
D.B.E.

**Digital Business Ecosystem**

Contract n° 507953

# WP 24: DBE Implementation

# D 24.3: DBE Peer-to-Peer Architecture Design

**(M 24.7: First Release of Distributed Identity Framework and Service)**

**(M 24.18: P2P Architecture Document)**

**(M 24.19: Design of Service-Oriented Routing Protocol)**



Project funded by the European Community under the "Information Society Technology" Programme.

# Contract Number: 507953

**Project Acronym**: DBE

**Title**: Digital Business Ecosystem

**Version**: 1.1

**Deliverable N°**: 24.3

**Due date**: 31st October 2005

**Delivery Date**: 14th December 2005

**Short Description**:
This document describes a design of the common DBE Peer-to-Peer architecture that enables a distributed implementation of the DBE infrastructural services. The design includes a specification of the P2P network topology, a neighbour selection algorithm for the topology generation, a bootstrap mechanism, a replication strategy that improves the reliability and performance of the DBE infrastructural services, and a service oriented routing protocol for the proposed P2P topology.

**Partners owning**: TCD

**Partners contributed**: TCD

**Made available to**: Public

| Versioning | | | |
|---|---|---|---|
| *Version* | *Date* | *Name, organisation* | *Description* |
| 0.1 | 25 Feb 2005 | Bartosz Biskupski, TCD<br>Jan Sacha, TCD<br>Jim Dowling, TCD | Discussion document presented to the partners. |
| 0.2 | 28 Apr 2005 | Bartosz Biskupski, TCD<br>Jan Sacha, TCD<br>Jim Dowling, TCD<br>Jean Marc Seigneur, TCD | Draft version presented to the partners. |
| 0.3 | 25 Jun 2005 | Jim Dowling, TCD<br>Bartosz Biskupski, TCD<br>Jean Marc Seigneur, TCD<br>Andy Edmonds, TCD | Identity management section added. DHT overlay analysis added. |
| 0.4 | 12 Jul 2005 | Jan Sacha, TCD<br>Jim Dowling, TCD<br>Andy Edmonds, TCD | Architecture use cases and implementation hints added. |
| 1.0 | 31 Oct 2005 | Jan Sacha, TCD<br>Bartosz Biskupski, TCD<br>Rene Meier, TCD<br>Raymond Cunningham, TCD | Final version submitted for review. |
| 1.1 | 14 Dec 2005 | John Kennedy, Intel<br>Jan Sacha, TCD<br>Rene Meier, TCD | Reviewed version by Miguel Vidal and David Singer. Distributed Storage System included. |

**Quality check**

**Internal Reviewer**: Miguel Vidal, SUN

**External Reviewer**: David Singer, IBM Almaden Research Center, 650 Harry Road, San Jose, CA 95120, USA. E-mail: singer@almaden.ibm.com

# Table of Contents

# Table of Figures

# Related Publications

Jan Sacha and Jim Dowling, *"A Gradient Topology for Master-Slave Replication in Peer-to-Peer Environments"*, In proceedings of the 3rd International Workshop on Databases, Information Systems and Peer-to-Peer Networks, Trondheim, Norway, July 2005.

Jim Dowling, Dominik Dahlem, and Jan Sacha, *"Matching Distributed Systems to their Environment using Dissipative Structures"*, To appear in Workshop on Stochasticity in Distributed Systems (StoDiS), 2005.

Dominik Dahlem, David McKitterick, Lotte Nickel, Jim Dowling, and Bartosz Biskupski, *"Binding- and Port-Agnostic Service Composition using a P2P SOA"*, To appear in ICSOC Workshop on Dynamic Web Processes (DWP), 2005.

Bartosz Biskupski, Jim Dowling, and Jan Sacha, *"Designing Self-Organising Systems for MANETs and Overlay P2P Networks"*, Submitted for publication to ACM Transactions on Autonomous and Adaptive Systems (TAAS), 2005.

Jan Sacha, Jim Dowling, Raymond Cunningham, and Rene Meier, *"Searching in a Gradient Peer-To-Peer Topology"*, Submitted for publication to the 5th International Workshop on Peer-to-Peer Systems (IPTPS), 2006.

# Executive Summary

This document describes a design of the DBE P2P architecture in fulfilment of Deliverable 24.3 “Final version of the P2P architecture for service search” in Workpackage 24 “DBE Implementation”. The reported work has been done by TCD in fulfilment of Task C57 “P2P Architecture and Service-Oriented Routing” and Task C18 “Distributed Identity Service”.

The objective of this deliverable is to provide a design of an overlay P2P network that enables a distributed implementation of the DBE infrastructural services. The design includes a specification of a P2P network topology, a neighbour selection algorithm for topology generation, bootstrap mechanism, and a replication strategy that improves the reliability and performance of the DBE infrastructural services. The document also proposes a design for a service-oriented routing protocol based on collaborative reinforcement learning (CRL) that allows clients to route query requests to replicated DBE infrastructural services in the overlay P2P network. The P2P architecture design and the routing protocol design are associated with milestones 24.18 “P2P Architecture Document” and 24.19 “Design of Service Oriented Routing Protocol” in Task C57 “P2P Architecture and Service-Oriented Routing”.

The document marks also Milestone 24.7 “First release of Distributed Identity Framework and Services” in Task C18 “Distributed Identity Service” by providing a requirement analysis and an initial design of the Distributed Identity Service. The Distributed Identity Service is a DBE infrastructural service that validates the identity of DBE service providers and users.

# 1 Introduction

This document describes a unified peer-to-peer (P2P) architecture for distributed infrastructural services in the DBE. The goal of the architecture is is to provide a P2P communication layer that will enable a fully decentralised implementation of the DBE infrastructural services. The decentralised implementation is intended to replace the initial 18-month prototype, where many infrastructural services were implemented in a centralised fashion, for example using centralised servers.

The P2P architecture has been primarily designed to provide support for the Knowledge Base (KB) and the Semantic Registry (SR), which allow DBE service registration, unregistration and discovery, the Lookup Service (LS), which enables DBE service consumption, the Identity Service (IS), which enables DBE user authentication, and the Distributed Storage System (DSS) for persistent data storage.

Each of the DBE infrastructural services has its specific requirements and its own characteristics. However, building a separate P2P network for each service is non-optimal, with much duplicated development and maintenance effort resulting in prohibitive costs over the short, medium and longer terms. The aim of the architecture is to provide a common P2P layer that can be shared by the DBE infrastructural services in order to reduce the overhead for P2P overlay network maintenance.

This document describes the design of the P2P architecture, however, a detailed description of the design of the individual services built on top of it is considered beyond the scope of this document. The P2P architecture can be seen as a logical layer *below* the DBE infrastructural services, and as such it is not part of the services. Hence, this document focuses on describing a P2P topology that the KB/SR can be built upon and proposes a scheme for KB/SR data distribution and synchronisation between peers rather than describing interfaces and data structures used in the Knowledge Base and Semantic Registry.

## 1.1 Organisation of the document

The remaining part of this document is organised as follows. In section 2, we describe the general properties of P2P environments and outline the challenges of building P2P systems. In section 3, we focus on specific challenges that arise when designing a P2P architecture for the DBE. In section 4, we analyse the existing DBE infrastructural services and discuss the requirements for a common P2P architecture. Section 5 contains an overview of the proposed P2P architecture design. Sections 6, 7, and 8, respectively, describe in detail the three main components of the P2P architecture: the Gradient P2P overlay, the DHT P2P overlay, and the Distributed Identity Service. Section 9 describes the relations between the components and explains how DBE infrastructural services can be built on top of the common P2P architecture. Finally, section 10 summarises the document.

# 2 Challenges of Peer-to-Peer Environments

In this section, we describe general properties of peer-to-peer environments and discuss potential difficulties that arise from these properties when building peer-to-peer systems.

### *Decentralisation*

The most fundamental property of any P2P system is decentralisation. Nodes can never rely on a single, centralised server or entity. This applies to all aspects of the system, and requires distributed storage of data, distributed searching, and decentralised coordination for any function of the system.

We do not, however, make the assumption that all nodes are functionally equal or have similar capabilities. The system might distinguish between nodes and assign a special role to some of them. In popular P2P systems, such as Kazaa, distinguished nodes, that have higher capacities than other nodes, are called *supernodes* or *superpeers*. They might be selected at configuration time by human administrators, or might be elected dynamically at runtime.

### *Open Environment*

The network topology is dynamic, changing constantly during the system's operation. Nodes continuously enter and exit the system. New nodes can join at any time and member nodes can leave at any time and concurrently. Communication is unreliable, connections can be dropped, messages lost or arbitrarily delayed. Any node may fail and stop communicating.

### *Lack of Security*

Internet communication is generally untrusted, however, some data in the DBE is confidential and requires protection from unauthorised access. Additional mechanisms such as secure communication (data encryption) and identity verification (authentication) are often used to address this.

### *Scalability*

The system must scale with the number of nodes, users, services, amount of data stored, number of generated queries and updates. Modern P2P systems proved to be extremely scalable, millions of users have been reported to simultaneously access file-sharing systems such as BitTorrent or Napster. Similar number of users and nodes should be planned for in the design of the P2P architecture of the DBE. The system must be able to function in such conditions offering sufficient performance.

### *Robustness*

In large-scale massive systems, deployed in an open environment such as the Internet, random and frequent errors are inevitable. However, the system must be constructed in such a way, that a single fault can never prevent it from functioning. For example user data cannot be lost because of single node failures, and the searching mechanism should be robust against broken links and attacks. The system must be able to tolerate a certain level of random node, and network failures, potentially due to attacks, transparently to the users.

### *Self-Managing Systems*

Many existing systems, as they grow in size, require maintenance of skilled operators in order to function correctly. The emerging area of *autonomic computing* [1] envisions that systems will soon become too massive and complex to be manageable, and be beyond an administrator's ability to install, configure, optimise and maintain. Consequently, it has been postulated to build *self-managing* systems.

In such systems, administrators should be required only to make critical, high-level decisions, while the system should autonomously *self-configure* according to some high-level directives. The system should be aware of its performance and continuously s*elf-optimise* during the operation time, for example, by discovering shorter routing paths, respond to changes in the node availability and adapt to user demand, for example, by dynamically replicating certain data in regions of the network. Since it is not feasible to solve failures manually by human operators in a reasonable time, the system should *self-repair*; it should detect and fix problems automatically.

Many self-organising decentralised systems use the application structure and in particular its network topology to solve system problems. In our case, the P2P network topology should adapt itself to exploit the heterogeneity among peers in terms of their uptime, restriction caused by NATs/firewalls and available resources such as bandwidth, latency, storage space, etc. A simple example of adapting a network topology are super-peer networks, where super-peers have higher capacity, hence are more centrally located in the overlay network. The overlay network can also reflect the trust relationships between participating parties and evolve as peers gain first-hand and second-hand experiences when dealing with other peers.

# 3 Building a Peer-to-Peer Architecture for DBE

In this section, we identify specific challenges that arise when building a peer-to-peer architecture for the DBE. We consider specific properties of the DBE deployment environment, such as the characteristics of DBE users, network connectivity, support for hosts behind NATs and firewalls, peers with different capabilities as well as malicious and greedy users. We outline potential approaches to address the challenges.

### *Fully Connected Network*

It has been suggested at the March 2005 Computing Meeting that the DBE P2P system should be a partially connected network, where DBE networks could evolve around "connected" SMEs, and over time aggregate to form a larger DBE. This suggestion has natural attractions, in that one can visualise the DBE growing and evolving in a manner similar to how the Internet evolved from connected local area and private networks. However, the idea that SMEs from potentially disadvantaged, geographically remote areas would not automatically plug in to an EU-wide network, or, may only access the system through a highly connected (and potentially abusive) SME is undesirable. The suggestion also disregards the massive benefits of a fully connected DBE. Here Metcalfe's law, that states the utility of the network is proportional to the square of the size of the network, is instructive. Since, there are no technical limitations that prevent us from designing a fully connected DBE, it is our intention to propose a fully connected DBE P2P architecture where SMEs can discover and interact with one another, regardless of location or prior business relationships.

### *In B2C, many Users look for Needles in a Haystack*

In contrast to many existing P2P systems, where popular content is easier to find helping it, in turn, to become more popular, the DBE provides a lookup service (the Semantic Registry) that ensures that all registered services can be located and, if running, used. Many successful B2C companies, such as Amazon, have shown how a lot of their business is derived from niche markets and hard-to-find products, the so-called long end of the (popularity) tail [2]. Based on the experiences of these online retailers, we believe that a P2P system that supports users searching for hard-to-find services would provide compelling value to SMEs. In any case, the DBE also provides a recommender service which can be used to find more "popular" services in the DBE.

### *Supporting Firewalls and NATs*

Network Address Translation (NAT) and Firewalls present a substantial technical problem to the DBE. DBE infrastructural services and SME services provided behind a NAT gateway or a firewall cannot be directly accessed by clients unless the firewall or NAT gateway has been reconfigured to allow access or some NAT/firewall traversal technique is used (see [3]). Manual reconfiguration of NAT gateways or firewalls represents a substantial challenge to typical SME users who may not even be aware they have a NAT gateway or firewall installed. If we assume that users are required to (manually) reconfigure NAT gateways or firewalls, *it may massively reduce the potential adoption of the project*.

In order to enable the DBE services to operate in the presence of NATs, *a super-peer technique* based on the *STUN* (Simple Traversal of UDP through NATs) protocol and a *Relay Peer* has been identified that enables peers to auto-detect NATs/firewalls and traverse service requests through them. The STUN protocol is described in the IETF RFC 3489 [4]. An implementation of STUN is available on sourceforge, written in C++, at [5].

A standard is available for allowing NAT-restricted peers to send and receive messages over the public Internet called Traversal Using Relay (TURN) Server [6]. It allows a client to obtain a transport

address (IP address and port number) from a Relay Server, with which it can receive data from any peer and through which it can send packets to the public Internet. However, as we do not require an IP address and port number for each client, and the opening of new ports for each client consumes unnecessary resources, this solution is only considered as an aid to designing our super-peer solution.

In the super-peer approach, each peer behind a firewall/NAT has to keep a network connection open with one or more super-peers that are not behind a NAT or a restrictive firewall. A variant of a STUN protocol [4] will be used to discover if a node is behind a NAT. Services on peers behind a NAT/Firewall can be accessed via the super-peer that acts as an proxy for the peer. The same problem must be addressed for peers that initiate queries. If they are behind a firewall/NAT, then the reply has to be routed back through the assigned super-peer. Since one super-peer can maintain only a limited number of open connections with peers behind firewalls/NATs, an algorithm for assigning super-peers to NAT-restricted peers has to be designed. Another algorithm for finding a super-peer that has an open connection to a NAT-restricted peer has to be designed. Finally, in order to guarantee robustness, every peer behind a firewall/NAT should have more than one super-peer assigned since a super-peer might also crash.

A variant on the above approach is to route messages directly to peers behind a firewall. This approach requires the use of a firewall/NAT "hole punching" technique, such as the technique described in [3]. It was estimated that this technique works roughly for 82% of NATs when using the UDP protocol and for 64% of NATs when using the TCP protocol for communication.

