34	97.98	100.00	2.69	1.70	2.06
5	Class-V	100	75.81	88.71	93.55	100.00	17.02	5.45	6.90
		500	91.04	92.77	95.95	100.00	1.90	3.43	4.22
		1000	91.73	96.63	99.16	100.00	5.35	2.61	0.85
6	Class-VI	100	79.41	85.29	92.65	100.00	7.41	8.62	7.94
		500	94.95	96.21	98.11	100.00	1.33	1.97	1.93
		1000	92.63	97.05	98.53	99.87	4.78	1.52	1.36
7	Class-VII	100	80.95	88.89	92.06	100.00	9.80	3.57	8.62
		500	90.58	93.91	98.34	100.00	3.67	4.72	1.69
		1000	89.28	94.57	96.24	100.00	5.93	1.77	3.91
8	Class-VIII	100	80.39	86.27	92.16	100.00	7.32	6.82	8.51
		500	95.51	96.79	98.40	100.00	1.34	1.66	1.63
		1000	91.78	97.31	98.51	99.85	6.03	1.23	1.37
9	Class-IX	100	81.82	88.64	90.91	97.73	8.33	2.56	7.50
		500	83.17	89.44	96.37	99.34	7.54	7.75	3.08
		1000	89.35	93.34	98.17	99.50	4.47	5.17	1.36
10	Class-X	100	53.85	71.79	89.74	94.87	33.33	25.00	5.71
		500	86.83	90.95	97.12	98.77	4.74	6.79	1.69
		1000	89.97	93.94	98.27	99.13	4.42	4.60	0.88

services with Class X of QoS factors combination and the next best-least recall value registered by the [24] model is 89.74 for 100 services with Class X. Thus, based on the observation of the recall measures, the proposed dUDDI model outperforms the various existing decentralized UDDI models considered for evaluation regardless of the type of QoS combination class and the number of services in the concerned UDDI and the same has been shown in the Fig. 13. The three parts of the Fig. 13a, b and c represents the recall performance of the models with 100, 500 and 1000 number of services in the UDDI respectively.

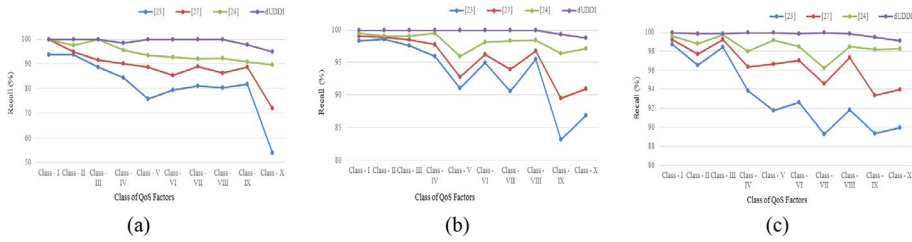


Fig. 13 Recall based performance analyses on different decentralized UDDI Models

Table 4 also exemplifies the performance improvement rate of the proposed model over the existing techniques. From the table, it can be justified that the proposed model shows a significant improvement rate w.r.t the best working decentralized UDDI models. At maximum, the proposed model obtained 33.33% over the [23] for Class X with 100 services, 25.00% over the [27] for Class X with 100 services, and 8.62% over the [24] for Class VII with 100 services. The recall-based performance improvement rate on the proposed model w.r.t the different models is shown in Fig. 14. The three parts of Fig. 14a, b and c represents the recall performance improvement rate of the models with 100, 500 and 1000 number of services in the UDDI respectively. From the figure, it can be identified that the proposed model shows a significant improvement in the performance of the relevant web service retrieval over the existing models despite the size of the UDDI and the complex combination of the QoS factors provided for the service requestors.

6.3.3 F-Measure

The experimental results for the various techniques w.r.t the F-measure factor and the improvement rate are shown in Table 5. The table illustrates that the proposed model achieved greater than 90% of the f-measure value for all the classes of QoS factors regardless of the number of services considered in the UDDI. On the other hand, the f-measure value of [24, 27] and [23] drops below the 90% mark at the class IV, II and I of QoS combinations respectively. The least f-measure value obtained for the proposed model is 90.10% for 100 services with Class X of QoS factors combination and the next best-least f-measure value registered by the [24] model is 79.21% for 100 services with Class IX. Thus, it can be claimed that the proposed dUDDI model overtakes the various existing decentralized UDDI models considered for evaluation irrespective of the type of QoS combination class and the same has been shown in Fig. 15. The three parts of Fig. 15a, b and c represents

