Open Philosophies for Associative Autopoietic Digital Ecosystems

Contract n° 034824

Workpackage 3: Autopoietic P2P networks

Deliverable D3.1: Preliminary architecture for P2P network focusing on hierarchical virtual super-peers, birth and growth models

Project funded by the European Community under the “Information Society Technology” Programme

Contract Number: 034824

Project Acronym: OPAALS

Title: Open Philosophies for Associative Autopoietic Digital Ecosystems

Deliverable N°: D3.1

Due dates: 12/2006

Delivery Date: 06/2007

Short Description:

Workpackage 3 is focused on the development of autopoietic P2P networks. This document reports on the preliminary p2P architecture for OPAALS. In order to clarify the services that such an architecture needs to support, however, we do need to first provide details of the transaction model for open collaborations of SMEs. Consequently, this report starts first with a detailed description of our model for long-term distributed transactions. The report then continues with a review of existing P2P architectures, identifying their weaknesses with regard to support for open communities of collaborating enterprises in a digital ecosystem. The report then introduces and describes our p2p architecture, its birth and growth models, and mechanisms for prevention of fragmentation in the network.

Author: UniS

Partners contributed:

Made available to: Public

VERSIONING		
VERSION	DATE	AUTHOR, ORGANISATION
0.1	**12/2006**	**AMIR RAZAVI, SOTIRIS MOSCHOYIANNIS, PAUL KRAUSE (UNIS)**
0.9	**02/2007**	**AMIR RAZAVI, SOTIRIS MOSCHOYIANNIS, PAUL KRAUSE (UNIS)**
1.0	**03/2007**	**AMIR RAZAVI, SOTIRIS MOSCHOYIANNIS, PAUL KRAUSE (UNIS)**
2.0	**04/2007**	**AMIR RAZAVI, SOTIRIS MOSCHOYIANNIS, PAUL KRAUSE (UNIS)**
2.1	**05/2007**	**AMIR RAZAVI, SOTIRIS MOSCHOYIANNIS, PAUL KRAUSE (UNIS)**

Quality check:

1st Internal Reviewer : Miguel Vidal (TI)

2nd Internal Reviewer: Jésus Gadalbon (TI)

Table of Contents

1 Introduction

Work package 3 addresses Objective 6 of the OPAALS Network of Excellence. That is, it has a key target of developing Peer to Peer (P2P) architectures to support "distributed long-term transactions, accountability, identity and trust". However, this statement is not sufficient to guide the tasks in WP3. It is important to also consider the context within which this objective was identified. In particular, the P2P architecture(s) must enable open and trusted collaborations between small-to-medium enterprises (SMEs), to ensure their sustainability within a pan-European (and ultimately, global) Digital Ecosystem.

Our research hypothesis is that in order to do this, we must move away from "traditional" centralised solutions for P2P architectures and transactional modelling, to fully distributed solutions. This report provides initial statements for both the proposed transaction model, and the P2P architecture that supports the transaction model. We also review existing work in both areas, and highlight their weaknesses with regard to meeting the OPAALS objectives. The overriding weakness is that existing approaches rely on a centralised, or limited decentralised architecture that threatens the local autonomy of participating SMEs and leaves them vulnerable to governance from a single large organisation.

In keeping with the spirit of OPAALS our P2P architecture proposals derive much of their inspiration from studies of networks of interaction and collaboration in the social and natural sciences. In particular the P2P network model can be viewed as an example of a birth (node duplication with divergence), death (node inactivation or removal), and innovation (horizontal "gene" transfer) model (BDIM). What we are aiming for here, is the development of a digital environment that evolves naturally to support a sustainable economy, where interactions through e-commerce are an integral part of that economy. In a sense the e-infrastructure will become "invisible", or non-intrusive, with regard to the continued development of a sustainable economy that resides within that infrastructure. So, we would expect to see that economy, and its associated social interactions, continue to evolve in a way that is not constrained by the properties of the underlying digital ecosystem.

In order to clarify the fundamental properties that we expect to see of a digital ecosystem, Chapter 2 provides a short introduction to biodiversity that emphasises the importance of a pool of rare species in sustaining (innovation in) an ecosystem. In a digital ecosystem, for "rare species" read SMEs. At the moment our claims for the relevance of this work to DBEs is drawn by analogy only, and so needs to be treated with caution at this stage of the project.