Since the DBE requires a reliable solution that works in all situations, we decided to use the approach using relay peers to forward messages for peers behind firewalls and NATs. This approach, however, requires that some SMEs provide resources such as bandwidth and storage space for those that are behind NATs or firewalls. Therefore, incentives should be included to reward peers becoming "relay" super-nodes, which is a challenging research opportunity.

### *Exploiting SME Heterogeneity*

Beside being limited by firewalls and Network Address Translators, SME hosts differ in resources they can provide to the P2P system. SMEs provide resources such as network bandwidth, latency, uptime, storage space and CPU speed. These resources influence their ability to provide DBE infrastructural services and this should be reflected in the overlay network topology, i.e. all other factors being equal, peers with more resources should have more neighbours, receive more queries and cache more replicas. Exploiting peers heterogeneity have already showed to improve peer-to-peer networks scalability and performance [7].

### *Preventing Malicious and Greedy SMEs/Users/Customers*

The DBE system should be able to deal with malicious SME peers that introduce false information to the system in order to save their local resources or maybe even perform an attack on the system. Incentive mechanisms, security and trust models can be used to prevent such SMEs undermining the system.

# 4 Analysis of DBE Infrastructural Services

In this section, we analyse the individual properties of DBE infrastructural services that will be built on top of the DBE P2P architecture. The purpose of this analysis is to drive the design a P2P architecture that matches characteristics and requirements of these services. The issues analysed include estimated query and update rates for the services, estimated size of data that must be maintained by each service, data consistency and integrity requirements, and peer dynamism (arrival and departure rates, i.e., churn rate).

## 4.1 Knowledge Base and Semantic Registry

The DBE Knowledge Base (KB) and Semantic Registry (SR) components have similar functions in terms of querying and updating. Clients should be able to perform advanced queries using the Knowledge Base and the Semantic Registry. Much research has been devoted to querying mechanisms in peer-to-peer networks. However, these mechanisms have mainly focused on using Distributed Hash Tables where queries are based on a unique identifier (an index) that maps to a peer (or set of peers). It is not trivial to design a peer-to-peer network that enables efficient complex queries (such as range queries), where results are gathered using multiple indexes from multiple peers (see the "Queries" section in section 6.14 for a detailed explanation of the problem).

It is assumed that the SR/KB services will be *queried frequently* due to the expected high frequency with which clients browse or search for service data. Queries are issued both by the service suppliers and service consumers. It is our assumption that *updates will be issued less frequently.* Updates to the SR/KB occur when a new service is specified or composed, or when an existing service is being modified. Since it is expected that queries will be much more frequently issued than updates, the emphasis of the P2P design has been placed on improving the query performance.

Data replication is one method to improve the query performance in P2P networks. The potential size of the whole SR/KB storage is, however, difficult to estimate. In the case of just Semantic Registry that stores Service Manifests with size of at least 50kb each and 100 000 services deployed in the DBE, the required storage is at least 5Gb. If the system is to be scalable, we must be able to support an even greater number of services. Moreover, the Knowledge Base is at least the same size as the SR. Hence, it is not feasible to replicate the KB and SR on all SME nodes since their storage capacity is expected to be much lower. Therefore, data should be partitioned and replicated on peers with high storage capacities.

The content of the SR/KB has to be replicated for the purpose of fault tolerance. Data stored by the SR/KB is considered to be *persistent data* and should always be available to clients in the system. As stated in a TUC's report [8] on the P2P DBE Knowledge Base, we assume that Regional Catalysts will contribute resources (i.e. nodes of the distributed system) to a DBE community. It is assumed that nodes hosted by Regional Catalysts will be much less dynamic because they are part of the DBE infrastructure. This fact can be exploited in the peer-to-peer architecture design. A goal of the P2P architecture is to replicate the contents of the SR/KB to peers in the DBE that are stable and have high performance (e.g. high bandwidth, low latency, fast processor, large storage capacities). Clients in the DBE can then use replicas of the SR/KB instances to discover services in the DBE.

In their report on a P2P architecture for the Knowledge Base [9], TUC have identified two particular types of peers that in the P2P system that will host the SR/KB contents:

- KB Nodes: These nodes are willing to have a KB instance installed (contributing to the P2P Knowledge Base). The local KB will store knowledge of that node and will keep replicas of knowledge hosted in various other KB nodes.
- Index Nodes: These nodes will host a KB instance and will also have indexing responsibilities.

However, we assume that all nodes may become KB nodes or Index nodes, and that the algorithm selects the nodes on which KB or index data will be replicated. The design of the KB and SR does not require strict consistency, i.e., the updates to service manifests and other data may not be propagated instantaneously to all replicas and hence the updates may become available to clients with some (reasonable) delay, typically hours rather than days.

The integrity of data in the SR/KB should be guaranteed by the P2P architecture. If the SR/KB data is to be distributed among untrusted peers, it has to be secured from malicious modifications. Therefore, it should be either digitally signed by owners or supplied only by trusted nodes. In particular, update operations to the SR/KB require authentication, which will be based on the use of decentralised Credential Server (CRES), and authorisation, which is considered beyond the scope of the DBE project.

Since any SME in the DBE can deploy a local SR/KB, there may be many SR/KBs deployed in the DBE at any time. This, however, requires a SR/KB Name Service to discover the list of SR/KBs in the DBE.

## 4.2 Semantic Registry / Knowledge Base Name Service

Every Semantic Registry and its associated Knowledge Bases (SR/KB) will be maintained by either a SME or a catalyst node, and there may be many different SR/KBs in the DBE. All SR/KBs will conform to a common meta-model. This leaves peers in the DBE with an issue of discovering SR/KBs within the DBE. This is relevant in allowing a SME to decide in which SR/KB it should publish a service as well as enabling service consumers to decide to which SR/KB a query for a service should be sent. We propose that on entering the DBE a peer should be able to discover and use a SR/KB.

We propose the creation of a DBE infrastructural service called a SR/KB Name Service that will be used as a distributed name service to discover the list of active Semantic Registry Servers in the DBE. Each peer that provides a SR/KB listed in the SR/KB Name Service will provide a replicated copy of the service. The service will provide registration and unregistration operations for allowing SR/KBs to be registered and unregistered. We expect only a limited number of SR/KBs, therefore we propose using a full-consistency model based on a simple flooding scheme to synchronise the contents of each copy of the replicated service. However, as nodes may periodically be unavailable the service needs to ensure that when a node reconnects, it synchronises its entries with the other nodes.

When a peer registers, it provides credentials that are evaluated based on the SR/KB Name Service policy. Depending on this policy, the provided credentials may or may not be sufficient to grant registration.

## 4.3 Service Lookup

Service Registry and Knowledge Base enable users to discover Service Manifests of services that are provided by SMEs in DBE. However, Service Manifests provide only identifiers of services and SR/KB do not support mechanisms for locating actual services on SME hosts. Therefore, the Service Lookup component was designed to assist users in locating hosts that provide requested services (based on their unique identifiers) as well as obtaining service proxies that enable method invocation on these services. Service Lookup has to be efficient as we assume that users will perform lookups frequently,

each time a service is to be consumed. However, there are *no persistency requirements* as a service proxy may only be available when the corresponding service is running. In particular, a service lookup request may fail when the corresponding service cannot be accessed (e.g., due to a failure). This is in contrast to Service Registry and Knowledge Base, where Service Manifests have to be constantly available to users and SMEs.

## 4.4 Distributed Storage System

The DBE Distributed Storage System (DSS) provides for medium and long term persistence of content on the DBE. From a functional point of view, the DSS will accept an array of bytes, persist it on the network, and return a hashcode to the invoker. On the subsequent receipt of this hashcode, the DSS will retrieve the relevant content. A key property of the DSS is that every data item stored in the system is uniquely identified by its hashcode.

From this brief description it can be seen that queries of the DSS will be functionally trivial: locate the peer or peers on which a particular hashcode is stored (or should be stored). As the hashcode is based on the content, the concept of updating a block of content on the DSS does not arise. Content is written, and after a "time-to-live", is automatically deleted. It is envisioned that each peer in the P2P architecture will contribute at least some storage to the DSS.

Unlike the KB or SR, the DSS does not need to support complex queries. Since all data items in the DSS are uniquely identified by hashcodes, the DSS supports exact-match queries only. This allows the data to be fully distributed between peers in the system. Complex queries, on the other hand, usually incur high overhead in distributed systems and can be handled more efficiently if the system's data is centralised.

Another requirement that applies to the DSS is redundancy. As absolute redundancy cannot be guaranteed, and redundancy implies extra overhead, it is envisioned that the user will be requested to specify the degree of redundancy required. The more redundancy requested, the more copies of the data that are saved. Several approaches for identifying the target peers for duplication have been identified. The content could be saved to peers adjacent to the `primary' target, the content could be saved on multiple overlay networks, or the content could potentially be distributed across an appropriate spread of peers of appropriate profiles.

For performance reasons, the DSS will also need to support caching. This can be achieved by allowing multiple entries to be added for each hashcode in the distributed lookup table of content locations.

## 4.5 Conclusions

In the course of our analysis we have discovered that some infrastructural services share similar characteristics, while other infrastructural services have different characteristics. In particular, the KB, SR, and the KB/SR Name Service share similar properties of high data persistency and support for complex queries. The Lookup Service, however, does not impose strong persistency requirements and does not require complex queries, since data in the LS is always uniquely identified. Similarly, in the DSS, and in the Identity Service, as discussed in section 8, the data is uniquely identified and the system supports exact-match queries only.

This lead us to a conclusion that the P2P architecture should be based on two different P2P overlays. A separate P2P overlay will be used for persistent data management and complex query support (SR, KB, Name Service), and a different P2P overlay will be used for efficient, exact-match identifier routing (LS, IS, DSS). The overall design of the architecture is presented in section 5.

# 5 DBE Peer-to-Peer Architecture Overview

In this section, we describe the P2P architecture, giving the reader an overall view before describing in details the individual components in the subsequent sections.

The DBE P2P architecture is composed of two overlay networks, where one of them is a structured P2P network, and the other one is an unstructured P2P network. For the reasons outlined in section 4, the KB, SR, and SR/KB Name Service share one overlay (topology), which is the unstructured topology, while the LS and IS are built on top of another, structured overlay. In addition, there is a central server, called DBE Portal, that is used for bootstrapping, authorisation and firewall detection. Each SME node participates in two topologies (overlays), maintains connections and states of neighbours in both topologies, and occasionally connects to the DBE portal (see Figure 1).

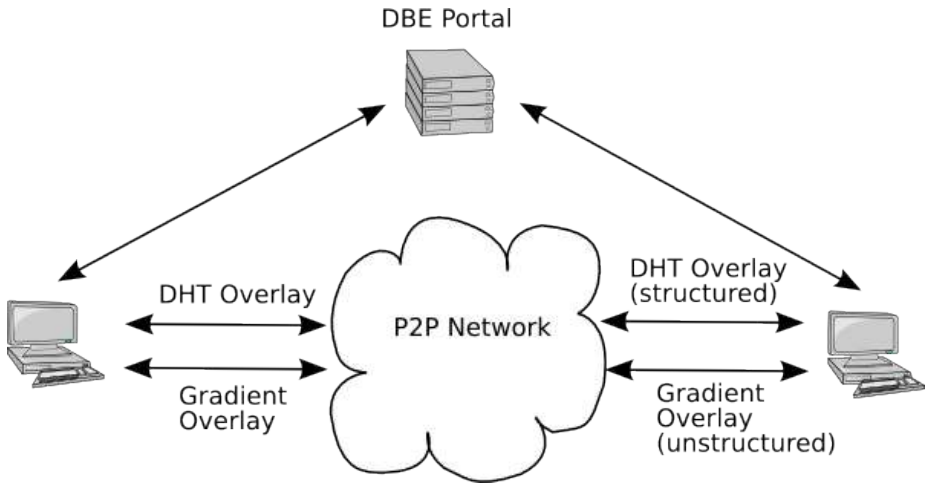


*Figure 1 DBE P2P Architecture*

The unstructured topology called gradient topology enables SMEs to register and unregister services as well as to locate their service manifests using a query tool or keyword-based search. The gradient topology improves the efficiency of SR/KBs by replicating their data to highly available and fast nodes (i.e., replica suitable nodes). This improves the DBE scalability as well as search performance by replicating data closer to clients. The main characteristic of this overlay network is its gradual design, where higher performing nodes maintain more replicas and receive more queries. The design of the gradient topology is discussed in details in section 6.

The structured DBE P2P overlay topology is based on Distributed Hash Tables (DHTs) that provide efficient mapping from an identifier space (keys) to hosts. Each peer is assigned a unique identifier (key) and responsibility for part of the key space. All operations on DHT are routed in a form of requests in a multi-hop fashion, i.e., from one peer to another, towards the final destination (the peer responsible for the requested key). State-of-the-art systems provide efficient algorithms that enable lookups in time $O(log(N))$, where $N$ is the maximum allowed number of peers in the system. The particular DHT overlay used in the DBE P2P design is used for locating SME hosts that provide services that a user is willing to use, based on their identifiers (SERVICE_IDs) obtained from Service Manifests with the use of the gradient topology. Details of the structured DBE overlay network are presented in section 7.

Figure 2 shows the relations between DBE infrastructural services and the P2P overlay networks. The Knowledge Base, Semantic Registry and the SR/KB Name Service share the gradient overlay. The Identity Service and the Lookup Service are built on top of the DHT network.

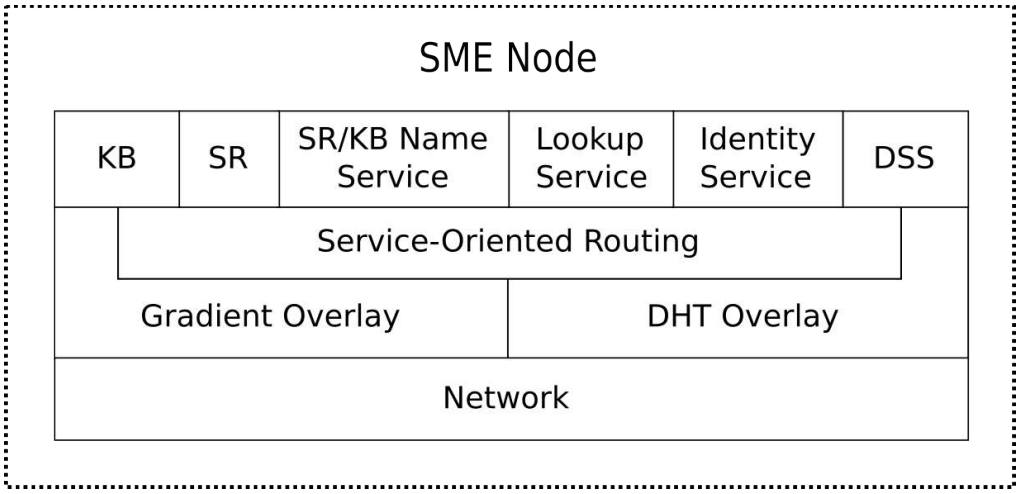


*Figure 2 DBE Infrastructural Services on top of DBE Peer-to-Peer Overlays*

Figure 3 shows examples of interactions between DBE nodes (SMEs) and DBE infrastructural services through the common P2P architecture. The first scenario shows a service discovery and consumption. A consumer contacts the well-known SR/KB Name Service (1) using the gradient P2P overlay, and obtains a list of active SR/KB instances. Subsequently, the consumer contacts a chosen SR/KB instance (2) using the gradient topology and discovers services of interest. Finally, the consumer binds to a selected service (3) using the DHT overlay and eventually, uses the service (4). The identity of the service provider and a list of credentials can be obtained from the provider's Identity Service available over the DHT overlay.

