2. BACKGROUND AND RELATED WORK

2.1 INTRODUCTION

The Web has evolved into islands of services that cannot interoperate easily with each other. Interoperable Web Services are the fundamental building blocks of the next evolution of the Internet. Collaboration of the services will be the value-added service model. As a result, web site interactions will become transparent to the end user. In the world of technology, where evolution is certain, program managers are constantly challenged to make wise investments in technology. The need for interoperability is raising many questions such as: Should the system be based on DCOM or CORBA? Should the system support multiple platforms? All answers agree that the systems must be web enabled. Within an organization, no one technology prevails. Information systems are implemented using different technologies and by different contractors. Yet, it is highly desirable to have these systems integrate and interoperate with one another.

Web Services enable interoperability of heterogeneous systems and allow the use of the best of breed. Web Services basically help mitigating the application integration crisis. It helps integrating applications at a significantly lower price point than any other integration technology. It is a kind of middleware based on XML and the Web. XML and the Web help solve the challenges associated with traditional application-to-application integration like heterogeneity. They are platform and language independent. Web Services also has the advantages of i) Exposing the API onto a network for connecting different applications, ii) Low Cost of communication, iii) Support for Loosely Coupled Applications, iv) Self Describing using WSDL, v) Automatic Discovery using UDDI and vi) Business Opportunity for the growth of businesses. UDDI is an online Web Service that can be used from the applications to dynamically discover other online services, all neatly packaged in a simple XML interface.
UDDI is the de facto standard for Web Services management on the web. It provides an XML/SOAP standards based framework for describing, discovering and managing the world of Web Services that will facilitate the growth of an integrated e-commerce environment. The UDDI standards and tools developed by the commercial world can provide a non-proprietary market place where agencies and contractors can describe their mission related roles and the types of services and data that they provide. The searchable central registries will provide a publish and subscribe mechanism to store the agency and service descriptions and to point to detailed technical specifications that define the interfaces to these services.

Metadata for the organizations, the services and the specifications are also stored in standard XML schemas so that the information can be discovered using standard search tools. The registries maintain technical fingerprints for specifications so that any change in the status of the service interface is immediately advertised on the web. While UDDI is not to solve the problem of controlling data formats or to force standard service interfaces, it provides a uniform system for organizations to define their role, a uniform way for these organizations to present their resources (services and data), and a uniform way for the rest of the community to discover, develop integration strategies, and manage their access to distributed services in the web environment.

No doubt, few UDDI registries surfaced on the Internet. Since the evolution of the UDDI registries mimics those of search engines, UDDI registries suffer similar disadvantages. Those disadvantages include broken links and invalid or irrelevant entries. To prevent these problems, there is a need of having a standard mechanism for removing outdated entries from the registry. Also, a mechanism for validating entries must be in place. In other words, the integrity of the data in the UDDI registry must be preserved. Another issue is whether UDDI registries of different vendors are synchronized with each other. This implies that a registration is propagated to all registries. The disadvantage is that an invalid registration must be removed from all registries. Alternatively, a web service can be registered in multiple UDDI registries.
UDDI acts a very important role in the web service. It can be used to publish and lookup services. Most of the current UDDI models are centralized and so that the performance decreases if there are too many services to be registered or queried. Furthermore, it is hard to maintain and manage the centralized UDDI servers (Libing Wu 2008). The main goal of UDDI is to implement the UBR (Universal Business Registry) and maintain the public service information. The goal of UBR is to store all kinds of service information which is not classified to different types. It will affect search efficiency. Bad interoperability of these private UDDI servers leads to more new isolated service islands. Current UDDI is a passive service directory, so how to actively detect service changes and mark the failure Web Services are still unknown. In view of the above problems, this thesis surveys an interoperable model of Distributed UDDI and its related problems to enhance the query efficiency, availability, and offer the way to improve information exchange among servers. This thesis also tries to address one of the core issues, Replication, which is nothing but Replica Placement and Management in the Distributed UDDI registries.

While Section 2.2 studies the existing problems of Centralized Universal Description, Discovery and Integration, Section 2.3 thoroughly analyses the Distributed UDDI and its problems. Section 2.4 is dedicated for a detailed discussion on the issues related to one of the major problems of Distributed UDDI, the Replication Strategies. This section thoroughly analyses the way how replication is currently done and what is the point of failure or the bottleneck in performing the replication management. This section also discusses the Replica Placement and Replica Management related issues.

2.2 CENTRALIZED UDDI

A main requirement of Web Service based Systems is the ability to find Web Services by querying UDDI registries. The Universal Description, Discovery, and
Integration (UDDI) protocol is a key member of the group of interrelated standards that comprise the Web Services stack. It defines a standard method for publishing and discovering the network-based software components.