We then describe our model for long-term distributed transactions in Chapter 3. We start by a discussion on the nature of transactions in the context of a digital ecosystem for business and give a brief overview of existing popular transaction models which have been designed with web services in mind. Apart from the fact that current designs consider a centralised (and recently, a limited decentralised) model, we highlight their shortcomings with regard to supporting multi-service long-running transactions in a distributed manner. The proposed distributed transaction model for OPAALS supports the design of long-running transactions whose underlying services can be composed in a variety of ways. These are coordinated in a fully distributed manner, based on the log structures provided by local consistency graphs. In addition, the obligation for cooperation between transactions can be specified in a business process rule (a requirement for availability of "partial results"). Finally, the instability of the internet environment can define a new requirement for keeping important results even when the connection between two platforms is lost ("omitted results") or providing alternative execution scenarios so that a transaction can be completed even when some failure occurs ("forward recovery").

Chapter 4 reviews existing P2P architectures and examines whether they meet the requirements of a digital ecosystem for business. These last two chapters provide important material to motivate the development of the P2P architecture which is proposed in Chapter 5. A prerequisite for a network supporting an open community of SMEs within a digital ecosystem is that the design is fully distributed.

The highly dynamic nature of the transactional environment implies that 'nodes' may enter or leave the network at will at any time. This requires that the underlying network evolves in a way that adopts to change dynamically. It is thus necessary to consider a birth and growth model for the *autopoietic* P2P network in OPAALS that ensures the ever changing network topology reflects the activity of the participating entities and adopts in a way that perpetuates the characteristics of the network that bring stability and facilitate the sustainability of the participants.

Finally, we outline the formal modelling tasks that are underway to refine the P2P architecture in Chapter 6. The results of the formal modelling will be reported in Deliverable D3.2.

2 Biodiversity: Networks of Interaction

A fundamental hypothesis amongst environmentalists is that diversity in natural ecosystems is a fundamental indication of "health" or long-term stability of ecosystems. The empirical support for this is, of course, taken from the extreme ends of the scale of diversity. Dependence on extensive monocultures (potatoes in Ireland, wheat in Northern America) has led to catastrophic failures of the associated ecosystems. In contrast, areas of richest biodiversity (tropical rainforests, marine ecosystems) have remained stable for possibly thousands or tens of thousands of years until the recent large-scale human interventions. Perhaps an important comment to add to that is that the human civilisations that evolved within the rain-forest ecosystems were and are important components of those ecosystems. The sustained damage typically comes from large-scale interventions from human economic groups that have evolved outside of, and that are hence unsympathetic to the long-term survival of, the respective ecosystems.

The challenge for environmental policy makers comes in identifying measures of diversity that can provide meaningful measures of risk for ecosystems that are intermediate between these two extremes. Can we identify a measure that will discriminate between an ecosystem where human economic activity has degraded biodiversity, yet that level of biodiversity is still sufficient to ensure the long-term survival of its essential qualities, and one where the "natural" element of the ecosystem is completely dominated by human activity?

Currently, there is no single measure that is a sufficient statistic for biodiversity. We deliberately use the term "sufficient statistic" to emphasise that the fundamental reason for this is that there is no agreed model of diversity from which to derive such a statistic. In this section, we will look at some existing proposals for measures of diversity. We will then look at distributions of relative abundance for species in a range of ecosystems to provide an empirical analysis of such distributions. Finally, we will review some of the current evidence for how the global ecosystem responded to significant stress during the break up of the Pangea super-continent.

Figure 2.1: (a) (b)

2.1 Measures of Diversity

We can use a simple "classic" example to illustrate the difficulty of identifying a simple measure of biodiversity. Consider the two example ecosystems in Figure 2.1 (a) and (b). In terms of a simple count of the number of different species, ecosystem (a) "wins". However, ecosystem (b) has a potential advantage in that the likelihood of picking two different species in a random sample is greater than for population (a). Although this is an extremely simple example, it does indicate that both numbers of species and relative abundances of species are important factors in determining relative "goodness" of ecosystems.

This is addressed to an extent in some of the more complex measures of diversity that are currently in use. Simpson's diversity index D (or strictly (1-D)), for example, represents the likelihood of two randomly selected individuals belonging to different species. Shannon's diversity index is based on information theory and provides something analogous to a measure of entropy, but it is very hard to estimate, and again an underlying thermodynamical model that would explain the significance of such a measure is not currently available.

Empirical and theoretical evidence is, however, for the form of statistical distribution that arise in naturally evolving ecosystems. Karev et al [KWR02] provide both empirical evidence for power law distributions in relative abundances of protein domains, and also provide a theoretical explanation for these distributions in terms of a relatively simple birth, death and innovation (BIDM) model. Currently, we are not able to report on the theoretical modeling for macroscopic ecosystems. However, in the next section we will provide a number of empirical species abundance curves to illustrate that this is a widely occurring phenomenon.