A second scenario in Figure 3 shows a service creation. A provider deploys a service on a local host in the provider's SME (A). When the service is deployed, the provider connects to the SR/KB Name Service using the gradient P2P overlay (B) and obtains a list of SR/KB nodes. The provider then selects a SR/KB instance, connects to it through the gradient topology (C) and registers its service.

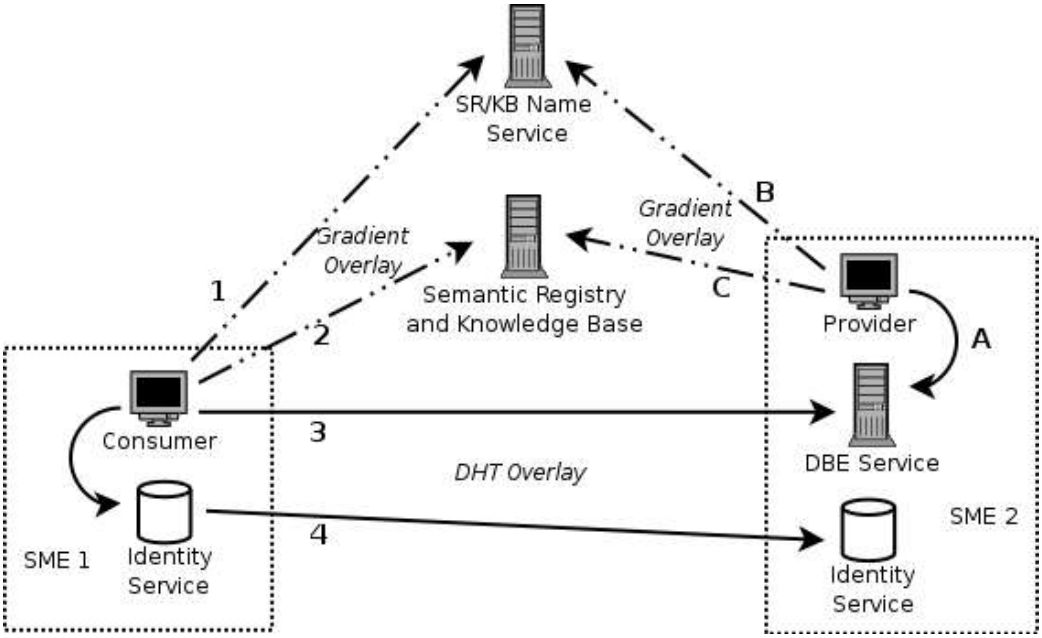


*Figure 3 Service-Oriented DBE Peer-to-Peer Architecture*

# 6 Gradient Overlay

The overlay network we propose for the Knowledge Base and Semantic Registry is based on an unstructured P2P network, where the system's topology, replica placement and searching mechanism will dynamically adapt to the system's environment. The key properties of this system is that data will be persistent, replicas will be managed and the system will be scalable. Other goals include supporting complex queries, general search mechanisms, and the ability to discover both popular and unpopular services efficiently.

When addressing the persistent data requirement, we must decide on where to store the data. There are two extremes, one is to store all data in a centralised server, which introduces scalability problems, and the other one is to partition the data among a set of peers using some indexing scheme, which is often the case in structured P2P networks. A potential problem when partitioning the data among peers is that the use of peers with lower bandwidth/performance/stability/trust to store data would degrade the performance of entire system.

Existing P2P systems, such as Distributed Hash Table (DHT) based approaches, usually assume that all peers are similar and have equal capabilities for maintaining data [10]. For example, Chord [11] assumes that the distribution of resources among the peers is uniform. However, this was shown not to be the case in real-life systems, where the distribution of peer characteristics, such as the number of connections, the uptime, available bandwidth or the storage space, usually exhibit the so called scale-free or heavy-tail property [12][13][14][15]. Systems that do not address the heterogeneity of the environment and do not adapt their structures to the environment often suffer poor performance, especially in the face of high churn rates, i.e., high frequency of peers entering and leaving the system [16][17][18].

In order to improve the performance, many existing systems only allow data to be stored on the fastest, highest bandwidth and most reliable, trusted peers, called *superpeers*. It is non-trivial though how to identify and select superpeers from peers in a system, due (mainly) to the lack of global knowledge. Potential solutions include hard-wiring them in the system, configuring them manually using an administrator, or automatically electing superpeers using runtime information. For example, in the DBE catalysts could be manually selected as superpeers. However, this conflicts with the assumptions of self-management, decentralisation, and the lack of a central authority that controls the structure of the system. An adaptive self-organising system is preferable, where peers automatically and dynamically elect superpeers, according to demand, available resources and other runtime constraints.

Finally, a flexible system should recognise a large diversity of peer capabilities and not just be limited to a two-state distinction between superpeers and ordinary peers. The role of a peer should be related to the peer's capabilities and, in general, more powerful peers should have more responsibilities, while slow or unreliable peers should have less duties. For example, the amount of data stored by a peer could be calculated from the available storage space, network bandwidth and reliability. Similarly, the amount of searching traffic forwarded by a peer should be proportional to its ability to forward packets.

## 6.1 Catalysts as Trusted, Reliable Peers

In the DBE environment, a special role is assigned to the catalysts, which are entities (enterprises) trusted by all SMEs in the system. We propose that catalysts maintain a fixed set of highly available peers known to all SMEs. These peers automatically obtain the status of superpeers, and are responsible for tasks such as storing primary replicas of data, maintaining public key and certificate repositories, and managing the Semantic Registry name service.

As the system grows in size, we envision that the number of peers provided by catalysts becomes too small to maintain all data in the system and handle all queries generated by the users. Therefore, we propose a replica placement algorithm that selects the most reliable, stable peers and assigns them the role of superpeers, i.e. to store replicas, handle queries, provide routing. Such an approach should maximise the system's efficiency, since the most powerful and reliable peers have the highest impact on the system's performance. Slow or unreliable peers, when assigned no particular function in the system, have very little influence on the overall performance.

The catalysts, as trusted entities, are responsible for running the most critical DBE infrastructural services: Semantic Registry Name Service and KB / SR Services.

## 6.2 Replica Placement Criteria

The selection of peers for replica placement is based on the following criteria:

- ***Stability*** – Every peer joining or leaving the system introduces extra overhead to P2P maintenance. The assignment of DBE infrastructural services to the most reliable peers reduces the risk of system reconfiguration when peers leave. Stable peers can increase the system's availability, provide reliable routing and maintain long-lived replicas of system data.
- ***Bandwidth & Latency*** – Peers connected to high-bandwidth, low-latency networks can maintain more connections with peers, handle more queries, forward more packets, and update replicas more frequently.
- ***Storage –*** High storage capacity allows peers to host more replicas of system data.
- ***Processing Performance –*** Faster CPU allows peers to handle more DBE infrastructural service requests.
- ***Trustworthiness*** – In some systems only trusted peers may be allowed to provide certain functionality. The system might track the level of trust of peers, e.g., their reputation (please refer to section 8 on computational trust), in order to select the most suitable candidates for some tasks.
- ***Open IP Addresses –*** Peers with open IP addresses can be used as relays that manage interactions between clients that are behind Network Address Translation (NAT) gateways or firewalls.
- ***Willingness to Share Resources –*** In order for a peer to become a superpeer, the SME must be willing to facilitate the installation of DBE services.

We believe that many of the above parameters are strongly correlated. Stable peers are more likely to offer high bandwidth and low latency, have better reputation and probably provide more connections, storage space and CPU power than non-stable peers.

## 6.3 Peer Utility

We define a peer's *utility* as a function of the above parameters. Utility, essentially, describes a peer's ability and willingness to act as replica of SR/KB data. Utility is a critical factor in the algorithm that generates the proposed P2P topology and is used in the replica placement strategy presented in sections of this document.

A key principle of the system is that *persistent data is stored by the highest utility peers.* This strategy addresses a problem faced by many existing peer-to-peer systems, where some data items, especially less popular items, are hardly accessible or even lost due to a peers' instability or lack of resources such as bandwidth. In our design, the system tries to maximise data availability, security and quality of service by placing data replicas on the highest utility, selected hosts.

In a closed system, where all peers trust each other, it's sufficient that every peer evaluates its own utility level. In this case, we assume that the trustworthiness parameter corresponds to its maximum for all peers. Neighbouring peers can exchange their utility information without any verification procedure due to an assumption of peer fairness. In an open, untrusted environment, such as in the DBE, a separate mechanism is needed to assure that utility claimed by peers corresponds to their actual status. In particular, the system must be protected against malicious peers providing false utility information, either about themselves or about other peers, peers changing or spoofing identity or creating multiple virtual peers. The system should also be robust against greedy peers. These challenges are further discussed in section 8.

## 6.4 Gradient Topology

We have designed a P2P topology, which we call a *Gradient Topology* [19], where the highest utility peers, maintaining persistent data, are highly connected with each other and form a logical *core* of the network, while the network around the core is composed of other peers with lower performance or reliability (see Figure 4).

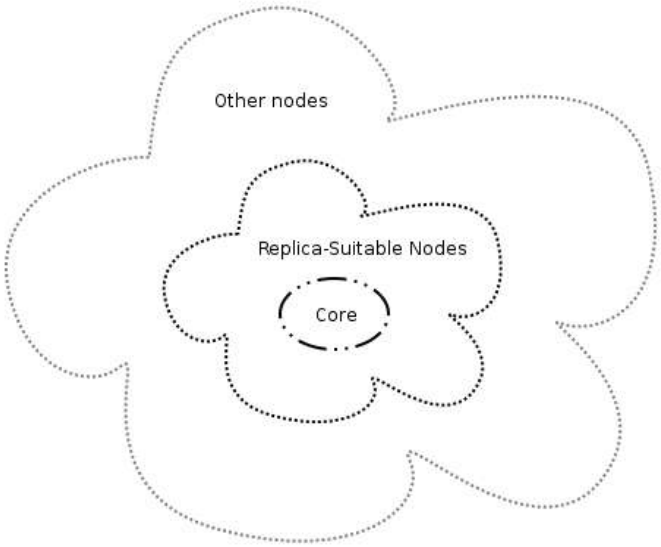


*Figure 4 Self-Organising system topology based on peer utility*

Assuming the scale-free, power-law distribution of resources [13][14][15], a great majority of peers belong to the category of ordinary peers (with lower performance or reliability). The number of core peers is relatively small, but the amount of available resources and stability of core peers is significantly higher than the outer peers. Core peers are more likely to act as servers, while the outer peers are more likely to act as clients. The main advantages of grouping high utility peers in a logical core are the following:

- Searching for high utility peers suitable for maintaining data replicas is less expensive in terms of number of messages generated, since it only requires communication with a small subset of peers in the network.
- Synchronisation cost for replicated data is lower, since peers storing replicas are located close to each other (in terms of number of hops) and are connected by stable, high quality routes.
- Complex queries that require cooperation of multiple peers can be handled more efficiently, since routes between peers storing data are stable and distances between peers are low.

This structure of the system can be seen as a set of concentric zones, where each zone consists of peers with similar levels of utility. The inner zone contains the highest utility peers. The subsequent zones are delimited by gradually lower status peers and the most outer zone contains the lowest utility peers.

As core peers will be well-connected, have high bandwidth and processing power, they will be able to maintain a relatively high number of connections. Consequently, this enables fast data dissemination and more frequent replica updates. On the other hand, peers far from the core are not suitable to maintain replicated data, due to poor utility, resource constraints or the owner's unwillingness to contribute resources to the system. It is not desirable to place such peers in the core of the network since they would decrease a system's performance.

## 6.5 Neighbour Selection

In order to generate the gradient P2P topology and to enable the emergence of a stable core, we have designed the following neighbour selection rule: *two peers may become neighbours if they have similar utility status*. Additionally, high utility peers will have more neighbours, since they have more resources available to maintain network connections.

However, early simulations (see section 6.16) show that a greedy selection policy, where peers always select neighbours with the most similar characteristics, leads to high network clustering and in consequence long distances between peers. In some cases the clusters get disconnected and the network becomes partitioned. Also the highest utility peers do not always connect to a single core, and sometimes there are multiple clusters of high utility peers separated from each other by a number of lower utility peers.

The algorithm has been improved by allowing peers to connect to other non-similar peers, with the probability exponentially decreasing with the difference in two peers utility. However, it turned out that a randomised strategy, where a percentage of neighbours was always selected at random, gave the best results. Random connections serve several purposes. First of all, they allow peers to discover potential neighbours with similar utility level, even in remote regions of the network, which in turn enables the formation of a single cluster containing the highest utility peers. Random connections thus play a similar role to the exploration in traditional multi-agent systems. Second, random links prevent the graph from being disconnected. As shown in [20], even a small number of random connections, for example 20 per peer, is sufficient to make the probability of network partitioning negligibly small. Finally, the randomised approach has the advantage that it is simple and it does not require parameter tuning.

A peer maintains two independent sets of neighbours, randomly-selected connections, and greedily-selected connections to peers with similar utility status (see Figure 5). New connections are discovered by gossiping, i.e., by randomly contacting already existing neighbours and exchanging with them the lists of connections. It is important to note, that the connections inserted to the random sets are always selected from other peers' random sets, which guarantees that the sets remain random. In section 6.16

we show that the proposed neighbour selection approach generates a gradient network topology consistent with our goals, where the highest utility peers are highly connected with each other and form a stable core of the network.

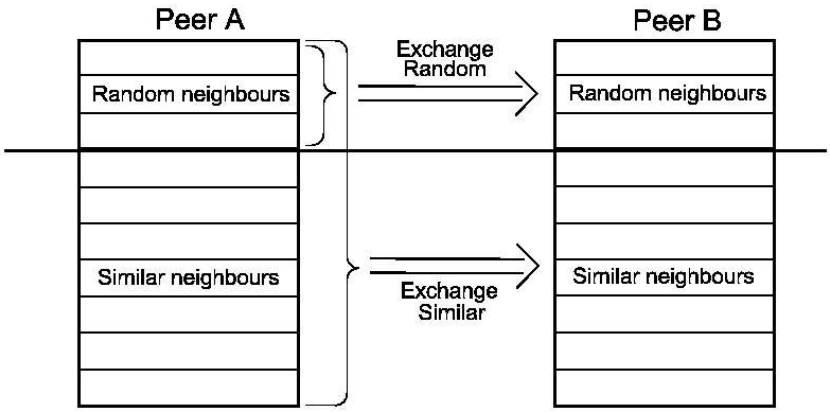


*Figure 5 Neighbourhood set exchange from Peer A to Peer B.*

## 6.6 Incentives

In an open system, it must be assumed that peer behaviour is selfish, and peers are generally interested in consuming resources rather than sharing. This means that a peer is interested in getting a position close to the core, in order to decrease its distance to replicas and to improve the performance of lookup and update operations, potentially without contributing resources to the P2P system. Therefore, the system must provide incentives to share resources by increasing the status and the proximity to the core for peers which provide resources to the system.