UDDI is the central element for the Web Service based Systems in software design. The developers, architects and business policies are brought together through UDDI registry and thereby the software flexibility, reuse and control is increased (Oasis UDDI 2004).

### 2.2.1 Role of UDDI in Service Oriented Environment

UDDI is an important enabling element of the service-oriented approach to software design. The standard specifies protocols for accessing a registry for Web Services, methods for controlling access to the registry, and a mechanism for distributing or delegating records to other registries. In short, a UDDI registry provides a standards based approach to locate a software service, to invoke that service, and to manage metadata about that service.

UDDI is analogous to the Domain Name System (DNS), which controls the Internet’s network addresses, the CORBA, which implemented its Trading and Naming services to help direct the flow of system calls etc., UDDI looks similar in terms of the location transparency and other services of these systems but it also provides a mechanism for managing an otherwise ad hoc, chaotic, and un-scalable series of interactions and thereby it differs from the said systems. While web service based implementations are most sort of by the system developers and architects, discovery, manageability and security of the developed Web Services are the need of enterprises and that is addressed by UDDI.

Web Services based software development benefits and the need of code reuse, ongoing maintenance, and documentation becomes more vital. Moreover,
managing huge number of Web Services created by IT organizations and the need for making all of them available increases exponentially. UDDI helps the development teams in answering questions like how can development managers systematically organize and manage Web Services across multiple systems and development teams?, how can developers systematically manage the process of moving services through each phase of development: from coding to testing to public deployment?, how can programmers document interface specifications, message transports and authentication mechanisms with other developer groups?, as the services change over time, how can external applications accommodate the changes?, etc.,. UDDI enables the developer to reuse codes during development and deployment of Web Services and thereby the developer productivity improves. UDDI also supports run time binding, control over publication and distribution of Web Services information, client authentication, publish/subscribe for peer registries etc.,.

2.2.2 Challenges of Centralized UDDI

Existing UDDI technology uses central server (operators) to store pointers to registered Web Services which may be located elsewhere. However, using centralized approach has many drawbacks (Sittichai Laoverakul 2002) as it appeared in many research studies. The major risk in centralized UDDI is being the single point of failure.

As mentioned earlier about the role of UDDI, any web service registers to UDDI can be searched and then remotely invoked. Users can query for some specific services whose lists are registered to the UDDI system. The standard UDDI was created with incomplete concepts. There are a lot of expectations for a better or even a new management system. The flaws of the original design can be listed as follows:
1. First, if UDDI server fails, all the registered services will be gone. Currently 100% uptime is hard to achieve. When 100% uptime is not guaranteed, centralized UDDI servers are subject to non availabilities due to failure of different reasons. Whatever may be the reason, non-availability of UDDI server will make all the registered services unavailable to the service consumers too. UDDI server does not monitor the availability of the computing resources which contains the underlying Web Services. Consequently, it does not guarantee if the registered Web Services would be readily executable.

2. With centralized UDDI, controlling the flow of running multiple services continuously is tedious. When the UDDI registry is available in close proximity, control of such flow would be feasible but if the UDDI registry is available in a server which is very far off from the users’ location, flow control is highly difficult as it would consume lot of network bandwidth. This is due to the number of message passes that happens between the service consumers and the service registry.

3. In addition, for a normal user, registering to UDDI is complicated and users need to do it manually every time a Web service is created. The user has to register the created web service in the centralized UDDI by reaching the server through the network, possibly the internet. There is no guarantee that the UDDI registry is available in the near proximity. The user may be able to register the web service in the UDDI registry which is located in the same geography or in another continent. The users have to be highly dependent on the networking infrastructure to publish as well as to retrieve the Web Services.

4. Also, UDDI server does not monitor the availability of the computing resources which contains the underlying Web Services. Consequently, it does not guarantee if the registered Web Services would be readily executable.
5. It is noted (Penserini, Liu (2003), Zakaria Maamar (2005)) that a central authority constitutes a bottleneck and may completely break down the system. Having a centralized UDDI registry makes all its users to approach that single server for all the Web Services publishing and retrieving activities. This increase the network traffic in all the networking infrastructure, leading to that node.

6. Centralized UDDI registry also suffers from the disadvantage of interoperability among registries of different vendors which would create isolated service islands (Libing Wu, Yanxiang He 2008).

7. Centralized UDDI cannot exploit the advantages of different operating environments. Because, once the registry is established in a particular environment, whatever be the behavior of the registry in that environment, it has to be operating on that.