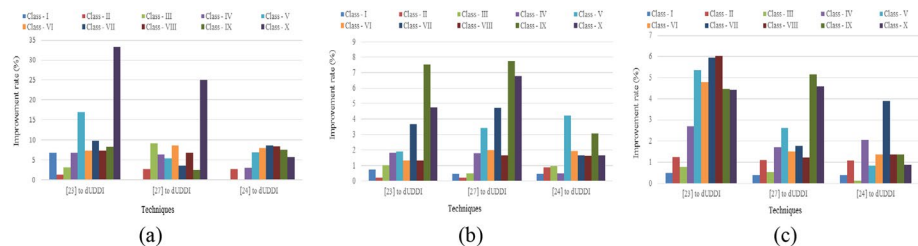


Fig. 14 Recall based performance improvement rate on different decentralized UDDI Models

Table 5 Experimental results for the various decentralized UDDI techniques w.r.t the F-Measure factor

S. no	Class of QoS factors	Services considered	F-Measure (%)				Improvement Rate (%)		
			Existing decentralized UDDI models			Proposed model dUDDI	Over [23]	Over [27]	Over [24]
			[23]	[27]	[24]				
1	Class-I	100	82.95	95.12	95.71	99.36	14.6	0.61	3.82
		500	97.07	97.77	99.29	99.76	0.72	1.55	0.48
		1000	96.44	98.69	98.88	99.69	2.33	0.19	0.82
2	Class-II	100	78.49	87.06	95.60	98.11	10.9	9.81	2.63
		500	95.40	95.83	97.55	98.88	0.45	1.79	1.36
		1000	96.24	97.38	99.34	100	1.18	2.01	1.15
3	Class-III	100	74.12	86.67	97.26	99.30	16.9	12.2	2.10
		500	94.13	97.36	98.43	99.64	3.43	1.10	1.23
		1000	95.78	97.92	98.83	99.54	2.24	0.92	0.72
4	Class-IV	100	69.82	84.00	91.16	95.17	20.3	8.52	4.41
		500	92.49	95.34	98.26	98.76	3.08	3.06	0.51
		1000	91.96	94.84	96.52	98.94	3.14	1.77	2.51
5	Class-V	100	59.12	80.29	85.29	99.20	35.8	6.23	16.3
		500	90.26	92.51	94.99	99.86	2.49	2.69	5.12
		1000	90.02	95.23	98.88	99.79	5.79	3.83	0.92
6	Class-VI	100	67.50	79.45	85.71	95.77	17.7	7.88	11.74
		500	89.99	92.71	95.84	98.45	3.02	3.38	2.72
		1000	91.28	95.96	98.39	99.80	5.12	2.54	1.43
7	Class-VII	100	71.83	81.75	86.57	96.18	13.8	5.89	11.11
		500	87.20	91.37	96.34	98.77	4.79	5.43	2.52
		1000	88.23	93.40	95.38	99.86	5.86	2.12	4.70
8	Class-VIII	100	64.06	72.13	81.03	93.58	12.5	12.34	15.48
		500	91.13	93.50	96.69	98.89	2.60	3.42	2.27
		1000	89.90	96.44	97.99	99.55	7.28	1.61	1.59
9	Class-IX	100	62.61	72.90	79.21	90.53	16.4	8.66	14.29
		500	77.78	85.49	93.74	97.89	9.91	9.65	4.42
		1000	85.78	90.48	96.33	98.44	5.48	6.46	2.19
10	Class-X	100	39.25	56.57	79.55	90.10	44.1	40.63	10.75
		500	81.00	86.84	93.65	96.97	7.21	7.85	3.54
		1000	86.74	92.11	97.18	98.71	6.19	5.50	1.58

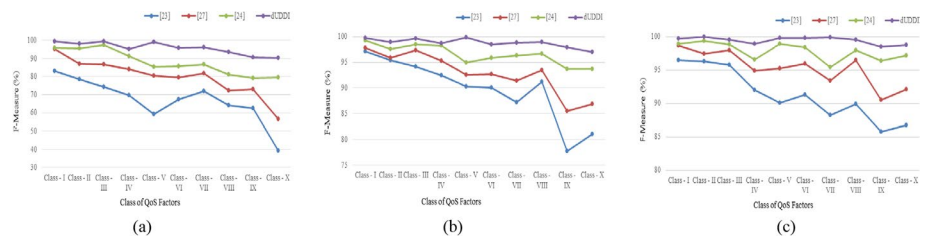


Fig. 15 F-Measure based performance analyses on different decentralized UDDI Models

the f-measure performance of the models with 100, 500 and 1000 number of services in the UDDI respectively.