This is consistent with the neighbour selection algorithm. A high utility peer hosting replicas is not usually interested in having neighbours that consume its resources (e.g. generate queries) and offer no resources in return, and might close connections to such peers. However, two peers storing multiple replicas have a mutual interest in becoming neighbours, since they can both profit from an exchange. Peers may agree on becoming neighbours generally when their status is comparable. In particular, direct access to the core should be limited to the highest utility peers.

## 6.7 Bootstrap

The bootstrap problem, which is the task of finding an initial set of neighbours and configuring the initial settings when a peer joins the system, seems to be especially difficult in peer-to-peer networks, as there is very little or no support for broadcasting in the Internet infrastructure. Initially, peers require explicit, sometimes external, support to discover potential neighbours.

Solutions applied in existing peer-to-peer system rely on a centralised name registry, such as KaZaA [21] or Skype [22] or a fixed set of name registries, which provide addresses of initial neighbours to all peers entering the system. In Skype, the address of the name registry itself is widely known and hard-coded into all peers' software.

We propose the use of the DBE Portal and Regional Catalysts for bootstrapping SME nodes. The DBE Portal is a central DBE server, which can for example host DBE websites, authenticate and authorise

DBE users, and provide a SME firewall detection service. The location of the catalyst peers may be provided together with the DBE software or located on the DBE portal. Each catalyst peer maintains a limited registry with addresses of peers in the catalyst's region. The list of peers is provided upon a request. Each user should choose its own regional catalyst for bootstrapping in order to assure locality of network connections. The peers may cache addresses of neighbours between sessions in the system, so that when they restart they don't need to contact catalyst peers again.

## 6.8 Data Distribution

The amount of data stored by a peer depends on a peer's utility, available resources and current demand. It is acceptable for core peers that have sufficient storage and bandwidth capacity to store full replicas of the entire SR/KB database. However, as the amount of data grows in the system, we expect that no single peer in the network will be able to maintain a full copy of the SR/KB, and hence data will have to be partitioned among peers according to some distribution policy.

It has been suggested by TUC that the KB and the SR may grow by partitioning along different business domains. It is expected that different expert companies in different domains will develop different business models and install them locally on their own Semantic Registry and Knowledge Base. Other companies within those domains may reuse these business models and become associated with the Semantic Registry of the expert company.

Each data object (replica) contains all service manifests belonging to a particular SR, potentially together with other meta-data from the KB, such as BML models. At a low level, a SR may correspond to tables in a relational database. Such a solution has the advantage that a single replica maintains enough data to answer a simple query, such as, for example, getting all services from a certain SR that satisfy certain conditions. However, this solution has also a drawback, as the sizes of SRs are variable, and some of them may become extremely large.

## 6.9 Replica Placement

In this section, we outline how the gradient network topology may be exploited by a master-slave database replication strategy. In particular, the master-slave paradigm can be used to replicate the SR/KB. We present an approach, where the replica placement is based on peer utility, available resources and demand. Due to the information contained in the network structure, the selection of high utility peers suitable for replica placement does not require communication between all peers in the system.

In order to create a new instantiation of the SR/KB, the owner creates a *master replica* first (also known as *master copy*). There is only one master replica for a SR/KB instance and it can be responsible for handling and synchronising updates on this SR/KB instance. Ideally, the master replica should be created on a dependable peer located in the core of the network, for example on one of the catalyst peers. However, the owner is free to choose an arbitrary peer, in particular, the owner's own machine, as the system does not restrict users to place data in any particular location.

Subsequent replicas of the SR/KB (named *slave replicas*) are created automatically, on demand. We restrict the set of peers that are allowed to create SR/KB replicas to those with utility above a replica-suitable threshold. A *master replica threshold* defines a minimum peer utility to host a master SR/KB replica, and a *slave replica threshold* defines a minimum utility level to host a slave SR/KB replica. It is important to note, that the system does not require any form of consensus between peers on the threshold values, since the thresholds can be determined by each peer individually.

There are two general approaches for the slave replica placement. In the *server-initiated* strategy, a peer that already hosts a replica, in particular the master, requests a new replica placement on one of its neighbours, for example, when the number of user queries exceeds some critical level and a new replica is needed to handle the load. It is crucial in this approach that a peer initiating the replication must be able to find other peers suitable for hosting new replicas, i.e., peers with utility above the slave replica threshold. This should be satisfied in the gradient topology, since all peers storing replicas are clustered in the highly connected core, and share a similar, high utility status. Search messages do not need to be propagated to lower utility neighbours, since all high utility peers are located in the core of the network (see Figure 6).

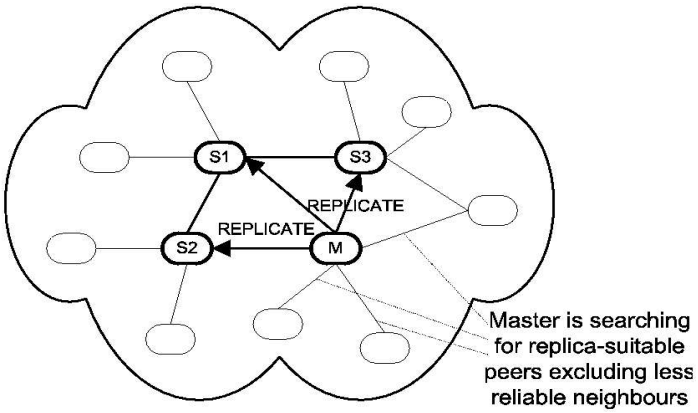


*Figure 6 Master compares the reliability of peers S1, S2 and S3 to select the best peer for slave replica placement.*

In the *client-initiated* approach, the replication is initiated by a peer that doesn't maintain a replica yet. Such a peer, in most circumstances, forwards user queries to its neighbours. However, if a certain criteria is met, and the peer possesses enough resources, it may decide to store a replica. The simplest criteria for client-initiated replica placement is described by a threshold for the frequency of queries for a database, which is easy to implement since it requires that peers only maintain the statistics of queries. New replicas may be created with a probability *proportional* to the query frequencies, which has the advantage that popular data is stored by many peers and can be accessed easily. Another technique, called *square-root replication* [23], calculates the probability of replica placement from the query search area, the greater the search area the higher the probability, so that no replicas of the same data are created next to each other. This approach was shown to perform better than proportional or uniform replication, and has the advantage over proportional replication that rare data is less likely to be removed from the system [24].

Replicas are also removed on demand, adaptively. If a peer decides that the cost of a replica maintenance is higher than the benefit of handling queries, the replica can be deleted. Alternatively, replicas might be selected for removal with the *LRU* strategy (Least Recently Used) as in FreeNet [25].

## 6.10 Replica Update

SR/KB replicas must be synchronised between each other after update operations. In the simplest design, we can add a constraint that an update operation, either a modification, an addition or a removal of an entry in the KB or SR, must be performed on the master replica. This requirement shouldn't significantly reduce the scalability of the system since we expect that the rate of updates will be much

lower than the rate of queries in the DBE SR/KB. If an update is delivered to an ordinary replica, the replica forwards the update to its master, and the master propagates the update to all replicas. This guarantees that concurrent updates from different peers are serialised and will be sent in the same order to all copies of the database, hence, there are no write-write conflicts.

However, if the DBE system supports the notion of identity, the updates can be performed on any SR/KB replica, including slave replicas. This is allowed, since each entry in the SR/KB can be modified only by its owner, and hence, if two independent SMEs perform concurrent updates on two different replicas, they modify different entries in the SR/KB and their updates can be easily merged. In other words, SMEs perform updates always on different data sets, and therefore, there are no write-write conflicts between them.

## 6.11 Replica Consistency

We propose a probabilistic replica synchronisation algorithm based on *periodic gossiping*, where updates on the SR/KB are continuously exchanged and merged by peers maintaining SR/KB replicas. Each peer periodically selects a random SR/KB replica and synchronises with it. The frequency of replica synchronisation can adapt to the available network capacity, i.e., the consistency of replicas can be improved by increasing the synchronisation frequency, at the expense of network traffic generation and bandwidth consumption. Replicas are also synchronised after a peer restart and after a peer has recovered from failure.

The replica membership information is managed using the same gossip mechanism. Each peer hosting a SR/KB replica maintains a list of other known replicas. The lists are exchanged between two peers each time the peers synchronise their replicas. This way, the information about newly created or deleted replicas is quickly propagated to other peers. Epidemic protocols for database replica synchronisation are described in [26] and [27].

The system provides eventual consistency of the replicas [28]. The core peers storing replicas should be well-connected, have high bandwidth and processing power, which should allow them to maintain a relatively high number of good quality connections, which in turn should enable fast data dissemination and frequent replica synchronisation.

## 6.12 Service-Oriented Routing

A searching mechanism enables peers to locate DBE infrastructural services, such as the SR/KB. As it is assumed that the copies of the SR/KB will be replicated in the system, a searching algorithm should route queries to locate the nearest available replica for that query. Traditional unstructured systems, such as Gnutella, have used flooding algorithms for finding resources. This approach works well for highly replicated data, however, it doesn't scale. A number of techniques have been proposed to improve search in unstructured peer-to-peer networks, such as random walk, iterative deepening and routing indices [29][30][24]. For an overlay topology, we are designing a *gradient routing* algorithm that will be based on two main factors:

- Neighbour utility information (utility *gradient*)
- Heuristic values, learned by the system (e.g. as in FreeNet [25] or Sample [31])

By increasing the probability of forwarding queries to high utility neighbours, the algorithm can effectively route queries towards the core of the network.

This work is an extension of the work on CRL in Workpackage 23 of the implementation of the Fitness Landscape.

## 6.13 Basic Overlay Operations

- *Insert* request are sent to one of the core peers where the master replica is created.
- *Update* requests are sent to the nearest SR/KB replica. The replicas synchronise their updates by periodic gossiping.
- *Delete* operation is analogue to the update, it's sent to one of the replicas and it's propagated to all other replicas by gossiping.
- *Lookup* operation is performed by routing the request to the nearest peer which stores a replica of the required data (see Figure 7 and section Queries below).

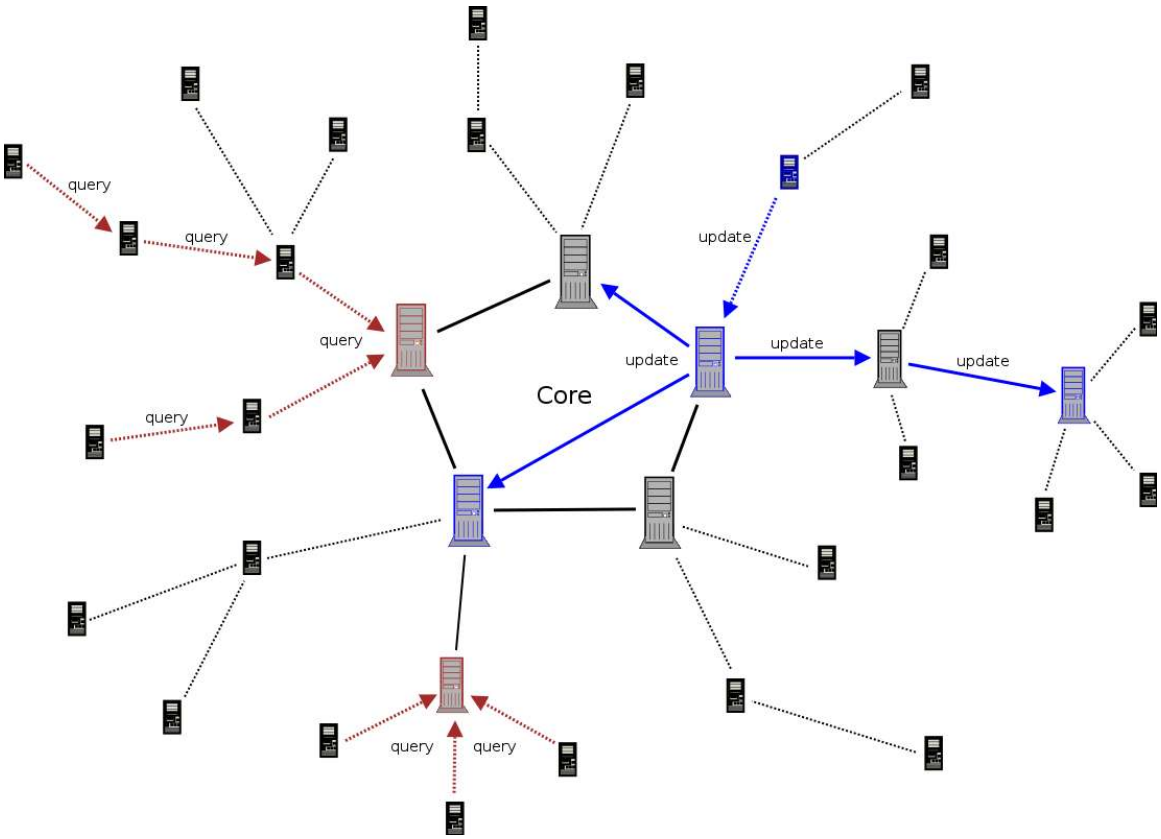


*Figure 7 Self-organising, emergent structure of a peer-to-peer network*

## 6.14 Queries

It has been proposed in the DBE to use different query languages, such as X-Query, for the Knowledge Base and the Semantic Registry. At implementation level, each SR is represented as a table in SQL and each service manifest attribute corresponds to a column in an SQL table.

Since data is partitioned and distributed by SRs in our model, a simple query requesting data from one table (SR) can be answered by a single peer which stores a replica of this table (SR). Such a peer can be found using the routing mechanism described in sections above, potentially in close proximity from the requester. For example a query

SELECT * FROM Hotels WHERE Location = “Ireland”

is routed to the nearest peer (peer A) maintaining a replica of the “Hotels” table. Peer A calculates the result of the query and returns it to the sender.

In case of more complex queries, where several tables (SR service manifests) are combined, several peers with replicas must be located and contacted. For example a query

SELECT * FROM Hotels, Restaurants WHERE Hotels.Location = Restaurants.Location

is routed from the requesting peer C to peer A storing “Hotels” SR. Next, peer A locates a nearby replica of “Restaurants” SR stored on peer B, and sends a list of hotels to this peer. Peer B at this stage has data from from both hotels and restaurants tables, so it can execute the query and send the result to the searching peer C (see Figure 8).