8. The goal of Universal Business Registry is to store all kinds of service information which is not classified to different types. It will affect search efficiency if it is being maintained by a single server.

9. The limitation of the UDDI in service discovery restricted to only the functional requirements is a major shortcoming of the UDDI. The extended UDDI, proposed by Wenli Dong (2007), takes into account the QoS in Web Service during the Service discovery process. Author states that the number of Web Services qualify for the given set of functional requirements could be many with different QoS attributes and that makes it important that the inclusion of QoS attributes in the Service discovery process becomes a necessity.

In view of the challenges faced in the centralized UDDI, as discussed above, the current UDDI attempts to alleviate the disadvantages of the centralized approach by replicating the entire information and put them on different sites
The Web Services information in the form of WSDL is registered not only in a single server in a single location but the same information is registered across many servers and those servers are distributed geographically (Libing Wu, Yanxiang He 2008).

2.3 DISTRIBUTED UDDI (DUDDI)

The present centralized UDDI structure is less robust and faces difficulty in supporting a large number of Web Services. UDDI registries collect service information in a passive manner, which means it waits for service publication, updating or discovery request passively and thus cannot guarantee the real-time validity of the services information. There should be a real time response to service enquiries with the service data that reflects the most updated information of services by the UDDI registries. But they do not have an automatic mechanism for updating the registry as and when services (and Service Providers) change. The current UDDI attempts to reduce the disadvantages of the centralized approach by replicating the entire information and putting them on different registries.

2.3.1 Interoperable Distributed Registries

Most of the current UDDI models are centralized so that the performance will decrease if there are too many services to be registered or queried (Libing Wu, Yanxiang He 2008). The authors proposed an interoperable model of Distributed UDDI which divides whole UDDI servers into three types: Root Server, Super Domain Server and Normal Server.

Root server mainly records the information of super domain servers and provides the function of registration and inquiry to super domain servers for the new UDDI registry. The super domain UDDI server is responsible for the
management of the nearby normal servers and provides registration and inquiry services. The normal UDDI servers can let users to publish and search the web service information. Using the simulation results of this proposed distributed UDDI architecture, it has been shown that, the response efficiency of inquiry requests descends in the Centralized UDDI architecture and the average response time has been proved to be less in the Distributed UDDI compared to that in Centralized UDDI. The authors observe that the response efficiency descends in Centralized architecture because it can only receive the web service update from service providers passively.

The major current service registry specifications, UDDI has the following drawbacks. First, it replicates all public service publications in all UBR (Universal Business Registry) nodes, which is not scalable and efficient, and second, it collects service information in a passive manner, which means it waits for service publication, updating or discovery request passively and thus cannot guarantee the real-time validity of the services information (Zongxia et al 2006). Authors proposed an active and distributed registry, Ad-UDDI, to provide available service information and to resolve the low validity of the public UDDI. In proposed design, the service information is distributed among multiple registries and thus the single point of failure and bottleneck in one public UDDI is reduced.

The proposed Ad-UDDI adopts a three-layered structure of distributed service registry namely, Root Registry Layer, Business Service Registry Layer and Service Layer. The Root Registry takes charge of managing the Ad-UDDI services without any business services, so the burden of root registry is lightened. The distributed architecture of Ad-UDDI may serve as a basic method of connecting the private or semi-private UDDIs. With the active monitoring mechanism, performed by a dedicated Ad-UDDI server to check the real time status of services and collects the service information periodically, the real-time availability of the service information in the Ad-UDDI is significantly improved.
While the simulation results of the Ad-UDDI design has been used to prove the metrics such as available rate, success rate, average response time, and total traffic cost of the Distributed UDDI architecture, the active monitoring mechanism does not check the correctness of the new update service information from the provider which would lead to the provider updating the information which is not relevant to the corresponding service. Also, on any crash of root registry, only Ad-UDDIs of same category can communicate with each other because they would only have the connectivity. Thus root registry leads single point failure for multi-category service search operations.

The service discovery issue is compounded by a plethora of different types of service directories that could be implemented by using various technologies in distributed systems (Serena Pastore 2006). The authors deployed the UDDI registry in an existing Globus based grid. Indexing, Searching and Retrieval of Services from a private UDDI deployed in a grid environment is studied by the authors. Grid principles focus on large-scale sharing in distributed systems, while Web Services are the software components necessary to provide remote programme-to-programme interaction, and offer interoperability and reusability. Both paradigms address such core issues as large-scale data transportation and management, high-performance remote access, discovery authentication and authorization in the context of each application in which they are used. Authors observe that the merger of these two approaches into service-oriented architectures (SOAs) (Erl, 2005) offers benefits in scalability, reuse and platform independence.