From Table 4, it can also be vindicated that the proposed model shows a significant improvement rate w.r.t the models considered for evaluation. At maximum, the proposed model obtained 44.1% over the [23] for Class IX with 100 services, 40.63% over the [27] for Class IX with 100 services, and 15.48% over the [24] for Class VIII with 100 services. The f-measure-based performance improvement rate on the proposed model w.r.t the different other models is shown in Fig. 16. The three parts of the Fig. 16a, b and c represents the f-measure performance improvement rate of the models with 100, 500 and 1000 services in the UDDI respectively. From the figure, it can be identified that the proposed model shows a significant improvement in the performance of the web service retrieval over the existing models despite the size of the UDDI and the complex combination of the QoS factors provided for the service requestors.

7 Summary

In summary, an experiment was designed and carried out to examine the proposed dUDDI model for various performance attributes. The preliminary results suggest that the proposed model outperforms the existing and best-working decentralized UDDI models concerning the performance factors considered. The interesting fact to be noted is that the proposed model shows extraordinary performance with an increase in the number of services deployed in the UDDI. This significant characteristic enables the proposed model suitable for the large-scale web service environment to adopt the model for high-level performance enhancements.

8 Conclusion

A complete study of web services and the various issues involved in emerging technology has been discussed. One of the important issues is centralized UDDI architecture, which leads single point of failure and traffic overhead, which is considered in this work. In this paper, we presented a distributed UDDI (dUDDI) architecture and also discussed various internal components of dUDDI elements. We also presented an efficient algorithm for web service publishing and discovery operations in the dUDDI architecture. The proposed dUDDI model is evaluated with the best-in-class decentralized UDDI models by considering different conditions like the registry size, QoS

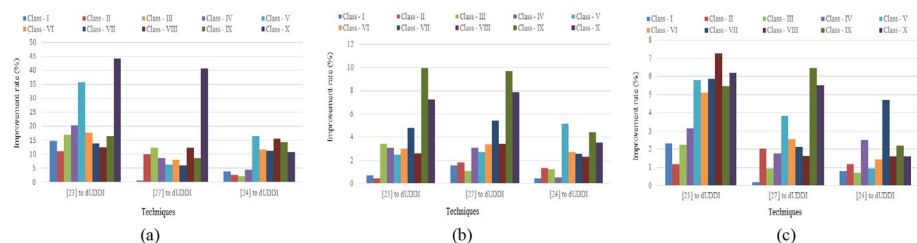


Fig. 16 F-Measure based performance improvement rate on different decentralized UDDI Models

factors and discovery of the relevant services based on user request. A testbed has been generated consisting of 1000 web services of various domains and services are manually divided into 21 domains with different QoS requirement combinations. The experimentation results justify that the proposed model outperforms the existing decentralized UDDI models in terms of precision, recall and f-measure factors.

Author Contributions The first draft of the manuscript was written by Dr. PVP and other authors commented on the versions of the manuscript. All authors contributed to the study conception and design.

Funding The authors did not receive support from any organization for the submitted work.

Data Availability Data sharing does not apply to this article as no datasets were generated or analyzed during the current study.

Code Availability The work is implemented specific to the requirement of the research and code can be accessed by communicating to the corresponding author.

Declarations

Conflicts of interest The authors have no conflicts of interest to declare that are relevant to the content of this article.