2.2 Example Species Abundance Curves

We are grateful to Dr Karl Ugland of the Marine Zoology Department, University of Oslo, and Dr David Roberts of the Herbarium, Royal Botanic Gardens, Kew, UK, for providing all the species abundance curves used in this section.

Figure 2-2 below provides an example species abundance curve for Benthic macrofauna on the Norwegian continental shelf. These include marine crustaceans, bivalves and small worms that are "large" in the sense that they will not pass through a 1mm sieve (in the European standard definition). As food for demersal ("flat") fish, and feeders on organic waste matter, they can be considered as representative of a single trophic level in a marine ecosystem. Indeed they are often used as an indicator of environmental quality. In figure 2.2 the x-axis is scaled by number of observed samples, and the y-axis by number of species with the corresponding sample count.

The nature of the species abundance distribution becomes clearer if we rescale using double log coordinates, as in Figure 2-3. Now the plot indicates that number of species C(x) that are found in exactly x samples follows a power law distribution: $C(x) = ci\text{-}\gamma$. With double log coordinates, the plot of $y = C(x)$ as a function of x yields a straight line with negative slope γ.

In Figure 2-3 and the subsequent figures, a straight line has been fitted to double log plots of species distributions for a wide range of sample ecosystems. There is a certain judgemental element here as we are currently unable to obtain complete samples of all orders in a complex ecosystem. Instead we have focused on members that are typically representative of a single trophic level within a given ecosystem.

It should be clear from all the species abundance curves that the power law distribution closely fits the observed distribution for all the samples we have been able to obtain so far. On scanning through the following figures, it is interesting to note that the slopes, γ, vary between approximately 0.7 and 1.6.