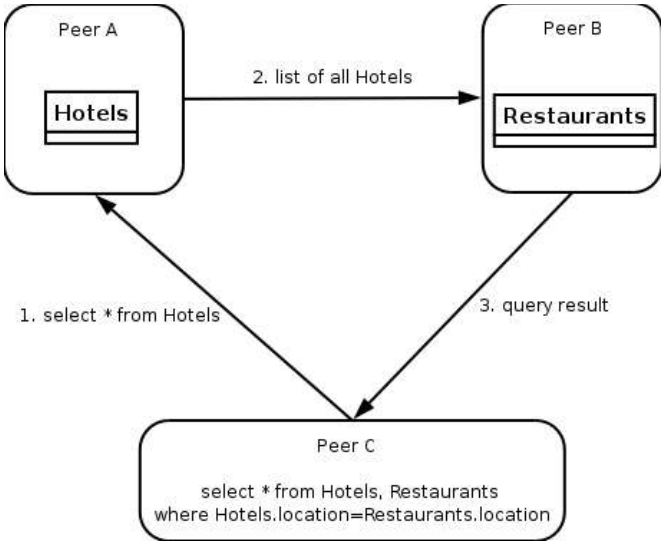


*Figure 8 Execution of a SQL statement*

In traditional Relational Database Management Systems (RDBMS) the results of join operations can easily be stored in main memory, whereas decentralised systems require transfers of significant amounts of data over the network. Bloom filters [32] are often used to limit the size of transferred data. A bloom filter is a space-efficient probabilistic data structure that allows to test whether an element is a member of a set. False positives are possible, but false negatives are not. In the example above, instead of sending a list of all hotels to from A to B, peer A can send a smaller (in terms of the number of bytes) bloom filter of hotel locations to B. Peer B, in turn, can send back to peer A a list of all restaurants which location is satisfied by the bloom filter. Finally, peer A eliminates all false positive results (i.e. all restaurants that are not related to any hotels) and sends the result of the query to peer C.

Expensive transfers of data can be avoided by a smart replication or caching algorithm, which could place replicas of frequently joined tables on the same peer or on peers located in close proximity. Distributed query optimisation techniques are discussed in [33] and [34].

*Keyword searching* is another approach to content-based searching, often used on the world wide web. Its aim is to locate data objects associated with a given key. It's usually implemented by maintaining an inverted index of keywords occurring in system's data, often distributed among multiple peers. These techniques are described in [33] and [34], although applying them to the DBE peer-to-peer model might require further research.

An alternative to supporting distributed range queries and keyword queries is to fully replicate the content of the KB and SR at all replica-suitable nodes.

## 6.15 Gradient Topology Simulation

We assessed our gradient topology approach by modelling a P2P network in RePast, a multi-agent simulation toolkit for large-scale systems. The simulation was implemented in Java. A network was created consisting of over 100,000 peers, which we believe is sufficiently large to model realistic large-scale applications. The experiments ran on a Pentium 4 machine with a 3GHz processor and 3GB main memory.

The simulation is based on a discrete time model and all events are executed in discrete time steps. The actions performed by peers are synchronous, however, our algorithm does not rely on the peer synchrony and hence the results obtained in the experiments can be generalised for asynchronous environments as well. Following the assumptions of decentralisation and a lack of a global view of the system, each peer maintains a limited number of neighbours and performs actions using local knowledge only. We assume also a scale-free distribution of resources with the maximum number of peer connections following the power-law (Pareto) distribution, starting from 10 connections and reaching about 50 neighbours for the most powerful peers. Similarly, the utility distribution follows the power-law. We model the network growth by adding new peers at each step of the simulation, where we start with a network containing only one peer, and at each time step the size is increased by 1 percent. Additionally, at each step a number of peers are removed, which models random failures or peer departures, with the probability of a peer departure being inversely proportional to its utility. Bootstrapping of the system is implemented with a centralised name repository containing the 50 most recent peer names, where peers are added and removed in FIFO order. Figure 9 shows the pseudo-code of the simulation.

```
for (N steps of the simulation) do {
  increase the number of peers by 1%;
  probabilistically remove peers according to their utility;
  forall (peers p in the network) do {
    p.step();
  }
}
```

*Figure 9 Main loop of the simulation.*

Figure 10 shows the randomised algorithm performed by each peer at every step of the simulation. A peer maintains two independent sets of neighbours, randomly-selected connections, and greedily-selected connections to peers with similar utility status. The latter set is twice as large as the former. At every step, if the number of neighbours of a peer reaches the maximum, the peer drops a random connection, and attempts to establish a new one by gossiping with a neighbour.

```
if (number of neighbours = MAX_NEIGHBOURS) then {
  disconnect random neighbour;
}
if (number of similar neighbours < MAX_SIMILAR) then   {
  choose randomly neighbour p from all known neighbours;
  get all neighbours n1..nk from p;
  choose peer n with the most similar utility from n1..nk;
  connect to n;
}
if (number of random neighbours < MAX_RANDOM) then {
  choose randomly neighbour p from all known neighbours;
  get all random neighbours n1..nk of peer p;
  choose randomly peer n from n1..nk;
  connect to n;
}
```

*Figure 10 Algorithm performed by an agent at one step of the simulation.*

Figure 11 presents a visualisation of a sample network consisting of 200 peers, generated by the neighbour selection algorithm, where we can see that the topology evolved to the desired form where the highest utility peers are clustered together and constitute a stable core.

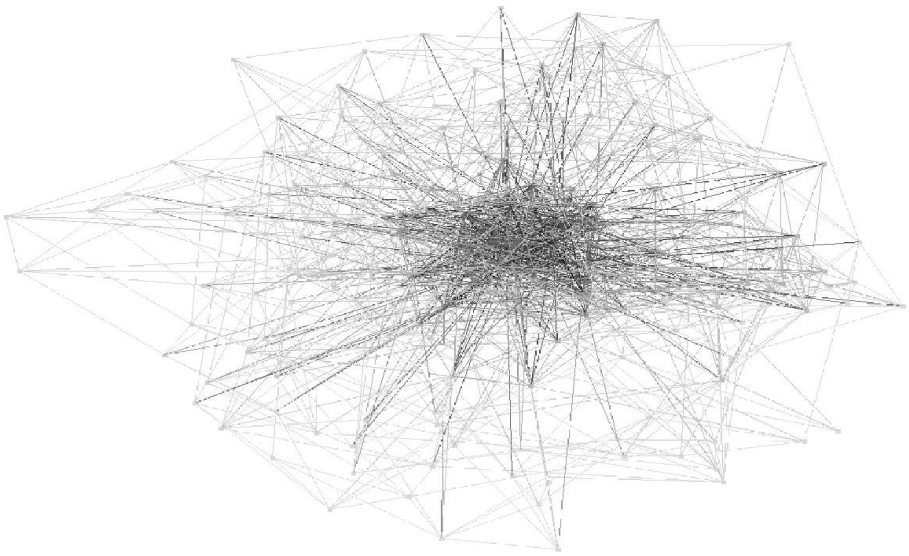

*Figure 11 Visualisation of a 200-node network in our RePast simulation using the Fruchmen-Reingold algorithm. High utility peers are marked with dark colors. Stable core of the network is visible in the center.*

Figure 12 shows the results obtained of our simulation. We can see that the average path length between peers is relatively low (about 5-6 hops for 100,000 peers), and that it varies with peer utility. The average distance between the highest utility peers is lower than the average distance between lower utility peers. This confirms our observation that the highest utility peers are highly connected with each other and form a stable core of the network.

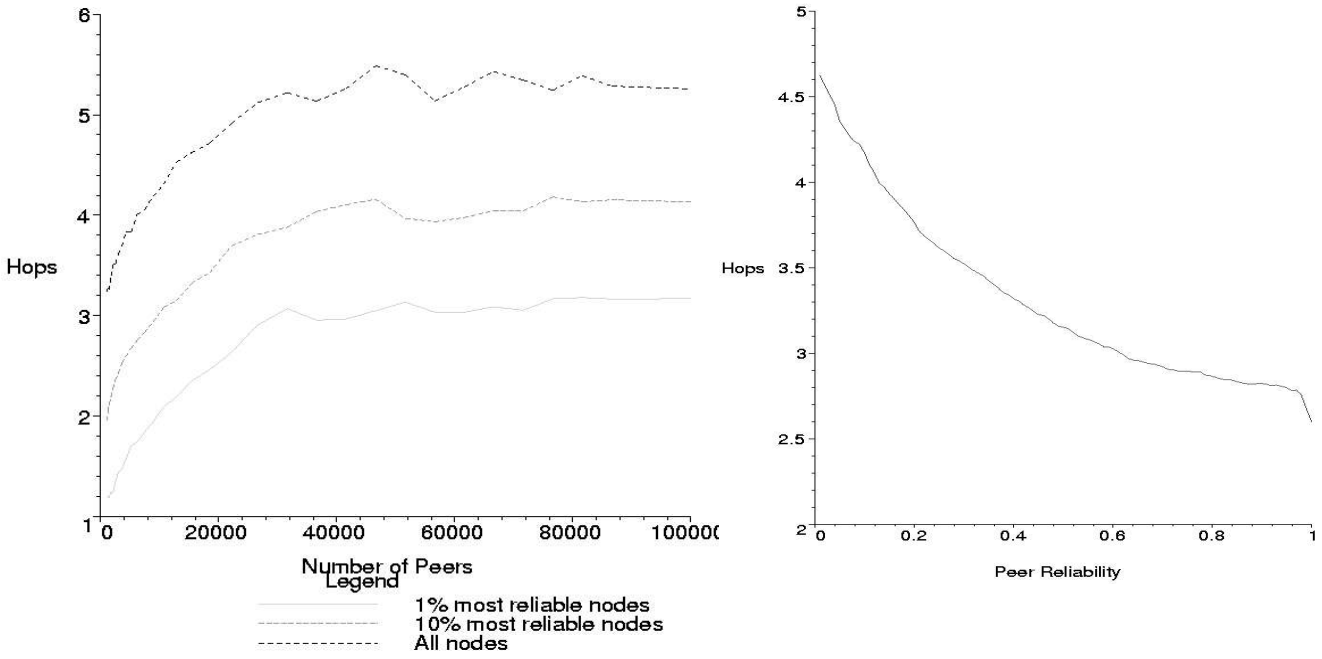


*Figure 12 Results of the neighbourhood selection algorithm. a) Average distance between peers as a function of the network size. b) Average distance between peers as a function of peer utility, network size 100,000 peers.*

The main advantage of the gradient topology is that the network structure contains information about the peer utility, which allows the peers to discover other high utility peers without flooding the entire network with search messages. The gradient topology allows the search space to be limited to a small subset of all peers in the system. This property allows us to address the problem of dynamic replica placement, master replica election, and replica discovery in an open, decentralised environment. The gradient topology also reduces the cost of replica maintenance, since peers storing data replicas are located close to each other and are connected by stable, high-capacity routes.

## 6.16 Related P2P Work

Most existing P2P systems that exploit the heterogeneity of their environment are based on structured P2P overlays. In Chord [11], it has been noted that a single physical peer can host multiple *virtual peers*, and therefore, the heterogeneity of resources in a DHT network can be addressed by assigning more virtual peers to higher performance hosts. OceanStore [35] proposed to elect a primary tier "consisting of a small number of replicas located in high-bandwidth, high connectivity regions of the network" for the purpose of handling updates, but no specific algorithm for the election of such a tier is presented. Mizrak et. al. [18] propose a super-peer based approach for the exploitation of resource heterogeneity, however, unlike our architecture, this approach relies on a DHT overlay.

In the area of unstructured P2P networks, a lot of work has been devoted to searching (e.g. [29][30]) and to replication strategies [24][23], but there has been limited research on network topology generation and peer neighbourhood selection algorithms. Yang and Molina [17] investigate general principles of superpeer-based networks and give practical guidelines for the design of such networks, however, they don't describe any specific algorithms for super-peer election or network structure maintenance. Jelasity [20] proposes a similar approach to our work in terms of creating and maintaining an overlay topology. However, he does not address the problem of reliable peer discovery and dynamic replica placement in an open P2P system.

# 7 DHT Overlay

This section presents the design of the structured overlay network that is used in the DBE system to locate SME hosts that provide services corresponding to the identifiers obtained from SR/KBs. The overlay is also used for selecting relay peers that act as proxies for peers which communication with other peers is restricted by NATs or firewalls.

## 7.1 Introduction to Distributed Hash Tables

Distributed Hash Tables (DHTs) are a class of peer-to-peer systems that provide a function of mapping keys to values. In DHTs <key, value> pairs are spread among peers participating in the system and efficient algorithms for key lookup, inserting and removing are provided by a system. Many DHT systems rely on structured overlay topologies, where each peer is assigned a random unique identifier and responsibility for part of the key space. Peers store all pairs whose keys fall in their part of the key space and provide operations such as lookup, remove, insert, which are routed in a multi-hop fashion (i.e. from one peer to another). State-of-the-art DHT systems provide efficient algorithms to performing these operations in time *O( log(N) )*, where *N* is the maximum number of peers in the system.

The DBE system must provide an efficient way to find services and corresponding service proxies identified by SMIDs (provided by service manifests). Since DHTs provide similar functionality, i.e. map keys to objects, and their high performance is already proven in P2P systems, it is logical to use them to locate services in the DBE P2P architecture. However, a significant shortcoming of DHT systems is their inability to handle clients behind NATs and firewalls. DHT implementations assume that all nodes have public IP addresses and thus any two nodes can communicate with each other. However, peers behind NATs cannot establish a direct connection between each other.

Existing DHT-based P2P systems include CAN [36], Chord [11], Pastry [37] and Bamboo [16]. We believe that Bamboo (a variant of the Pastry algorithm) is an appropriate base for building the DHT overlay for DBE since it already has a mature DHT implementation and provides desired properties such as good churn handling and the exploitation of network locality when routing messages, according to a scalar proximity metric such as the latency of a route [16].

Bamboo uses essentially the same algorithm for routing messages as Pastry. Each peer is assigned a 160-bit peer identifier. The identifier is generated randomly when a peer is created and it is assumed that the resulting set of identifiers is uniformly distributed in the space of all possible values. For a network of *N* peers Bamboo routes to the numerically closest peer to a given key in less then *log(N)* steps. The peer's neighbourhood is constructed from peers with identifiers that have the same first *i* digits and differ only at the *i+1* position. When routing, a peer normally forwards the message to a peer whose identifier shares with the key a prefix that is at least one digit longer than the prefix that the key shares with the current peer's identifier. Figure 13 shows the example scenario, where the peer with identifier *65a1fc* routes a message with key *d46a1c*.

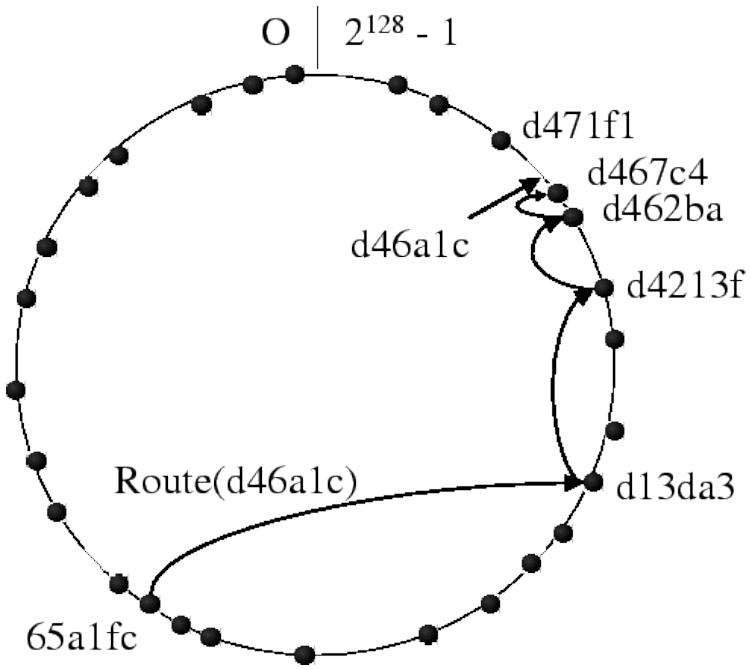


*Figure 13 Routing a message from node 65a1fc with key d46a1c.*
*The dots depict live nodes in Pastry/Bamboo's circular namespace.*

As the network grows in size, each peer on the path can choose among more neighbouring peers that are appropriate for forwarding the packet, hence, it can use additional criteria, such as the estimated route latency, to select the next hop.