References

1. Shang, K. (2020). Semantic-based service discovery in grid environment. *Journal of Intelligent & Fuzzy Systems*, 39(4), 1–10.
2. Yulin, N., Huayou, S., Weiping, L. & Zhong, C. (2010). PDUS: P2P-based distributed UDDI service discovery approach. In *International IEEE conference on service sciences*. <https://doi.org/10.1109/ICSS.2010.48>, pp.3-8
3. Wang, Z. & Hu, Y. (2007). A P2P network based architecture for web service. In *International conference on wireless communications, networking and mobile computing (WiCom)*, Shanghai, pp. 3446–3449.
4. Du, Z., Huai, J., & Liu, Y. (2006). Ad-UDDI: An active and distributed service registry. *LNCs technologies for e-services* (pp. 58–71). Springer.
5. Lv, W. & Yu, J. (2007). pService: Peer-to-peer based web services discovery and matching. In *Second international conference on systems and networks communications (ICSNC)*.
6. Verma, R., & Srivastava, A. (2018). A dynamic web service registry framework for mobile environments. *Peer-to-peer networking and Applications*, 11(3), 409–430.
7. Natarajan, B., Obaidat, M. S., Sadoun, B., Manoharan, R., Ramachandran, S., & Velusamy, N. (2020). New clustering-based semantic service selection and user preferential model. *IEEE Systems Journal*, 15(4), 4980–4988.
8. Agarwal, N., Sikka, G. & Awasthi, L. K. (2020). Web service clustering approaches to enhance service discovery: A review. In *The international conference on recent innovations in computing*, pp. 23–35. Springer, Singapore.
9. Chandrasekaran, S., Srinivasan, V. B., & Parthiban, L. (2018). Efficient Web Service Discovery and Selection Model. *International Journal on Future Revolution in Computer Science & Communication Engineering*, 4(2), 01–05.
10. Ayorak, E. & Bener, A.B. (2007). Super peer web service discovery architecture. In *IEEE 23rd international conference on data engineering (ICDE 2007)*, pp.1360–1364. <https://doi.org/10.1109/ICDE.2007.369011>
11. D’Mello, D. A. & Ananthanarayana V. S. (2009). A tree structure for efficient web service discovery. In *IEEE second international conference on emerging trends in engineering and technology, ICETET-09*, pp. 826–831.

12. Feng, Y. & Li, Q. (2016). The distributed UDDI system model based on service oriented architecture. In *7th IEEE international conference on software engineering and service science (ICSESS)*, pp. 585–589. IEEE.
13. Maamar, Z., Yahyaoui, H., & Mahmoud, Q. H. (2007). Dynamic management of UDDI registries in a wireless environment of web services: Concepts, architecture, operation, and deployment. *Journal of Intelligent Information Systems*, 28(2), 105–131.
14. Bhuvaneswari, A., & Karpagam, G. R. (2018). Semantic web service discovery for mobile web services. *International Journal of Business Intelligence and Data Mining*, 13(1–3), 95–107.
15. Seghir, N. B., Kazar, O., Rezeg, K., & Bourekache, S. (2017). A semantic web services discovery approach based on a mobile agent using metadata. *International Journal of Intelligent Computing and Cybernetics*, 10(1), 12–29.
16. Al-Masri, E. & Mahmoud, Q. H. (2007). Crawling multiple UDDI business registries. In *16th international conference on World Wide Web (WWW '07)*, ACM New York, pp.1255–1256.
17. Victor Paul, P., Rajaguru, D., Saravanan, N., Baskaran, R., & Dhavachelvan, P. (2013). Efficient service cache management in mobile P2P networks. *Future Generation Computer Systems*, 29(6), 1505–1521.
18. Shadjia, D., Mo R. & Hill, R. (2017). Towards an understanding of microservices. In *23rd international conference on automation and computing (ICAC)*, pp. 1–6. IEEE.
19. Sioutas, S., Sakkopoulos, E., Drossos, L., & Sirmakessis, S. (2008). Balanced distributed web service lookup system. *Journal of Network and Computer Applications, Elsevier*, 31(2), 149–162.
20. Jo, S., Lee, J., Han, J., & Ghose, S. (2020). P2P computing for intelligence of things. *Peer-to-Peer Networking and Applications*, 13(2), 575–578.
21. Hayyolalam, V., & Kazem, A. A. P. (2018). A systematic literature review on QoS-aware service composition and selection in cloud environment. *Journal of Network and Computer Applications*, 110, 52–74.
22. Rajendran, V. & Mani, P. (2021). Study of bond-agent-based resource discovery in cloud computing. In *Proceedings of international conference on communication and computational technologies*, pp. 607–615. Springer, Singapore.
23. Zhang, Y., He, H. & Teng, J. (2013). Chord-based semantic service discovery with QoS. In *fifth international conference on measuring technology and mechatronics automation*, pp. 365–367. IEEE.
24. Briola, D., Micucci, D., & Mariani, L. (2019). A platform for P2P agent-based collaborative applications. *Software Practice and Experience*, 49(3), 549–558.
25. Seghir, N. B. & Kazar, O. (2017). A new framework for web service discovery in distributed environments. In *First international conference on embedded & distributed systems (EDiS)*, pp. 1–6. IEEE.
26. Baresi, L., Miraz, M., & Plebani, P. (2016). A distributed architecture for efficient Web service discovery. *Service Oriented Computing and Applications*, 10(1), 1–17.
27. Seghier, N. B., & Kazar, O. (2020). Mobile agent and ontology approach for web service discovery using QoS. *International Journal of Reasoning-based Intelligent Systems*, 12(1), 17–33.
28. Almalki, J. & Shen, H. (2018). SORCER: A decentralised continuous integration platform for service-oriented software systems. In *International conference on service-oriented computing*, pp. 458–464. Springer, Cham.
29. Seghir, N. B., Kazar, O., & Rezeg, K. (2018). A personalized approach for web service discovery in distributed environments. *Handbook of research on contemporary perspectives on web-based systems* (pp. 308–339). IGI Global.
30. Li, J., Bai, Y., Zaman, N., & Victor, C. M. L. (2017). A decentralized trustworthy context and QoS-aware service discovery framework for the internet of things. *IEEE Access*, 5, 19154–19166.
31. Wu, L., He, Y., Wu, D. & Cui, J. (2008). A novel interoperable model of distributed UDDI. In *IEEE international conference on networking, architecture, and storage*. pp. 153-154. <https://doi.org/10.1109/NAS.2008.16>
32. Victor Paul, P., Saravanan, N., Jayakumar, S. K. V., Dhavachelvan, P., & Baskaran, R. (2012). QoS enhancements for global replication management in peer to peer networks. *Future Generation Computer Systems*, 28(3), 573–582.
33. Yuan, Bo., Liu, Lu., & Antonopoulos, N. (2018). Efficient service discovery in decentralized online social networks. *Future Generation Computer Systems*, 86, 775–791.
34. Victor Paul, P., Ramalingam, A., Baskaran, R., Dhavachelvan, P., Vivekanandan, K., & Subramanian, R. (2014). A new population seeding technique for permutation-coded genetic algorithm: Service transfer approach. *Journal of Computational Science*, 5(2), 277–297.
35. Deivamani, M., Murugaiyan, S. R., Ravisankar, V., Victor Paul, P., Baskaran, R., & Dhavachelvan, P. (2015). Web service composition framework using petrinet and web service data cache in MANET. *International Journal of Computers Communications & Control*, 10(2), 174–187.