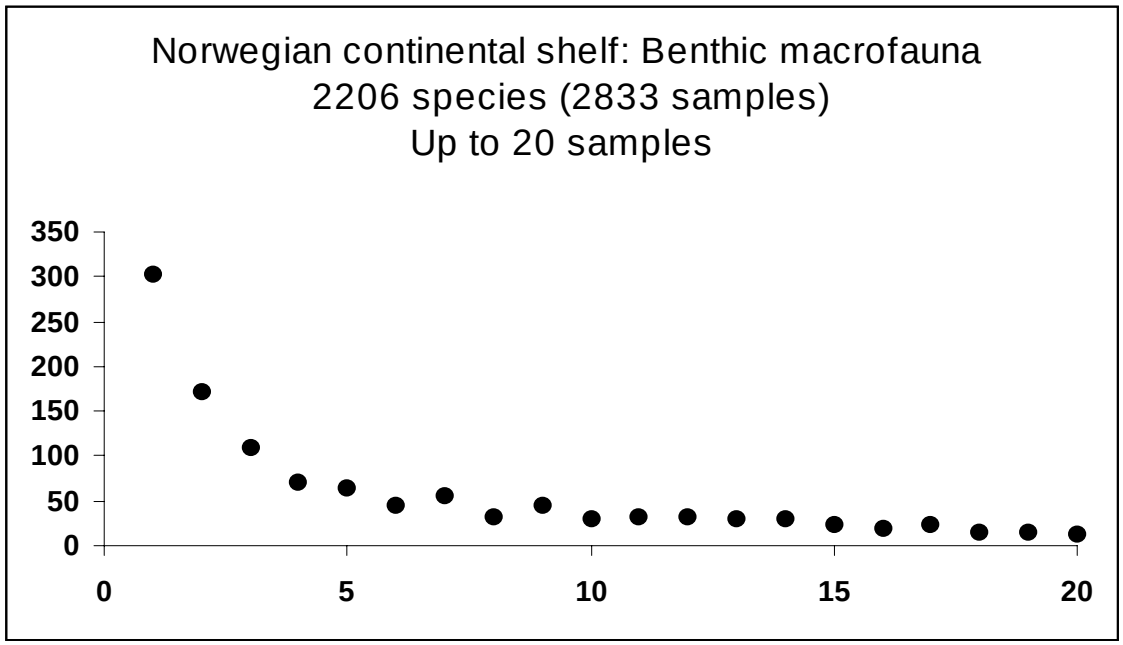


Figure 2-2

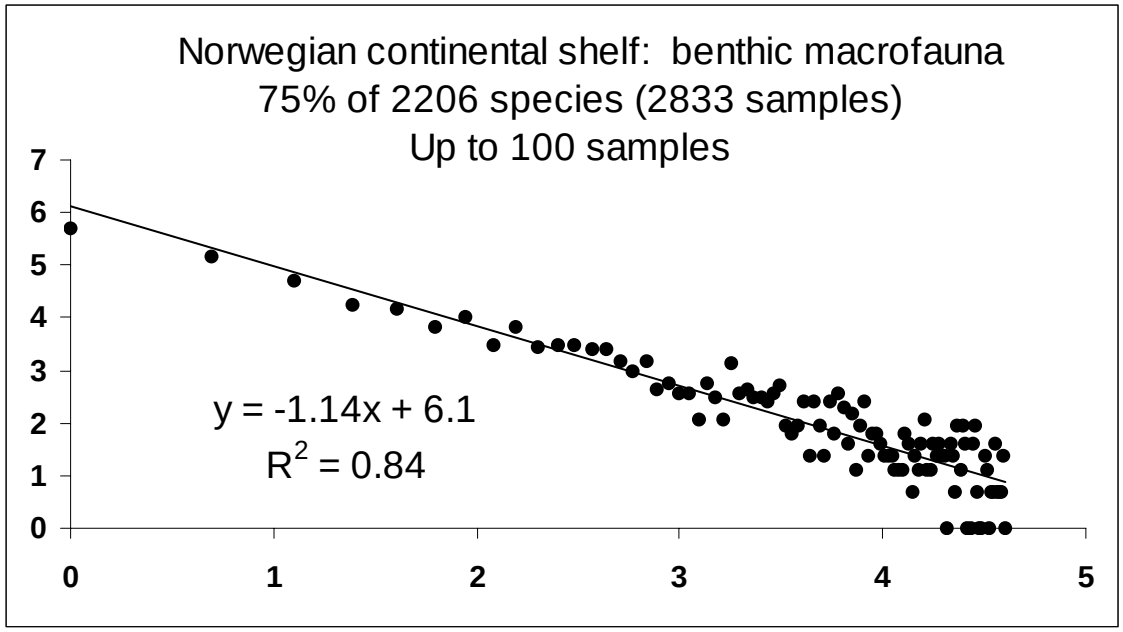