Authors proposal allows the VOs to act as a collector of providers publishing astronomical services into a standard UDDI registry (in accordance with the Web Services architecture), while requestors (grid clients or applications) will be able to perform their searches both through this registry and through the main grid information system. It will then be possible to carry on the development of grid
applications based on Web Services technologies to explore the new specifications and at the same time allow UDDI data to be described as grid resources accessible from the grid front-end.

A Crawler Engine for Large-Scale Discovery of Web Services called WSCE (Web Service Crawler Engine) has been proposed by Eyhab Al-Masri et al (2007). Authors state that the efforts that attempted to improve the discovery of Web Services have failed to address the issue across multiple UBRs. Clients who are searching for appropriate Web Services are experiencing performance issues in searching a multiple UBR due to the fact that UBRs are hosted on Web servers and they are dependent on network traffic and its performance. The proposed WCSE is able to crawl the multiple UBRs and collect the Web Service information from them. The proposed architecture is able to autonomous control to the businesses and organizations over their UBRs while the clients are allowed to perform search queries adapted to large-scale discovery of Web Services.

2.3.2 P2P Network Based Architecture for Distributed UDDI

In the traditional web service discovery method, service providers and requestors publish or query Web Services through a centralized registry, the UDDI. This approach suffers from single point of failure and many performance bottlenecks, such as too many data in UDDI, difficult to maintenance, too frequent to register and lookup and so on.

Whereas, a Peer-to-peer (P2P) computing is a distributed application architecture that partitions the tasks or workloads among peers. Peers are equally privileged nodes which can form a peer-to-peer network of computing nodes. Peers are equipotent such that they make a portion of their resources, such as processing power, disk storage or network bandwidth, directly available to other nodes of the
network, without the need for central coordination by servers (Rudiger). In P2P network, peers are both providers and consumers of resources.

Therefore, many researchers devote to utilize peer-to-peer (P2P) technology to solve the deficiencies of the traditional web service discovery method (Ni Yulin 2010). In this paper authors tried to combine P2P technology and UDDI-based centralized technology to provide a flexible and reliable service discovery approach.

The paper provides a novel P2P-based Distributed UDDI Web Service Discovery (PDUS) approach, which decentralizes the centralized UDDI. All published Web Services are distributed in many Register Center Nodes (RCNs), which is equal to distributed UDDI. PDUS stores service index in categories, which enables to improve the performance and efficiency for web service publish and discovery and also stores an extra index for each service, which avoids single point of failure and performance bottlenecks suffering from centralized approach.

But the proposed PDUS approach makes the UDDI system intricate with its complex architecture and each RCN should store large amount of route table information which may reduce the efficiency. Also, only theoretical study of the proposed approach is presented and an experimental results based justification would have made the proposal concrete.

Service descriptions are stored in a central repository, such as UDDI Registry, from where the operations of Services Publish and Services Discovery are performed. With number of accesses increasing, the bottleneck problem is becoming more and more severe. P2P is a non-centralized and dynamic and is a fast growing technology on distributed computing. Considering introducing it into the Web Service architecture, acts as the network basis of service discovery. Thus,
all information about Service description is published to the discrete Registry which is composed of P2P network (Zhenqi Wang 2007).

In this paper, Wang et al investigated a mechanism for Web Service publication, discovery and invocation with the method of P2P network using super/group peer with no “central server” to avoid the single node failure and improve the stability of the system. The proposed P2P infrastructure also makes the whole Web Service system more scalable, efficient than the traditional system by distributing the entries among all the peers, not focusing on only one or a few servers. In the proposed architecture, author publishes the service description information onto the P2P network built on JXTA, which is a set of protocols for creating P2P applications. A Web Service Broker (WSB) technique is also introduced which solves the incompatibleness between two isolated paradigms JXTA and Web service.

In the proposed architecture, Service Publishing and Service Discovery are done in the following way.

**Service publishing:** General service information in the P2P network is governed by the Super Peers and the service description is published via the publishing proxy to the super-peers. The proxy maintains a category of all the known domains. These domains must accord with the types of the services supporting by the super-peer. Therefore, the proxy can be seen as an interface between the service providers and the P2P network. When a publication message arrives, it has to choose an appropriate domain to which this business maps. Then the corresponding super-peers are assigned to deposit this service description information. One service is also allowed to map to multiple domains. If a new service wants to map to a domain that does not exist in the category, it is allowed to create a new domain to map.
**Service discovery:** Service requesters find services in the P2P network dynamically via a discovery proxy. The functionalities of the discovery proxy are to generate discovery request according to the service query submitted by the user, to transfer the query request to the super-peers on the P2P network and to summarize and dispose the query results that fed back from each super-peer, and then submit to the user. The authors presented a service discovery algorithm which permits service caching too.