## 7.2 Discovering Service Types using a Service Manifest Identifier (SMID)

A SMID is a unique identifier of a Service Manifest. Service manifests describe a service type, and a set of associated models. Service Manifests can be retrieved from the Semantic Registry using a Query tool, while a set of models referenced inside a Service Manifest can be retrieved from the Knowledge Base. These models include an interface model described using the Service Definition Language [38], and Business Modelling Language (BML) Models.

The SMID does not, however, identify specific service instances deployed in the DBE. There may be many instances of service manifests with the same SMID. As such, the SMID cannot be used as an identifier to locate specific service instances deployed in the DBE. For that purpose, we introduce the notion of a persistent, unique identifier assigned to each deployed DBE service, called SERVICE_ID. Multiple SERVICE_IDs can be associated with the same SMID.

## 7.3 Discovering Deployed Services using a SERVICE_ID

Every service in DBE has an associated SMID, although there may be many service implementations for each SMID. We introduce a SERVICE_ID as a unique persistent location-independent identifier assigned for each service deployed in the DBE system. Our proposed DHT protocol is then used to map SERVICE_IDs to service endpoints or their proxies. The SERVICE_ID is created by concatenating SERVENT_ID and LOCAL_SERVICE_ID identifiers (see Table 1). The SERVENT_ID is a random globally unique DHT identifier of the servent host, which is a SME's gateway for all its services. The servent's role in the DHT overlay is forwarding requests to services deployed in the company's LAN

and uniquely identified within a company by the LOCAL_SERVICE_ID identifier. Therefore, by using the suggested SERVICE_ID format, all services of a particular SME will have the same prefix (i.e., SERVENT_ID) and thus, the prefix-based Bamboo's routing algorithm will route all lookups through the SME's servent that knows how to find the hosts on which the services are deployed. Section 8 explains how DHT identifiers are generated locally. This enables SMEs to maintain proxies for their own services and thus provides fault tolerance in the sense that when a service is deployed and active, the proxy is also available. Moreover, we achieve location-independence since a service can be moved from one host to another within a LAN while only the local servent has to be informed. We suggest that the SERVENT_ID identifier consists of 160 bits (which is Bamboo identifier's length), and the LOCAL_SERVICE_IDSERVICE_ID consists of 32 bits. Hence, SMIDSERVICE_ID consists of 192 bits.

| SERVENT_ID | LOCAL_SERVICE_ID |
|---|---|

*Table 1 The DBE SERVICE_ID Format*

### 7.4 Design of a DHT Algorithm for DBE

As mentioned in section 2, DHT routing algorithms require that peers are not behind port-restricted NATs or firewalls preventing communication with other peers. Peers that have public IP addresses and are not blocked by firewalls will be called in this document open peers, in contrast to peers behind NATs or firewalls which will be called restricted peers.

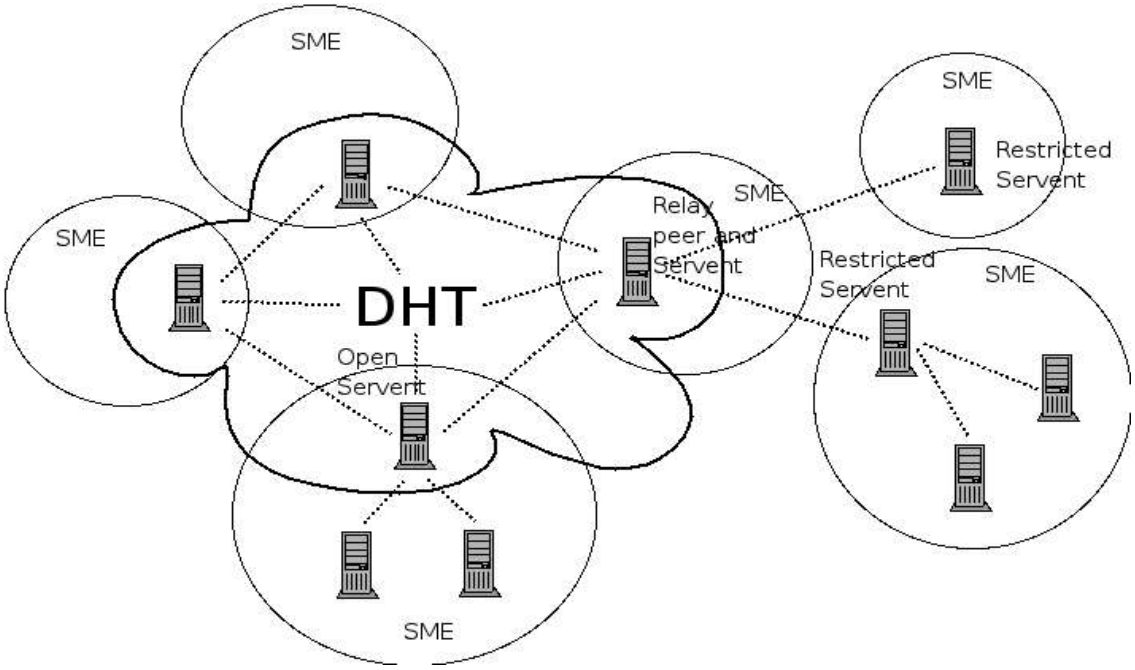


*Figure 14 The DHT Architecture for Open Peers and NAT Restricted Peers*

Each running service will be identified by SERVICE_ID whose particular format causes DHT lookups and method invocations to be routed to the servent host, which forwards it to the host that provides the service. Therefore, for each SME, only one host (the servent) will participate in the DHT routing

protocol. However, there is an exception when a servent resides behind a NAT. In such a case, the servent will select a peer in the DHT network that will create a 'virtual server' [39] (or virtual node) identified by the SME's identifier and effectively act as a relay peer for the restricted servent. A virtual server looks like a single DHT node with its own identifier (in our case, SERVENT_ID of the restricted servent), routing table and is responsible for routing lookup queries and service invocations. In this P2P design, we suggest that proxy objects are always stored on the hosts that actually provide related services, even when these hosts are restricted by NATs (i.e. service lookups and responses containing proxy objects will be forwarded via relay and servent peers). Figure 14 shows an example DHT network with both open and restricted servents and relay peers.

One disadvantage from the perspective of the SME providing a relay peer that hosts virtual servers is higher bandwidth utilisation which is proportional to the number of hosted virtual servers. Moreover, when a relay peer crashes, all restricted peers that were served by this peer have to find other relay peers. When a restricted servent notices the relay peer's failure (it notices this immediately since they keep a TCP connection constantly open), it selects another relay peer and asks it to create a virtual server identified with the same SERVENT_ID as before. From now on, the new relay peer will route messages to the restricted servent. After a relay peer failure, all service proxies requested before that are stored by customers become invalid since they carry the IP address of the relay peer that failed. Customer will notice this as soon as they try to invoke a method on the proxy and eventually request a new proxy (i.e., with the updated relay peer's address) from the DHT system. Figures 15 and 16 show an example of virtual server migration from one relay peer to another. In Figure 15, the servent with the identifier 20 serves as a relay peer for two restricted servents (50 and 60), hence, in the DHT overlay its machine acts as peers with identifiers 20, 50 and 60. When the relay peer fails (Figure 16) the restricted servents select another relay peer. Since the migration of all virtual servers from one relay peer to another is expensive, relay peers must be selected with caution. The proposed mechanisms for selecting relay peers are described in the 'Relay Node Selection Algorithm' subsection of this section.

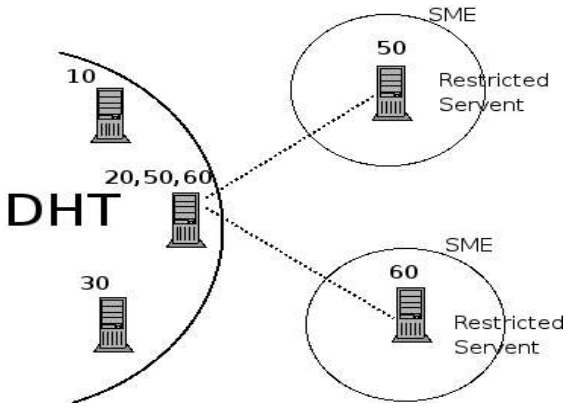


*Figure 15 Before the virtual server migration*

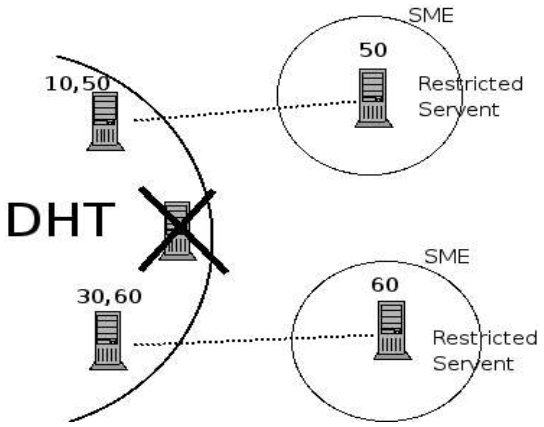


*Figure 16 After the virtual server migration*

### *Fault Tolerance*

Our peer-to-peer overlay for accessing services on the servent will have inherent fault tolerance. Proxies can be hosted on the servent host that provides services and therefore, if the servent is not available, we can assume that the machine hosting the service is also not available. When none of the SME's hosts have a public IP address, the servent peer has to keep an open connection with the assigned relay peer, hence it will immediately notice when the relay peer fails and will be able to reconfigure to use a new relay peer.

The P2P architecture also adapts to peer failures. If one of the peers on a DHT routing path fails, the system automatically re-routes requests and the query can still be completed.

### *Queries*

Queries are routed to SMEs machines that maintain the matching proxy or endpoint. The proxy or endpoint information is sent directly to the requesting peer or via a relay peer if it is behind a NAT. The query cost is small, i.e. each query is routed through only about $O(\log_2 N)$ peers.

As we do not expect that SMEs will perform general searches for service proxies (SMIDs will be discovered in the Semantic Registry and SERVICE_IDs will be discovered in a Lightweight Discovery Service), the lack of a general, free-form search is not significant.

### *Updates*

The cost of updates such as insert/update/remove is negligible since proxies or endpoint information is stored on the servent on the SME's internal network. Hence, when a service is inserted/modified/removed, it is sufficient to update the proxy or service endpoint on the SME's servent. Since proxies or service endpoint information is not replicated and stored only on the SME's own trusted network, it will be much easier to provide support for authentication and authorisation when proxy updates are requested.

### *Relay Node Selection Algorithm*

Relay nodes will be used to forward queries and method invocations to restricted servents that provide DBE services. Therefore, relay nodes will have to be selected from peers that are not restricted by

NATs or firewalls. The algorithm used for selecting relay peers must take into account many factors. The potential relay peer should have high uptime (peers with higher uptime have higher probability of staying long in the system), low latency (relay peer will communicate frequently with assigned restricted peers) and must have enough available resources to create a virtual server. Trust relationships may also be exploited when selecting a relay peer, i.e. a relay peer must be trusted since it routes all messages to the restricted peer. Moreover, the relay node selection algorithm must adapt to peers inherent heterogeneity and dynamism in the environment. For instance, when a relay peer becomes overloaded, some virtual nodes might have to be migrated to another relay peer. The details of this complex algorithm are still to be finalised, whereas in the first design a relay peer can be selected randomly from peers not restricted by NATs/firewalls, leading to a uniform distribution of the relaying effort among those peers.

### *Advantages Over the Current Design*

The presented fully decentralised approach is expected to improve the robustness of DBE, to improve the query performance by exploiting network locality when routing towards destinations and to distribute the load fairly among all peers in the system.

# 8 DBE Decentralised Identity

This section describes our proposal for the representation of the identities of DBE entities and the life-cycle of these identities. It also discusses remaining issues concerning security and trust.

## 8.1 Proposed Identity Representation in DBE

In the spirit of a self-organised DBE, we propose a entity-centric identity management solution rather than a system-imposed identity management solution. An entity-centric identity management means that any entity specifies its identifiers and credentials at time of interaction. Thus, entity-centric identity management is inherently decentralised. Third-parties may be used to store and retrieve identifiers and credentials but this is not mandatory. If two external DBE entities are given the same identifier and credentials, they can therefore deduce that they are interacting with the same Identity in DBE (IDBE).

We propose that any IDBE corresponds to the secure hash[1] of the public key of a generated asymmetric key pair. The certificates would be formatted according to X509 [40] and we would export the private key and its certificate protected by password thanks the PKCS#12 format [41].

IDBEs – the identifiers and their associated credentials – would be stored in a Credential Server (CRES), similarly to the GRID MyProxy credential repository [42].

The CRES may be either local or remote and would be used to retrieve proxy credentials [43] when needed. The advantage of remote CRES is that users can use IDBEs on different computers without complicated tasks to move the long term credentials into these different computers. When the CRES is managed by a professional DBE CRES provider, the long term credentials are protected by the security staff of the provider.

Any DBE entity – provider, consumer or part of the DBE infrastructure – would present at least one IDBE. Thus, any DBE entity is free to own one or several IDBEs and to link them by signatures. This is an advantage from a privacy point of view since the IDBEs can act as pseudonyms. This also means that mutual authentication between consumer DBE entities and provider DBE entities is possible, fulfilling a requirement for the accounting workpackage in DBE.

IDBEs may be certified by Certificate Authorities (Cas), for example, with regard to their binding with real-world identities. In fact, we allow the DBE entities to use a spectrum of technical trust in IDBEs from self-signed, to certified by web of trust and free CAs, such as Cacert [44], to professional certifications including insurance, such as Verisign Netsure [45] and Fraud Protection Services [46].

One of the goals of DBE is to foster business, which includes contracts and billing. Therefore, many regulatory issues must be taken into account as D32.1 underlines [47]. Non-repudiation of actions and long term (several years) storage of evidence are required. Asymmetric cryptography and certification authorities are important building blocks to comply with these requirements. The use of cryptoIDs-like identifier for the DBE entities ensures that the entities can prove the ownership of their claimed IDBE as well as to sign, validate signatures, encrypt and decrypt any messages sent from one DBE peer to another. The technique of interceptors has been thought to extend the current implementation for sending DBE messages with such asymmetric cryptography elements.

1 The security of SHA-1 has recently been challenged. So, SHA-256 may be used instead. Please refer to http://en.wikipedia.org/wiki/SHA_hash_functions.

## 8.2 IDBEs Life-cycle

Firstly, IDBEs are created and managed thanks to CRES servers. Secondly, a selection of these IDBEs are registered locally and then globally as DBE services.