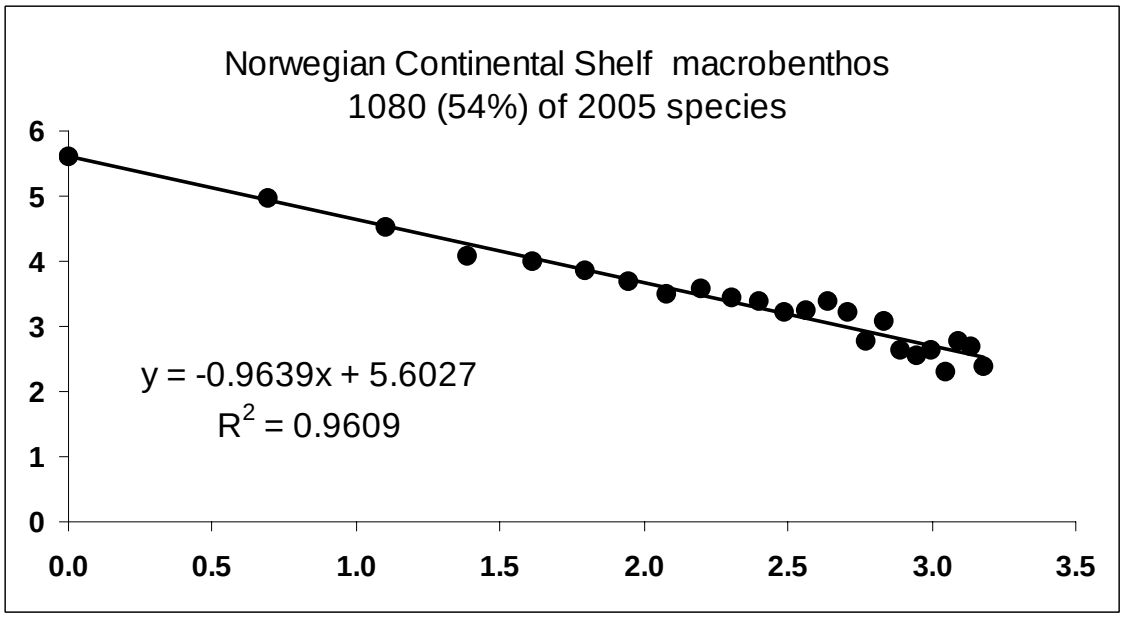
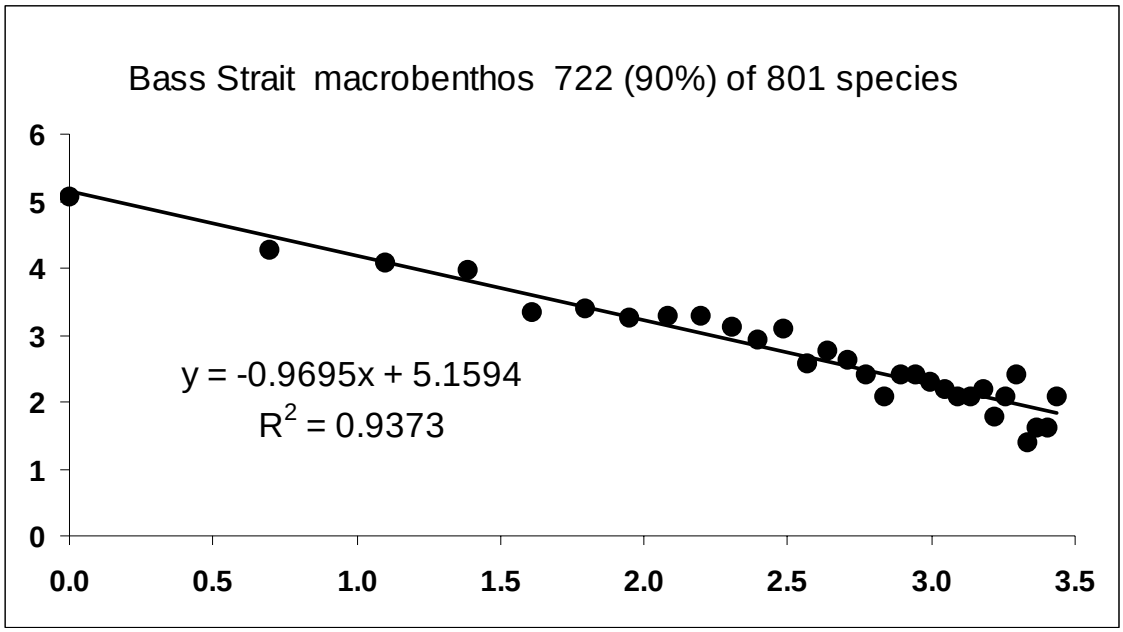