It is observed that the response time would be huge since WSB spends much time in converting message format from JXTA to Web Service and vice versa. Also, failure of a super peer, which are responsible for query routing and are used to organize and manage the peer groups, would make the whole system fail. Moreover, only theoretical study of the proposed approach is presented whereas an experimentation analysis would have supported the proposal for its validity and its improved performance over the existing UDDI techniques.

Weifeng et al (2007) developed a pService system, which is based on a P2P overlay network for Web Services distribution, discovery and invocation, which merges advantages of Chord overlay (supports exact key search, load balancing and better search efficiency) and Skip Graph (J.Aspnes 2003) (supports range query, prefix search and semantic search) supporting complex tree lookups, locality sensitivity and ontology based service discovery.

The pService system for Web Services discovery is based on Chord and Skip Graph P2P overlay network. The Service providers hash the most important service description into Chord overlay network and use Skip Graph to distribute other service descriptions. In the proposed system both service providers and service requesters behave as peer nodes and each one of them executes the pService system to publish service descriptions or discover service descriptions.
My observation leads to a conclusion that the authors did not discuss about an efficient method to distribute the hash value of most important service description and other service descriptions into Chord and Skip Graph overlay networks respectively. Since only most vital service description like service name (as used in the paper) is hashed into the Chord overlay network and that could return large number of resultant services which may distant related to search.

While P2P architecture has been widely adopted to bring Distributed UDDI into existence, Khaled Ragab (2008) suggests a methodology for wait time management during web service discovery in the P2P based Distribute UDDI environment. The authors proved through simulation results that a reasonable storage and bandwidth utilization can be achieved in the Web Services discovery process from the services description document. A user, who is in need of a web service, submits an XML query and wait till the nearest node delivers the required web service. In the proposed approach, a timer has been introduced through which the user’s wait time is managed before getting the queried web service from the UDDI registries.

Javed I Khan (2008) investigated a broad range of heuristics schemas which are designed based on the Cumulative Delay (CD) and a child timer. Based on this study the wait time is arrived at and it really helps because too short waiting period may cause less accurate query result whereas a too long wait creates unnecessary delay in forwarding the result of the query to the user.

Evren Ayorak (2007) experimented combining a super peer network protocol with the efficiency of a centralized protocols and P2P networks. Content Addressable Network structure has been represented by the authors in order to avoid flooding the network with Web Service search requests and to minimize the number of messages routed in the network. The architecture proposed by the authors offers self-maintaining and self-clustering network where the peer groups
classify the web service definitions and each peer group becomes the owner of the classification dynamically.

Authors proposed a Super Peer Web Service Discovery Architecture (SPWSDA) which is a hybrid peer-to-peer solution and consist of three types of JXTA peer; registry peer, index peer and client peer. Each peer has specific roles in the architecture and it has different group of services and pipes listening to inputs. Index-peer indexes the registry-peer information and registry-peer is the repository storage peer to hold the information. Client peer is a GUI application generated to insert the Web Services to the SPWSDA network and search a Web Service in the network. In this work the concept of having the index peers kept known the up-to-date information about the Web Services and they are connected to each other in the SPWSDA network with the Content Addressable Network Algorithm violates the preliminary idea of P2P network, since each different peers have different responsibilities and functions.

Jan Sacha et al (2009) presented an approach to fully decentralize a service-oriented architecture using a self-organizing peer-to-peer network maintained by service providers and consumers. While the service provision and consumption are inherently decentralized, as they usually involve direct interactions between service providers and consumers, the P2P infrastructure enables the distribution of a service registry, and potentially other SOA facilities, across a number of sites available in the system.

Authors presented a gradient topology approach, which pushes the state of the art of super-peer election algorithms by using aggregation techniques to estimate system-wide peer properties. The gradient topology has two fundamental properties. Firstly, all peers in the system with utility above a given value are connected and form a gradient sub-topology. Peer utility, denoted for peer p, is a function of local peer properties, such as the processing performance, storage
space, bandwidth, and availability. Most of these parameters can be measured, or obtained from the operating system, by each peer in a straight-forward way.