### *IDBE Creation and Management via CRES Servers*

A CRES server allows a user, if granted, to:

- to create an account and protect its access via login/password;
- to create new key pairs and credentials;
- to manage the use of the credentials stored in this account;
- to retrieve proxy-credentials;
- to upload (proxy)-credentials.

This is similar to the GRID MyProxy credential repository [42].

CRES servers will be accessible as DBE services.

### *IDBE Local Registration*

Any DBE Rich Client Application (RCA) can be populated with a number of IDBEs, either generated locally via the local CRES server or retrieved from a remote CRES service. In the DBE RCA, there is a Graphical User Interface (GUI), which allows the user to display the right SR and use it to search and retrieve his/her CRES services. It may even be CRES services with no previous account for the user and which allow the user to create a new account.

Then, the user can retrieve (or create) as many IDBEs from the selected CRES services to be registered inside the RCA. The registration should consist of the registration of proxy credentials corresponding to the selected IDBEs. The user may upload (proxy) credentials to remote CRES services if they wish.

### *IDBE Global Registration Service (GRS)*

The IDBE Global Registration Service corresponds to a DBE service that gives access to a particular IDBE. There is, so far, one method provided by this service but other methods may be added in the future. The method is called sayHello, has one TEXT parameter and returns a message containing the concatenation of Hello, TEXT and the current time and date signed by the IDBE private key.

Any GRS is registered in DBE as any other DBE services (please refer to the service registration use case in section 9).

## 8.3 Remaining Issues

In addition to authorisation issues, we discuss fully decentralised P2P systems from a security point of view, ending by an initial threat analysis.

### *Authorisation*

“Authentication is one problem and access control is a completely different one” [48].

Although the use of asymmetric cryptography for identity in DBE facilitates the assignment of authorisation credentials to IDBEs, the task to provide authorisation is far beyond the scope of task C18:

"Task C18: Distributed Identity System (TCD)

The distributed identity service will provide a DBE infrastructural service to uniquely identify DBE service providers. In order for the Distributed Identity Service to validate SMEs within the DBE it should be integrated jurisdiction-specific SME identification and authentication facilities. For this purpose, a framework will be built that can be specialised for each SME jurisdiction to validate SMEs. The Distributed Identity Service will use certification technology for SME identity and provide a framework for certification distribution and management. A possible adopter of the Distributed Identity Service would be the SME catalysts in the different EU regions".

### *Structured P2P Specific Issues*

The literature describes how attacks on structured P2P may happen when peers are free to place their identifier in specific zones of the structured naming space, even if the allocation is controlled by a centralised server [49][50]. Means are required to limit the number of peers that a unique real-world identity can introduce in the system either by means of payment for each new peer introduction or careful manual limitation of number of peers per real-world identity.

### *Initial Threat Analysis Discussion*

If we make the assumption that to compromise a personal computer is a feasible threat, one may argue that to locate identity management on the provider side is safer because dedicated security staff can ensure that it is more difficult to permanently compromise the servers. However, once the personal computer is compromised, the means to access the remote provider can be recorded and replayed. Thus, to rely on the provider's side security does not really solve the problem, especially because DBE providers may not have dedicated security staff employed.

As above, Sybil-like attacks are still possible and the self-allocation of IDBEs may allow an attacker to gain control of a specific portion of the structured P2P naming space. In order to mitigate these issues, we propose to force, when required, e.g., for structured P2P schemes, the DBE entities to present IDBEs certified by special CAs, who are known to restrict in terms of number of IDBEs per real-world identity. For example, the restriction may be based on an entrance fee or real-world credentials, such as passport numbers.

With regard to the regulatory issues on e-signatures and authentication mentioned in D32.1 [47], the establishment of a hierarchy of authentication for a specific sector will self-organise based on which CAs the interacting DBE entities allow. This will probably be based on third-party CA security services, such as Verisign, for valuable and high risk transactions.

The corruption of a CRES server implies that all the IDBEs inside the CRES are compromised. The use of one-time passwords may limit the damage due to compromised CRES and allow users to log into a CRES from untrusted computers without the risk to have their password recorded.

The revocation of credentials can be done based on current mechanisms for credential revocation. The first implementation of the identity service will not implement its own revocation mechanisms because such mechanisms are assumed to be provided by the potential external CAs involved in DBE, such as Verisign.

# 9 Component Relations

The purpose of this section is to show the role of the P2P architecture in the DBE system. The relations between components and the P2P architecture are presented in the form of use cases for both service providers and service consumers.

A Service Oriented Architecture (SOA) describes an architecture as a collection of services. Service Oriented Architectures are based around the principles of loosely coupled services, reusability, self-describing interfaces, composability, and language/platform independence. The idea of using service-oriented architectures to build distributed systems is not new thing, as we have had similar incarnations in CORBA and DCOM. However, in contrast to CORBA and DCOM, the main implementation of a SOA, Web Services, has been designed with explicit support for operation over the Internet. An illustration of a service consumer using a service is illustrated in Figure 17.

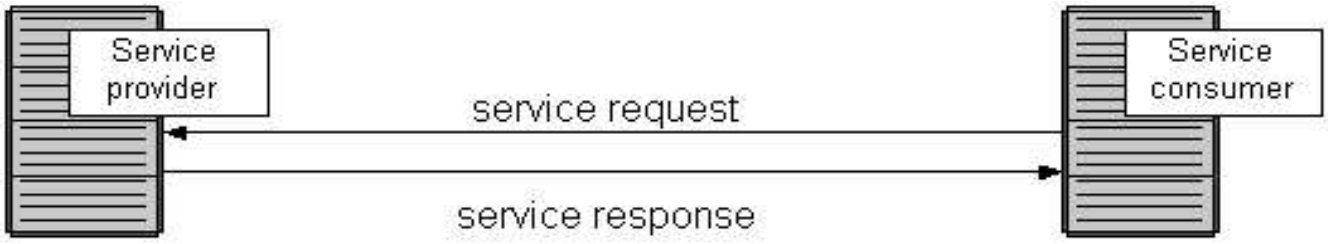


*Figure 17 Service Consumption in a SOA*

There are three main use cases that all SOAs fulfil. These are:

- Service Registration
- Service Discovery
- Service Consumption

In Web Services, service registration involves registering a service with a centralised UDDI server and service discovery involves browsing for the service at the same centralised UDDI server. In practice, these use cases are often skipped as consumers may know the location of their desired web service. However, in DBE these use cases are vital to building an ecosystem where services can be discovered easily, promoting competition and transparency among service providers. In this document we propose developing a P2P service discovery and registration architecture based on an unstructured P2P overlay topology (see section 6). The P2P system is based on the principle of clustering nodes with similar reliability and storing naming information on those nodes. Querying of those nodes will be routed using a heuristic search algorithm, for example, based on collaborative reinforcement learning (CRL) [51].

The consumption of services in web services involves using SOAP to route messages from a client to a static endpoint on a server identified using a URL. However, in a typical SME environment, hosts may have DHCP-allocated IP addresses or may be behind a Network Address Translation (NAT) Gateway. We propose a service consumption that uses relay peers to route requests to or from peers restricted by NATs/firewalls. Relay peers act as proxies in method invocations and as virtual servers for restricted peers in the DHT.

The deployment of web services involves deploying the service in an application server that hosts the service and marshals requests to and from the service over a protocol, such as SOAP. We are using the servent, based on the Jetty application server, to deploy services at the SME's host.

The final issue to consider in our architecture is the underlying service invocation platform. The servent is a RPC-based invocation platform, object serialisation over HTTP, which is not ideal given that the invocation path will span multiple hops in our P2P network. Another possibility would be to update the invocation protocol to a messaging or an event-based protocol, such as document-style SOAP messaging.

## 9.1 Lightweight Discovery Service

We propose the use of a Lightweight Discovery Service (LDS) as an alternative mechanism for service discovery. LDS is a DBE service itself and it maintains a registry of other DBE services. Services may be registered and unregistered by their providers independently from the registration in the SR/KB. Each registered service is associated with a description and a set of tags specified by the provider.

LDS allows users to search for services using keyword queries, in a similar way as google or other web search engines. Given a set of user-specified keywords, LDS returns a list services that match these keywords.

### *Categorisation of Services*

The LDS uses the notion of a folksonomy to categorise the services stored within it. The word folksonomy is an amalgamation of the words *folk* (people) and *taxonomy* (classification management). A folksonomy is the result when a set of items are categorized in a collaborative fashion using freely chosen keywords. It has the exact opposite approach to categorising data based on ontology, those being based on rigid structure and hierarchy. As such, folksonomies only arise in non-hierarchical communities (such as the web and unstructured P2P networks). Typically the organisers of the data categorised by the folksonomy are also its users it can be argued that folksonomies produce better conceptual models of the data being categorised. Folksonomies work best when there are a large number of entities describing the same data within a specific domain. This way, in a self-organised fashion, an emergent property of common understanding, semantics and categorisation, within the domain and amongst entities, arises.

The practise of organising and categorising data based on folksonomies is more commonly known as "*tagging*". Typically, when an entity "*tags*" a piece of data that entity associates a string, or multiple strings, with the data, providing information that describes (be it accurate or not) about that data. Already there is an abundance and very successful tagging service on the web. Examples that spring to mind automatically are flickr (http://www.flickr.com), an online photo storage service and del.icio.us (http:// del.icio.us), an online bookmark manager.

### *LDS and Folksonomies*

As the nature of the DBE is non-hierarchical and unstructured with an emphasis on lowest cost factor, using folksonomies is a most suitable and apt way to describe and categorise DBE services in a self-organising way.

## 9.2 Service Registration Use Case

When a service is created, it must be registered with a Semantic Registry and the Lightweight Discovery Service (LDS). Service Registration is performed by the Deployer Tool.

Firstly, the Deployer Tool deploys the service on the local servent, which creates a globally unique LOCAL_SERVICE_ID for the service. The Deployer Tool, then creates a service manifest for the service using the SDL, any BML available, and the SMID for the service. Finally, the Deployer creates a SERVICE_ID for the service by concatenating the SERVENT_ID and the servent-generated LOCAL_SERVICE_ID (see Figure 18).

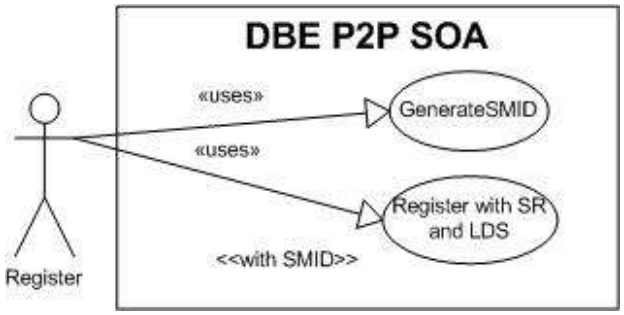


*Figure 18 Service Registration Use Case*

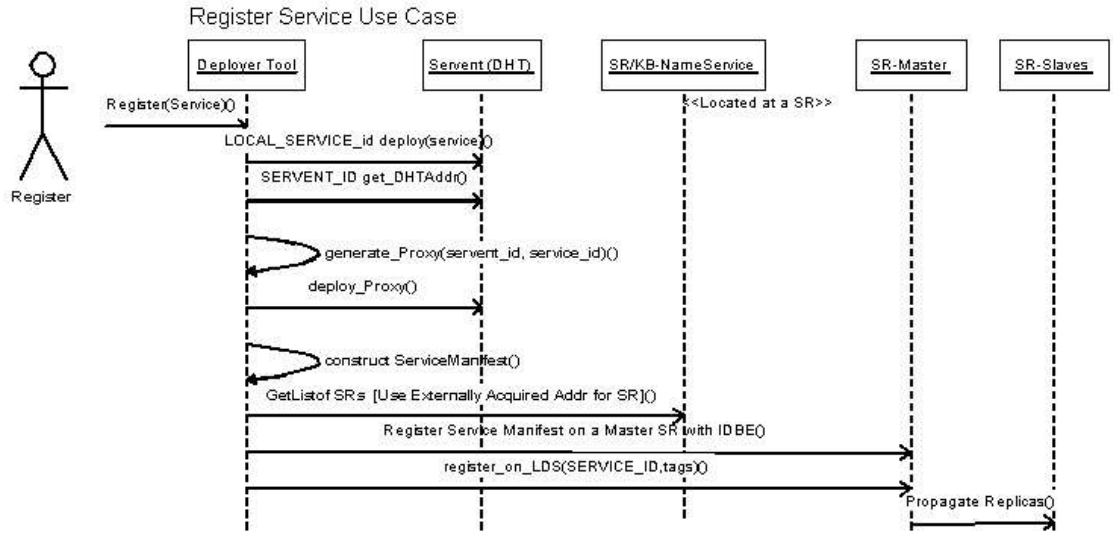


*Figure 19 Sequence Diagram of the Service Registration Use Case*

The Deployer Tool [52] can discover the appropriate SR/KB on which to register a service by using the SR/KB Name Service. An instance of the SR/KB Name Service is deployed on every node that hosts a SR/KB. The Deployer requires an externally acquired address for a SR/KB node, this could be a well known address in the first implementation of DBE (see Figure 19).

The SERVICE_ID along with a tag describing the service and the IDBE of the owner of the service are registered with the Master Copy of the Lightweight Discovery Service, colocated with the SR-Master. The Lightweight Discovery Service authenticates the Deployer before registering/updating the SERVICE_ID. The updates on the Master Lightweight Discovery Service are propagated to slave replicas of the Lightweight Discovery Service at some later bounded time, e.g. replicas are synchronised with the master every 10 minutes.

## 9.3 Service Discovery and Consumption Use Case

Services are discovered in a SR/KB using the Query Tool provided by TUC.

Currently, queries can be formatted using TUC's Query Language format or using XQuery. TUC plan to support keyword-based searches.

Queries will be forwarded to slave-copies of the SR/KB. Collaborative Reinforcement Learning [51] will be used to route queries to a slave-SR. The overlay network can be used to return results directly and cached DHT addresses to slaves can be used to optimise the lookups.

Queries may also be performed on the LDS using keywords.

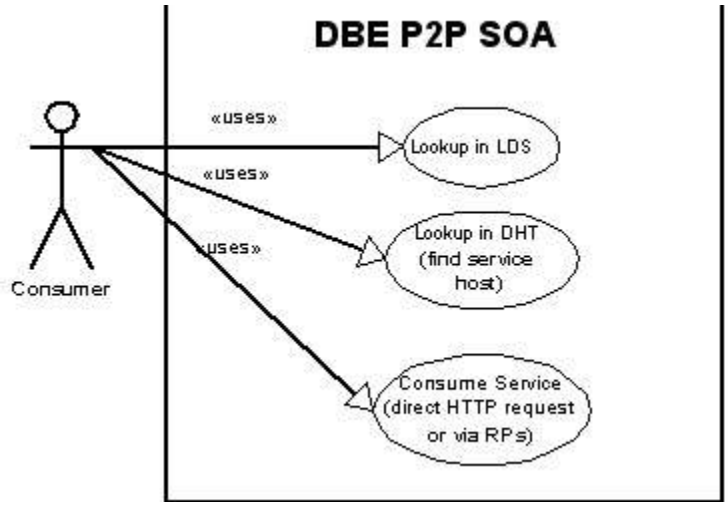


*Figure 20 Consume Service Use Case*

The SERVICE_ID is the DHT overlay network address of the service that can be mapped to a service endpoint information (IP address and port number of the host providing the service or its relay peer if it is NAT-restricted) through the Service Lookup component. The requests to consume a service can be routed to the relay peer for the service and then forwarded to the servent on the NAT-restricted peer, where the service is deployed (see Figures 20,21,22,23).