36. Victor Paul, P., Moganarangan, N., Sampath Kumar, S., Raju, R., Vengattaraman, T., & Dhavachelvan, P. (2015). Performance analyses over population seeding techniques of the permutation-coded genetic algorithm: An empirical study based on traveling salesman problems. *Applied Soft Computing*, 32, 383–402.
37. Pazhaniraja, N., Priyadarshini, V., Divya, P., Preethi, D., & Victor Paul, P. (2015). Bio inspired algorithm based web service optimization-a survey. *International Journal of Applied Engineering Research (IJAER)*, 10(5), 13231–13242.
38. Caviglione, L., & Roberto, P. (2022). Evolution of peer-to-peer and cloud architectures to support next-generation services. *Future internet services and service architectures* (pp. 227–246). River Publishers.
39. Wang, R., & Lu, J. (2022). QoS-aware service discovery and selection management for cloud-edge computing using a hybrid meta-heuristic algorithm in IoT. *Wireless Personal Communications*, 126, 2269–2282. <https://doi.org/10.1007/s11277-021-09052-4>
40. Heidari, A., & Jafari, N. N. (2022). Service discovery mechanisms in cloud computing: A comprehensive and systematic literature review. *Kybernetes*, 51(3), 952–981.
41. Balaji, B. S., et al. (2021). Automated query classification based web service similarity technique using machine learning. *Journal of Ambient Intelligence and Humanized Computing*, 12, 6169–6180.
42. Ben Seghir, N, et al. (2019). A decentralized framework for semantic web services discovery using mobile agent. *Web services: Concepts, methodologies, tools, and applications*, edited by Information Resources Management Association, IGI Global, pp. 530–553.
43. Seghier, N. B. & Kazar, O. (2021). A context-aware service discovery framework based on mobile agent. In 2021 *International conference on recent advances in mathematics and informatics (ICRAMI)*, pp. 1–6. IEEE.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.