Secondly, the structure of the topology enables an efficient search algorithm, called gradient search, which routes messages from low utility peers towards high utility peers and allows peers to discover services or data in the system. This approach allows the margin around the super-peer threshold to be configurable, which reduces the impact of random utility fluctuations on super-peer stability. This decreases the system overhead associated with creating or migrating super-peers. Peers successfully elect and update a set of highest utility super-peers, maintaining a total super-peer capacity proportional to the system load. The election algorithm can also reduce the frequency of switches between super-peers and ordinary peers, in case of fluctuating peer utility, by applying upper and lower thresholds and relaxing the super-peer utility requirements.

S. Sioutas et al (2009) proposed a new structured P2P overlay network infrastructure designed for Web Services Discovery, called NIPPERS (Network of InterPolated PeERS) which provides support for processing exact-match queries of the form “given a key, locate the node containing the key” and range queries of the form “given a key range, locate the node/nodes containing the keys that belong to this range”.

Author claims that the proposed new solution outperforms the most popular infrastructures used directly or as a basis for many solutions for P2P Web Service discovery including Chord (and some of its successors), BATON (and its successor) and Skip-Graphs.

Demian Antony D’Mello (2009) described that the Web Service architecture involving UDDI does not provide an efficient and effective discovery mechanism. To improve the performance of UDDI based Web service discovery, they proposed
the broker based architecture for Web service publishing and discovery. Authors explore the Service Operation Tree (SOT) to store Web Service information in a compact way which also speeds up the discovery process. Web Service discovery architectures have been broadly classified as Centralized Architecture and Distributed (de-centralized) Architecture. Authors further classified the distributed architectures as (a) Internet (Web) based architectures (b) Agent based architectures (c) P2P architectures. In this paper, Authors addressed the issues with a design of service knowledge (Tree structure) to represent Web Services of repository and a broker based architecture for effective and efficient Web service discovery along with an effective and efficient Web service discovery mechanism. The paper also proposes a broker based architecture for Web service publishing and discovery. The necessary information required for the matchmaking is pre-computed during Web Service publishing and stored at the broker for an efficient discovery of Web Services.

While the authors concentrated on only the insertion of new Web Service having N operations, service delete and update are not discussed in the paper. Also, the Web Service discovery has been explained using only the functional requirements and the non-functional requirements based Web Service discovery is not addressed. Authors also missed to explain about the specific type of registry (such as private, public and affiliated registries) in connection with the Broker Based Architecture which adds data management, security and key management issues to the system.

In a research paper authored by Fu-zhen SUN et al (2010), an alternative to UDDI has been proposed, which is more tolerant to single-node failures and utilizes an OWLS markup of services. Author indicates that a P2P based Web Service lookup is much better than a centralized directory service. Therefore a novel Semantic Web Service system has been presented by the authors, in which they resorted to P2P-based discovery framework to replace UDDI and distribute
Semantic Web Services among all the peers in P2P infrastructure based on the ontology contained in Semantic Web Services.

The P2P-based discovery framework also makes the decentralized system more scalable than traditional Web Service systems by distributing the system function among all the peers but not focusing on only one or a few servers. The proposed architecture consist of two layers which are, the first layer for the Web Service Translator, which translates WSDL files into OWL-S files and the second layer for the Web Service Distributor, which distributes and publishes the Web Services on the P2P infrastructure based on the ontology the Web Services contain.

Zhenqi Wang et al (2008) worked on P2P based Distributed UDDI environment and proposed an approach for semantic Web Service Discovery based on P2P network. In this paper, authors proposed a P2P network based (supporting semantic information) distributed Web Service discovery approach, using ontology to classify the registry peers into service domains. Due to the distributed P2P network, the performance bottlenecks caused by the traditional centralized Web Service discovery system have been resolved efficiently. While the two-phase service search algorithm based on the two-level topology has better search capability not only implements the accurate matching but also reduce the discovery time cost. The technique has been experimented from three aspects like time cost, precision and robustness.

Issues to Perfect and Optimize the whole Web Service discovery system and to implement automatic Web Service composition and execution are left open by the authors. There are scopes to further improve the accuracy of semantic based service matching algorithm and to balance the load on the registry peers by researchers.
2.4 REPLICA MANAGEMENT IN DUDDI

In a Web Services running-scenario, an UDDI registry participates in two operations. The first operation is to receive the announcements or publication of the description of the Web Services from the Service Providers. The second operation is to search the registry for the services that satisfy users’ needs. It is not necessary, in the context of a Distributed UDDI, that every Service Provider has to publish their Services information in a common registry. The Service Providers can register their Web Services information in any UDDI registry as per the policies defined in the Distributed UDDI setup. This kind of flexibility would always lead to a situation where the content of every registry in the entire UDDI registry need not be the same always. Every time a Service Provider registers a Web Service or Updates a registered Web Service with any one of the UDDI registries, the information need to be communicated to all those other registries where the copies of the Web Service need to be kept or updated. This will ensure effective Service Discovery.