When the servent attempts to return the results to the consumer, it can send the results directly to consumer's relay peer. However, there is the possibility that the servent's relay peer has failed, so the client must also supply a SERVICE_ID for an endpoint on which to receive results. The SERVICE_ID is used in the event that either the consumer's or provider's relay peers has failed.

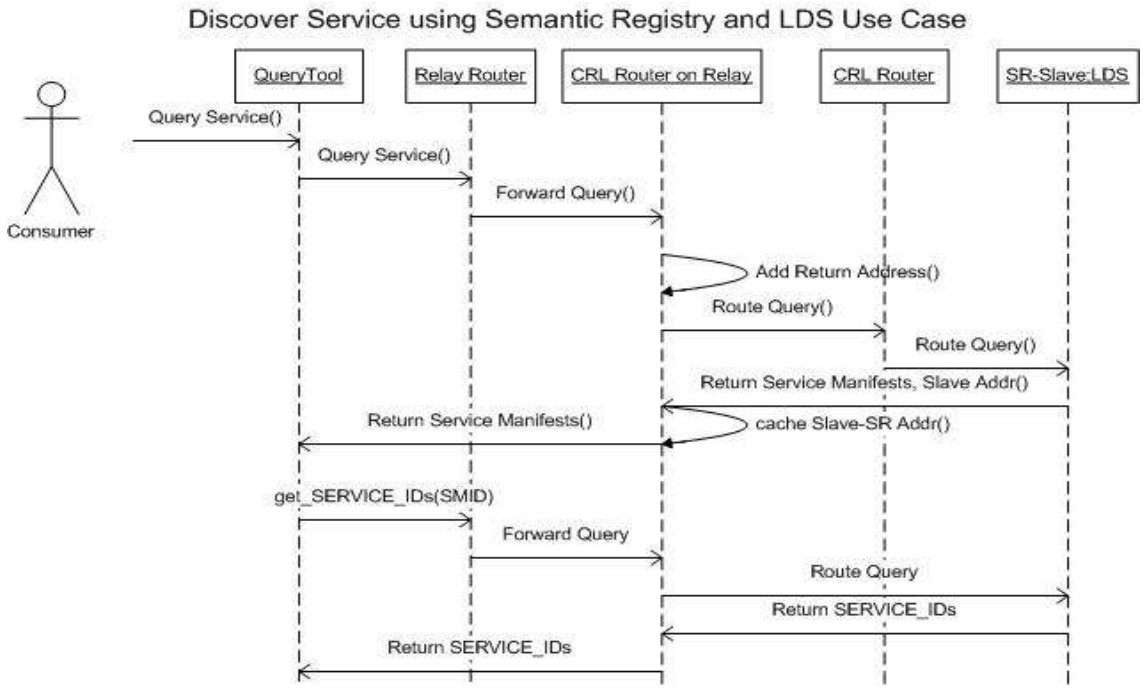


*Figure 21 Service Discovery by a Consumer using the Query Tool, Semantic Registry and LDS*

The problem of failure of relay peers in the DBE SOA opens up a new failure mode, which has not been encountered in existing RPC-style platforms. One potential means of minimising, or even eliminating, the use of the consumer's return SERVICE_ID for receiving the results of an RPC invocation is the use of replication to enable some level of utility in the DHT architecture.

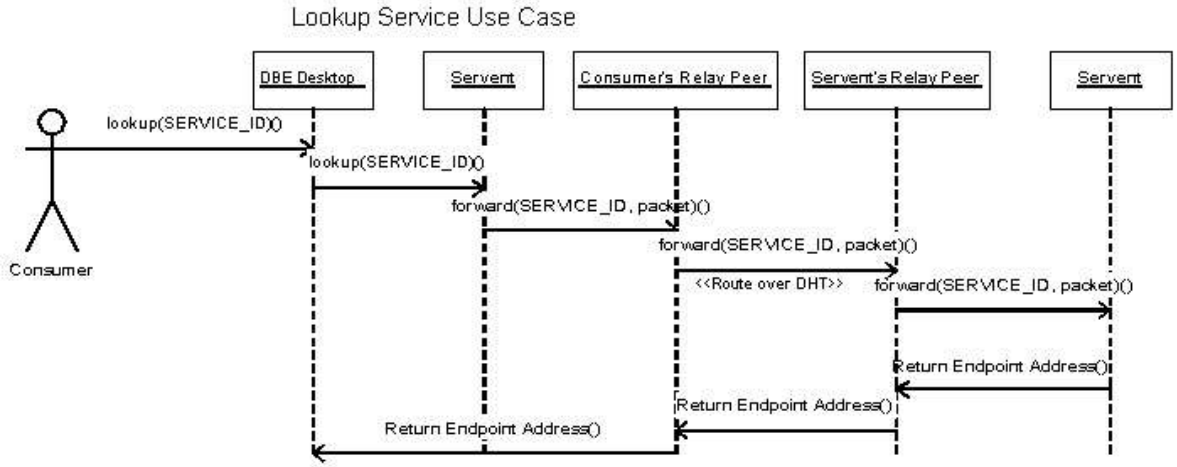


*Figure 22 Service Lookup using the DHT, Relay Peers*

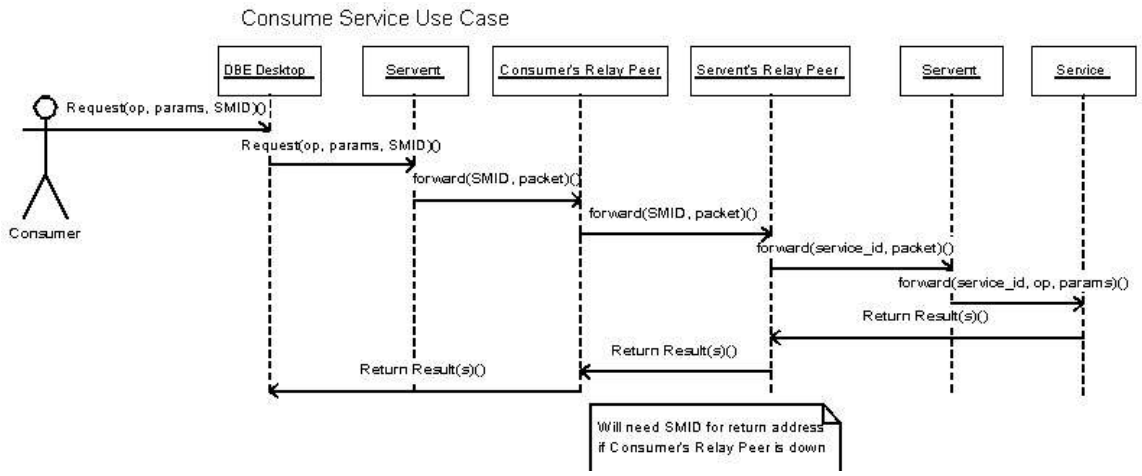


*Figure 23 Consume Service using the SMID, the DHT, Relay Peers and the Servent*

# 10 Conclusions

In this document, we firstly provided an analysis of peer-to-peer environments and DBE infrastructural services for the purpose of identifying challenges and potential approaches addressing these challenges. Based on this analysis, we proposed the P2P architecture for DBE services. The P2P architecture consists of two overlay network topologies optimised for different types of services. The unstructured P2P overlay, called Gradient Topology, was designed for the Semantic Registry, Knowledge Base and SR/KB Name Service. The P2P structured overlay, based on the DHT paradigm, was proposed for the Service Lookup, Identity Service, and Distributed Storage System. We also designed a decentralised identity model for the DBE that enables authentication in the untrusted Internet environment.

# Bibliography


1: Jeffrey O. Kephart and David M. Chess, The Vision of Autonomic Computing, 2003

2: Chris Anderson, The Long Tail, Wired Magazine, 2004

3: Bryan Ford, Pyda Srisuresh, and Dan Kegel, Peer-to-Peer Communication Across Network Address Translators, USENIX Annual Technical Conference, 2005

4: J. Rosenberg et al., Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs), RFC 3489, 2003

5: C. Jennings, STUN Protocol

6: Internet Society, Traversal Using Relay NAT (TURN), 2004

7: Qin Lv, Sylvia Ratnasamy, and Scott Shenker, Can Heterogeneity Make Gnutella Scalable?, IPTPS '02, 2002

8: Nektarios Gioldasis et al., Report on the P2P DBE Knowledge Base, 2005

9: Nektarios Gioldasis et al, P2P Report on the Knowledge Base, 2005

10: Antony I. T. Rowstron and Peter Druschel, Pastry: Scalable, Decentralized Object Location and Routing for Large-Scale Peer-to-Peer Systems, Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms, 2001

11: Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan, Chord: A scalable peer-to-peer lookup service for internet applications, 2001

12: Albert-Laszlo Barabasi and Eric Bonabeau, Scale-Free Networks, Scientific American, 2003.

13: Subhabrata Sen and Jia Wong, Analyzing peer-to-peer traffic across large networks, Transactions on Networking, 2004

14: Nathaniel Leibowitz, Matei Ripeanu and Adam Wierzbicki, Deconstructing the Kazaa Network, Proceedings of the 3rd International Workshop on Internet Applications, 2003

15: J.A. Pouwelse, P. Garbacki, D.H.J. Epema, and H.J. Sips, The Bittorrent P2P File-sharing System: Measurements and Analysis, 2005

16: Sean Rhea, Dennis Geels, Timothy Roscoe, and John Kubiatowicz, Handling Churn in a DHT, Proceedings of the USENIX Annual Technical Conference, 2004

17: Beverly Yang and Hector Garcia-Molina, Designing a Super-Peer Network, Proceedings of the 19th International Conference on Data Engineering, 2003

18: Alper Tugay Mizrak, Yuchung Cheng, Vineet Kumar and Stefan Savage, Structured Superpeers: Leveraging Heterogeneity to Provide Constant-Time Lookup, Proceedings of the 3rd IEEE Workshop on Internet Applications, 2003

19: Jan Sacha and Jim Dowling, A Gradient Topology for Master-Slave Replication in P2P Environments, In Post-Proceedings of the 3rd International Workshop on Databases, Information Systems and Peer-to-Peer Computing, 2005

20: M. Jelasity and O. Babaoglu, T-Man: Gossip-based overlay topology management, The 3rd International Workshop on Engineering Self-Organising Applications, 2005

21: J. Liang, R. Kumar, and K. Ross, The KaZaA Overlay: A Measurement Study, 2004

22: SA. Baset and H. Schulzrinne, An Analysis of the Skype Peer-to-Peer Internet Telephony Protocol, 2004

23: E. Cohen and S. Shenker, Replication strategies in unstructured peer-to-peer networks, ACM SIGCOMM Computer Communication, 2002

24: Q. Lv, P. Cao, E. Cohen, K. Li and S. Shenker, Search and replication in unstructured peer-to-peer networks, Proceedings of the 16th International Conference on Supercomputing, 2002

25: Ian Clarke, Oskar Sandberg, Brandon Wiley and Theodore W. Hong, Freenet: A Distributed Anonymous Information Storage and Retrieval System, Proceedings of the International Workshop on Designing Privacy Enhancing Technologies, 2000

26: A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, D. Terry, Epidemic algorithms for replicated database maintenance, 6th Annual Symposium on Principles of Distributed Computing, 1987

27: A. Datta, M. Hauswirth, and K. Aberer, Updates in Highly Unreliable, Replicated Peer-to-Peer Systems, 23rd International Conference on Distributed Computing Systems, 2003

28: Andy Tanenbaum and Maarten van Steen, Distributed Systems: Principles and Paradigms, 2002

29: Beverly Yang and Hector Garcia-Molina, Improving Search in Peer-to-Peer Networks, Proceedings of the 22nd International Conference on Distributed Computing Systems, 2003

30: Dimitrios Tsoumakos and Nick Roussopuoulos, A Comparison of Peer-to-Peer Search Methods, 2003

31: Eoin Curran and Jim Dowling, SAMPLE: Statistical Network Link Modeling in an On-Demand Probabilistic Routing Protocol for Ad Hoc Networks, Wireless on Demand Network Systems and Services, 2005

32: Burton H. Bloom, Space/time trade-offs in hash coding with allowable errors, 1970

33: Jinyang Li, Boon Thau Loo, Joseph M. Hellerstein, M. Frans Kaashoek, David R. Karger and Robert Morris, On the Feasibility of Peer-to-Peer Web Indexing and Searching, 2003

34: Patrick Reynolds and Amin Vahdat, Efficient Peer-to-Peer Keyword Searching, Middleware 2003

35: J. Kubiatowicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, C. Wells, and B. Zhao, OceanStore: An Architecture for Global-Scale Persistent Storage, Proceedings of the 9th international Conference on Architectural Support for Programming Languages and Operating Systems, 2000

36: S. Ratnasamy, P. Francis, M. Handley, R. Karp and S. Schenker, A scalable content-addressable network, In Proceedings of ACM SIGCOMM 2001

37: Antony I. T. Rowstron and Peter Druschel, Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems, Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms, 2001

38: Erik Christensen, Francisco Curbera, Greg Meredith and Sanjiva Weerawarana, 2001

39: Frank Dabek, M. Frans Kaashoek, David Karger, Robert Morris, and Ion Stoica, Wide-area cooperative storage with CFS, ACM Symposium on Operating Systems Principles, 2001

40: R. Housley, W. Ford, W. Polk, and D. Solo, Internet X.509 Public Key Infrastructure Certificate and CRL Profile, Request for Comments (RFC) 2459, 1999

41: http://en.wikipedia.org/wiki/PKCS, Public Key Cryptography Standards, Wikipedia, 2005

42: J. Basney, M. Humphrey, and V. Welch, The MyProxy Online Credential Repository, Software: Practice and Experience, 2005

43: S. Tuecke, V. Welch, D. Engert, L. Pearlman, and M. Thompson, Internet X.509 Public Key Infrastructure (PKI) Proxy Certificate Profile, Request for Comments (RFC) 3820, 2004

44: http://www.cacert.org/, CAcert, CAcert, 2005

45: http://www.verisign.com/repository/netsure_faq/index.html, NetSure® Protection Plan, Frequently Asked Questions, VeriSign, 2004

46: http://www.verisign.com/products-services/payment-processing/online-payment/fraud/index.html, Fraud Protection Services, VeriSign, 2005

47: G. Gordon, K. Glushkova, and S. Elaluf-Calderwood, Regulatory Framework, DBE Knowledge Base of Regulatory Issues, 2005

48: R. E. Smith, Authentication: from passwords to public keys, 2001

49: M. Castro, P. Druschel, A. Ganesh, A. Rowstron, and D. S. Wallach, Secure routing for structured peer-to-peer overlay networks, 2002

50: J. R. Douceur, The Sybil Attack, 2002

51: J. Dowling et al., Collaborative Reinforcement Learning of Autonomic Behaviour

52: D. Dahlem et. al, DBE First Implementation, EU Deliverable, April 2005