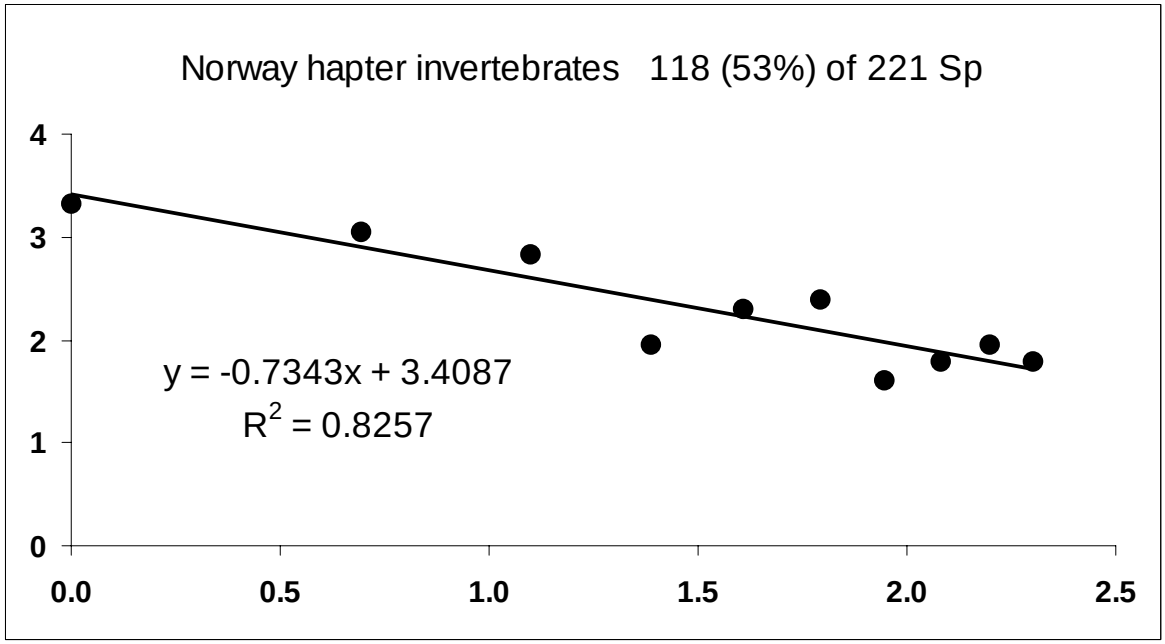
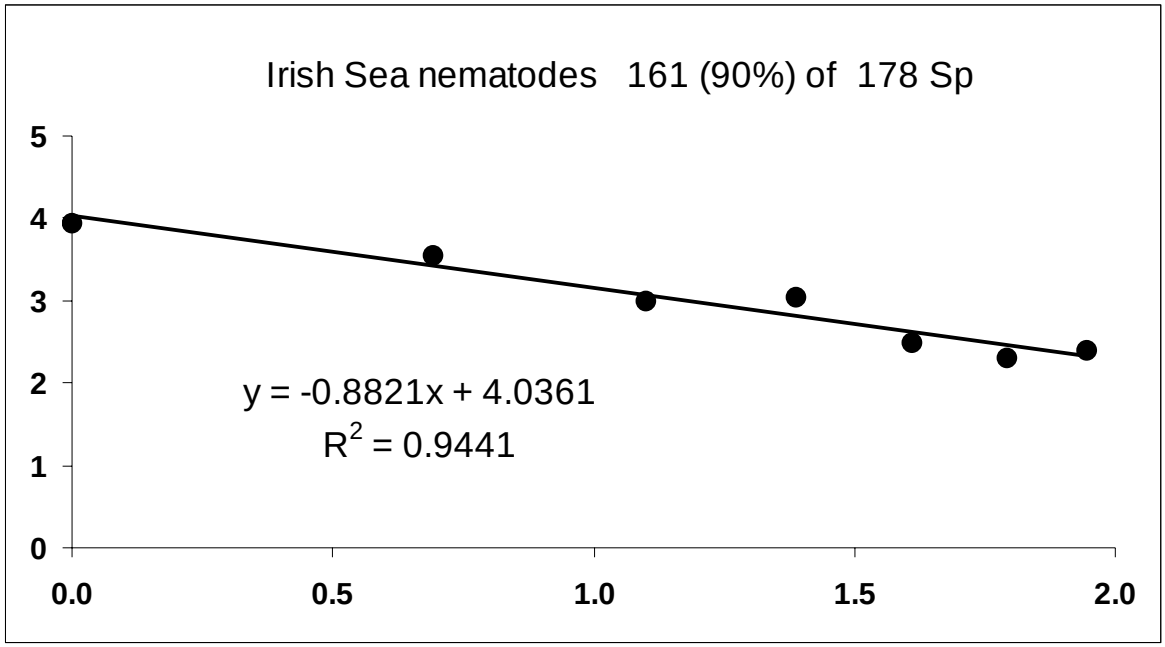