Zakaria Maamar et al (2006) write in their paper that the dynamic management of multiple UDDI registries is similar to the problem of information replication in the domain of distributed applications. An immediate solution to this dynamic management is to flood the communication infrastructure with the new content of any UDDI registry, which has lately been subject to changes. Changes in UDDI registries may become frequent as the number of Web Services that are announced continues to grow. While the flooding fits well a wired context, the lack of reliable and permanent connections in a wireless context is a major obstacle to the flooding, author notes.

Authors proposed messengers which integrates the users and software agents to ensure the availability of several UDDI registries in a system where multiple UDDI registries are there. The authors state that in a system of several UDDI
registries, every registry is aware of the presence of the other peers but does not exchange its content, the Web Service announcements, with them. Authors proposed this concept in the context where there is no centralized component, who is going to manage and co-ordinate the UDDI registries.

As the Web Service announcements are submitted to different UDDI registries, the content of the registries would not be the same and thereby the authors found a need to propose mechanisms to support the exchange of content between UDDI registries.

According to the authors, each UDDI registry is associated with a structure known as cluster of Web Services. Several clusters have been made available across the communication infrastructure, so that providers can connect to the most appropriate cluster using various criteria (e.g., proximity, workload). For tracking purposes a provider cannot be connected to more than one cluster, which means a provider cannot announce in the UDDI registries of multiple clusters.

As per my observation, the messenger proposed by the author is a user agent (software) which is resident in user mobile node which needs more attention on its storage capacity and computing resources. Also, an UDDI-agent can become a point-of-failure or a bottleneck if a large number of users are under the management of the mobile support station that identifies the UDDI-agent.

Chaogang Tang et al (2009) developed reliable Web Services using independent replicas. Most of the fault tolerance strategies for Web Services are based on replication, which allow several Web Services to perform the same task. While these strategies can improve the success-rate of implementing the tasks to certain extent, they fail to consider the possible relationships among these services, especially when these services are themselves composite ones. In this paper, authors have proposed an approach to determine the independence of replicas, in
which the Trusted Third Party (TTP) is introduced as an intermediate agent to obtain providers’ trust and conduct the detecting work. It is also proved that it is feasible and beneficial to avoid employing a second replica which shares some common component service(s) with the first one (primary replica) that was used to implement a task but failed. Independence among the services can be identified at TTP using the information which includes the service addresses, service names, interface description, parameters and so on.

This paper is based on the condition that there are two replicas for each task. But in practice there are chances that the number replicas for a given task could be more than two many a times. The proposed approach can be further extended for multiple replicas which should also consider the issues like efficient replication, replica selection, consistency maintenance, replica replacement etc.,.

Replicating Web Services may reduce the end-to-end Web Service response time and add fault tolerance to the system, says Kambiz Frounchi et al (2006). Authors proposed a Web Service replica selection framework wherein the selection procedure is based on dynamic factors such as communication delays between Web Service provider and the Web Service requestor and the workload of the Web Service provider. Two replica selection policies have been simulated by the authors to analyze the performance of the proposed system.

The authors introduced a broker component in the traditional architecture which performs the selection procedure. The selection procedure is transparent to the Web Service requestor and avoids the need for the Web Service requestor to directly interact with the UDDI registry. In selecting the web service replica the broker considers dynamic factors such as the communication delays between the gateways and the workload of the Web Service providers, thus achieving a high system performance. Though the proposed system looks simple and achievable, the broker takes the control which intern leads to single point of failure. Also, checking
communication delays between the gateways and finding the workload of the Web Service providers makes the system more complex and will be more complex when the number of services published is high.

Lukasz Juszczyk et al (2006) concentrated on making UDDI registries usable within ad-hoc networks by segmentation, replication of services for increased availability, and synchronization of stateful services. Proposed mechanism consists of two parts.

First, Web Service Discovery and Registry system sorts out static registry problem by combining different benefits from existing distributed fashion solutions and providing an automatic discovery functionality which keeps track of alternating network structures. Descriptions of Web Services are published in distributed UDDI registries and updated automatically once a service ceases to be available or changes its location.