Dr. P. Victor Paul is working as an Assistant Professor in the Department of Computer Science and Engineering at Indian Institute of Information Technology Kottayam, Kerala, India, an Institute of National Importance of India. He has completed his Ph.D. in the Department of Computer Science (2015) from Pondicherry Central University, Pondicherry, India. He has done his B.Tech. in Information Technology (2007) and M.Tech. in Network and Internet Engineering (2011) from Pondicherry Central University, Pondicherry, India. He is a recipient of eminent INSPIRE fellowship from Department of Science and Technology, New Delhi. He is a University Gold medallist and University Rank holder in M.Tech and B.Tech studies respectively. He has also received “TCS Best Student Award 2010” from TCS for academic and research excellence at Pondicherry Central University, Puducherry. He has 9+ years of academic & industrial experience and published over 100 research articles in various International Journals & Conferences. Currently, he is working in the fields of Data Analytics, Optimization algorithms and Cloud Computing.



Dr. L. Jayakumar received his B.Tech degree in Information Technology from Anna University, Chennai, India and the M.Tech degree and Ph.D in Computer Science and engineering from Pondicherry Central University, Puducherry, India, in 2012 and 2019 respectively. He is currently working as Associate Professor at Department of Computing Technologies, SRM Institute of Science and Technology, Kattankulathur, India. His current research interests include resource management algorithm designs for wireless communication systems, Cognitive Radio Networks, IoT.




Dr. Achyut Shankarr is currently working as an Assistant Professor in Amity University, Noida, India. He obtained his PhD in Computer Science and Engineering majoring in wireless sensor network from VIT University, Vellore, India. He has published more than 70 research papers in reputed international conferences & journals in which 45 papers are in SCIE journals. He is a member of ACM and has received research award for excellence in research for the year 2016 and 2017. He had organized many special sessions with Scopus Indexed International Conferences worldwide, proceedings of which were published by Springer, IEEE, Elsevier etc. He is currently serving as an Associate Editor in Journal of Intelligent and Fuzzy Systems(SCIE, IOS Press) & SN applied sciences(SCOPUS & ESCI, Springer) and in year 2021 handing 2 special issues as a Guest editor in International Journal of System Assurance Engineering and Management(Springer) and Journal of Interconnection networks(World Scientific journals). He is serving as reviewer of IEEE Transactions on Intelligent Transportation Systems, IEEE Sensors Journal, IEEE Internet of Things Journal,

ACM Transactions on Asian and Low-Resource Language Information Processing and other prestigious conferences. His areas of interest include Wireless sensor network, Machine Learning, Internet of Thing, Block-chain and Cloud computing.



Dr. Shailesh Khapre is working as an Assistant Professor in the Department of Data Science & Artificial Intelligence at International Institute of Information Technology Naya Raipur, Chhattisgarh, India, an Institute of National Importance of India. He has completed his Ph.D. in the Department of Computer Science (2020) from Pondicherry Central University, Pondicherry, India. He has done his B. Tech. in Computer Science (2008) from National Institute of Technology, Rourkela, Orissa, India, and M.Tech. in Computer Science (2011) from Pondicherry Central University, Pondicherry, India. He is a recipient of eminent UGC-NET fellowship from University Grants Commission, New Delhi. He has 8+ years of academic & research experience and published various research articles in various International Journals & Conferences. Currently, he is working in the fields of Intelligent Information Retrieval, Service Personalization, IoT, Smart Autonomous Systems, Active Vision and Field Robotics.

Authors and Affiliations

P. Victor Paul¹ · Achyut Shankar² · L. Jayakumar³ · Shailesh Khapre⁴ 

✉ Shailesh Khapre
shaileshkhaprekl@gmail.com

P. Victor Paul
victorpaul@gmail.com

Achyut Shankar
ashankar1@amity.edu

L. Jayakumar
jkaylogu@gmail.com

¹ Department of Computer Science and Engineering, Indian Institute of Information Technology, Kottayam, Kerala, India

² WMG, University of Warwick, Coventry, United Kingdom

³ Department of Computer Science and Engineering, National Institute of Technology Agartala, Agartala, Tripura, India

⁴ Department of Data Science & Artificial Intelligence, International Institute of Information Technology, Naya Raipur, Chhattisgarh, India