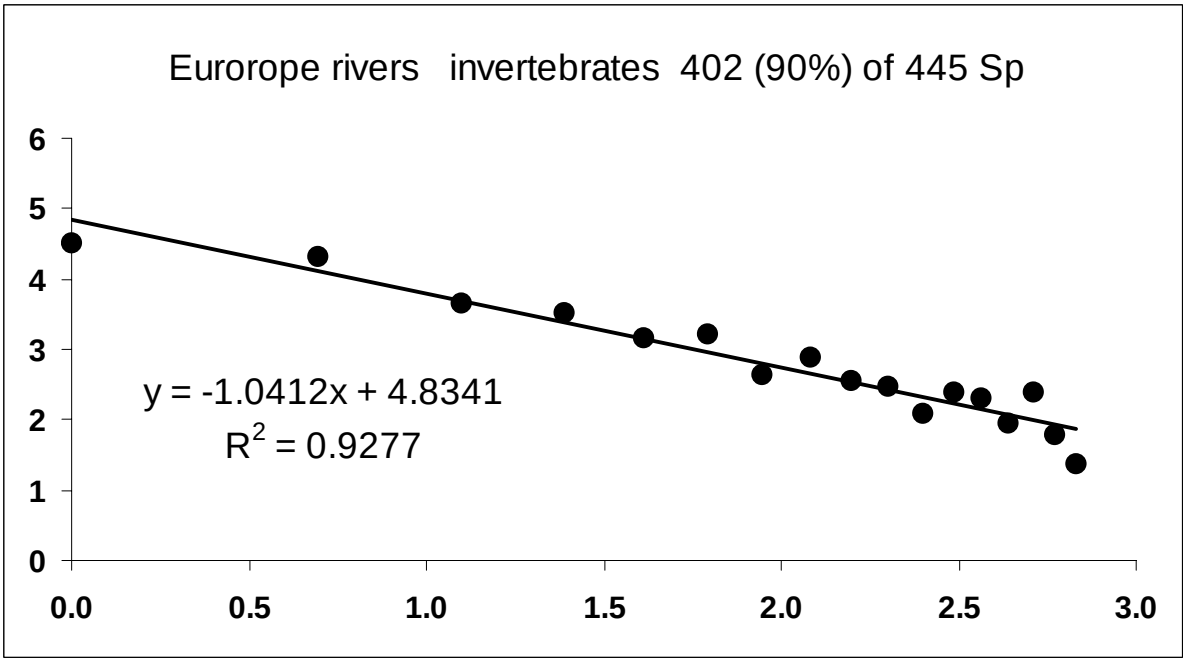
Figure 2-3

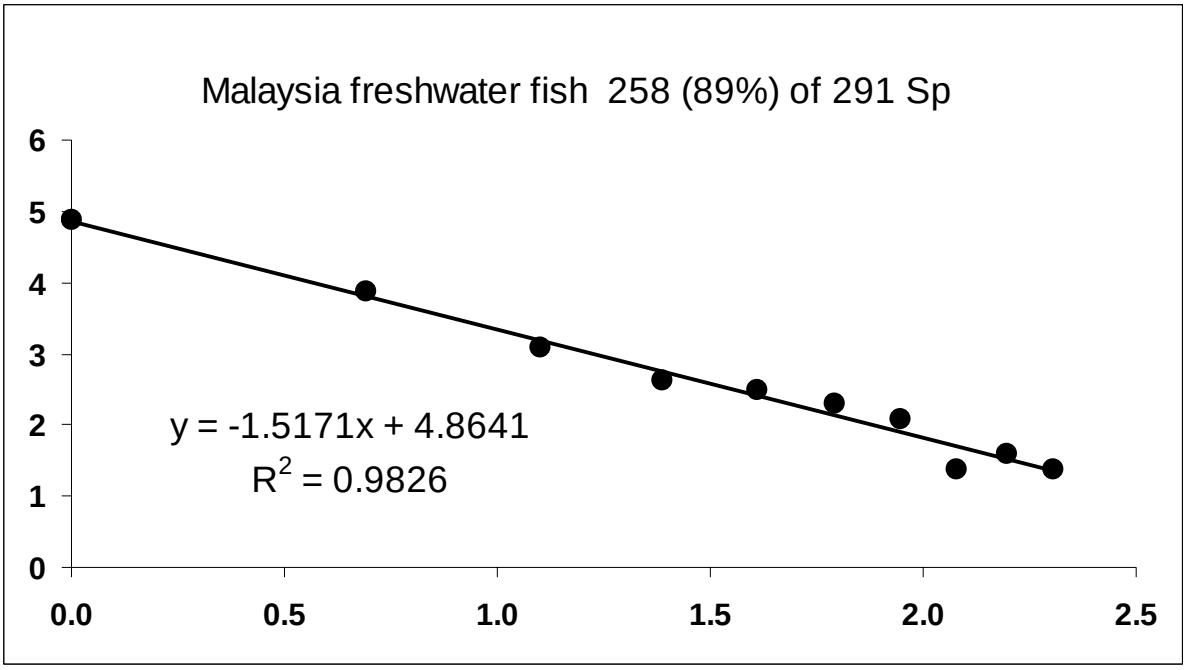

Norwegian Continental Shelf macrobenthos
1080 (54%) of 2005 species
y = -0.9639x + 5.6027
R2 = 0.9609
0
1
2
3
4
5
6
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5