Second, Web Service Replication and Synchronization system is self-configuring, uses flexible algorithms which adapt their behavior to the dynamic characteristics of the network, and is able to perform the replication regarding performance properties of hosts and requirements of services. Furthermore, it provides an interface which enables to apply replication strategies suited for the individual requirements. The whole replication system can be described as a cooperation of three separate parts, namely Replicator Web Service, Replica Placement Mechanism, and Service State Synchronization.

Service replication and synchronization is explained by the authors but a detailed methodology and analysis have not been provided. Also, it is pointed that for leader election in the entire setup, a global view of the network is expected to be available whereas maintenance of global information in an unstructured network is
remotely feasible. Authors have not proposed clear strategies for replication techniques and the consistency maintenance of those replicas left untouched.

SHI Jin-yu et al (2008) presented their idea of Web Service platform in P2P environment. P2P implements the sharing of resources and services using effective communication among peers. The authors designed and presented a prototype for Web Service platform in P2P environment using JXTA protocol as the implementation technology. The prototype can be applied to publish, inquire and invoke Web Services. The JXTA group services are built in order that the client peer can transparently visit all the UDDI registries and the same service is deployed on many client peers so that service requestor can invoke the service to execute parallel computing. The prototype helps sharing the registered services information with the other peers of the system but the authors have not devised a methodology for an effective and efficient Web Services information transfer from one peer to the other in the system.

Wubin Li et al (2008) worked on consistency preserving mechanism. XML is widely used as message format for service providers and consumers in Web Services environment. XML message packaging and parsing brings extra overhead to both ends. Authors state that Web Services response latency as well as throughput is becoming a bottleneck problem. In this paper, Wubin Li et al proposed a consistency-preserving mechanism for Web Services response caching, which reduces the volume of data transmitted without semantic interpretation of service requests or responses, and accelerates the services response. Hash-based similarity detection is used to find the corresponding cached XML messages. The proposed mechanism attempts to cache Web Services response messages at both ends (client side and registry side) which increase the overhead of service hosts along with the existing various other service operation problems.
The performance of Web Service replica selection has been analyzed by Partheeban Chandrasekaran et al (2007). Authors felt that providing replicas will improve the overall system performance and redundancy for hardware failures. In this paper, the authors analyzed the replica selection algorithm called minU in a 11 node network. Most recent utility value, weight given to communication delay and replica placement have been considered for analysis. From the results it is understood that the response time achieved using the minU replica selection policy greatly depends on the identified factors and also on the topology used. For instance, as the weight given to the most recent utility value is increased, the response time of the Web Services found to be decreased.

Zhu Zhengdong et al (2009) authored a paper to contribute P2P based semantic Web Service composition and delivered an architecture to improve the scalability, stability and reliability of the semantic Web Service composition system. The architecture combines the advantages of centralized and decentralized structures. In this architecture the functions of the UDDI are distributed to the local web nodes and groups web nodes and public web nodes.

In order to search service resources more effectively, improve the interoperability between different P2P systems and reliability and stability of the whole system, perfect the composition management of the P2P-based Semantic Web Services, the algorithms of services publication, discovery and composition based on Vector Space Model and Semantic Index have been implemented in the established prototype system by the authors.

Each service is ensured that it is registered in a specific node of the public web through a localized mechanism of semantic Web Services in a CAN based P2P networks. The registration services of the public web nodes are divided by the area and shared by all nodes and this helps to overcome the problem of failure of a single node. The information about the registered Web Services need to be shared
with other registries through effective and efficient replication strategies, which is not discussed in the paper. There have been multiple projects on the proposed concept and to name a few, the project METEOR-S WSDI is based on P2P network to replace UDDI and support semantic Web Services and another project JXTA from Sun Microsystems is also based on P2P framework.

2.5 SUMMARY

As discussed in the couple of sections of this chapter a centralized UDDI registry is suffering from the problems of Single Point of Failure, Performance bottleneck, Passive updations of services information, Scalability, Reliability etc.,. Many research outcomes have been discussed in Section 2.3 to advocate the need for a Distributed UDDI registry. Majority of the papers discussed have adopted P2P structured Distributed UDDI registry for a dynamic, effective and efficient Web Service discovery by eliminating the problems of centralized UDDI registry. Section 2.4 discussed the need for proper replication strategies because the Distributed UDDI registry is suffering from almost all issues of any distributed application as noted by some of the authors as discussed in the section. Based on these discussions, Methodologies for an Effective and Efficient replica placement and updation methodology is highly felt necessary. The methodology, if proposed, should be highly effective and efficient even at situations where the number of Web Services registered is huge and huge enough and the number of Web Service information transaction between the client and registry is more. This requirement becomes the motivation for this research and the clear objectives based on this requirement are discussed elaborately in Chapter 3.