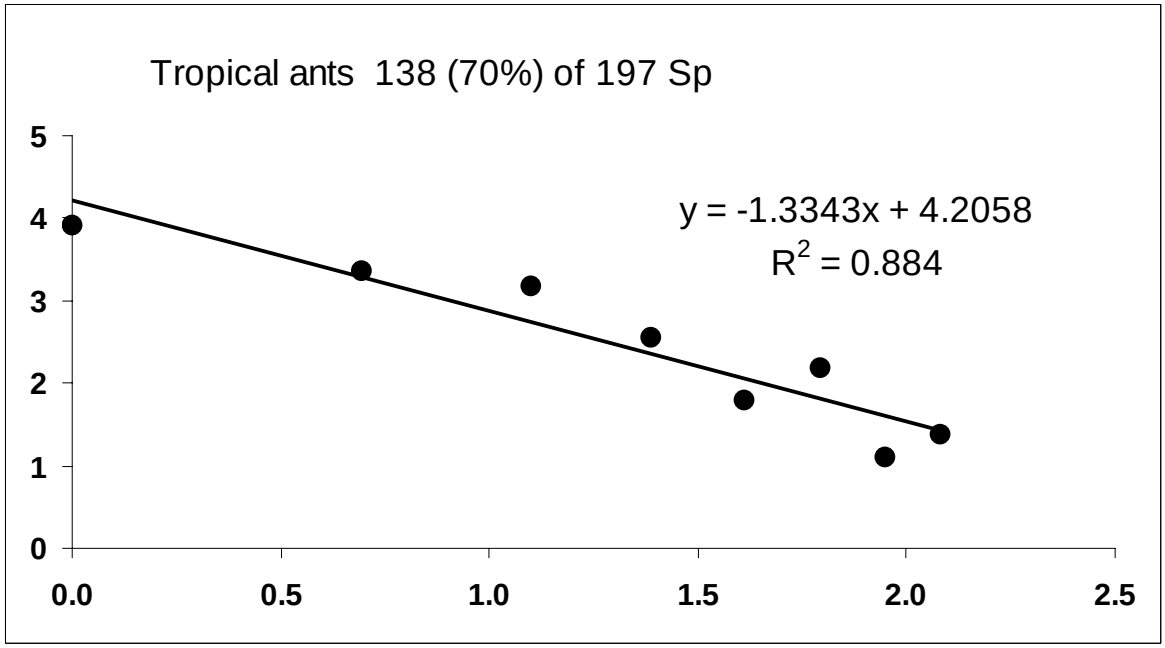

Bass Strait macrobenthos 722 (90%) of 801 species
y = -0.9695x + 5.1594
R2 = 0.9373
0
1
2
3
4
5
6
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5

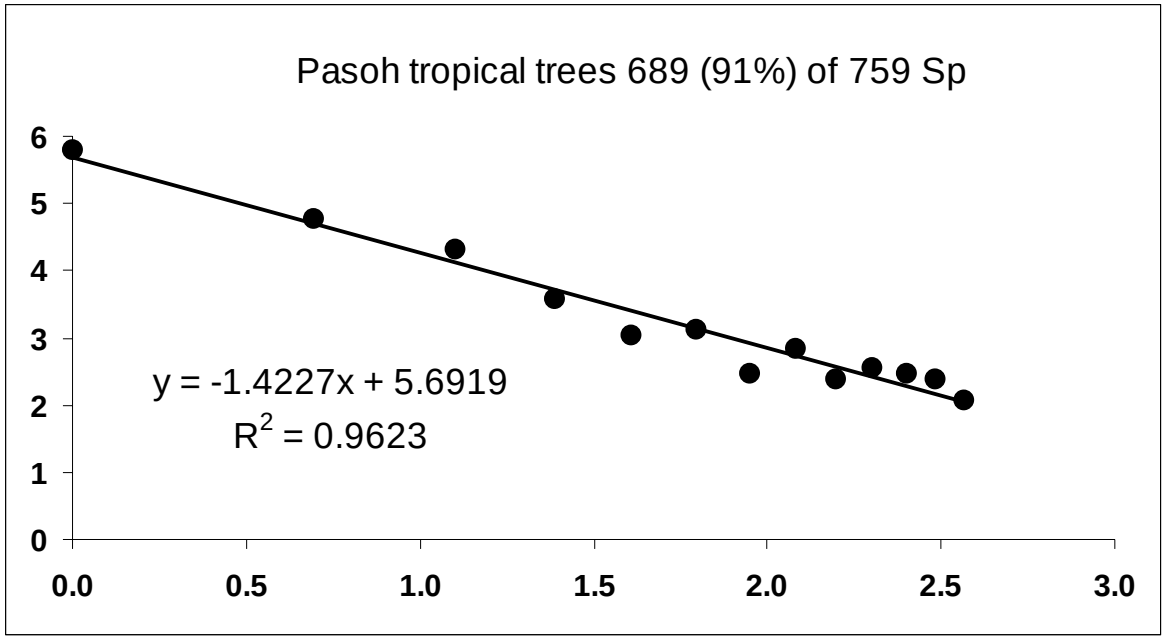

Norway hapter invertebrates 118 (53%) of 221 Sp
y = -0.7343x + 3.4087
R2 = 0.8257
0
1
2
3
4
0.0
0.5
1.0
1.5
2.0
2.5

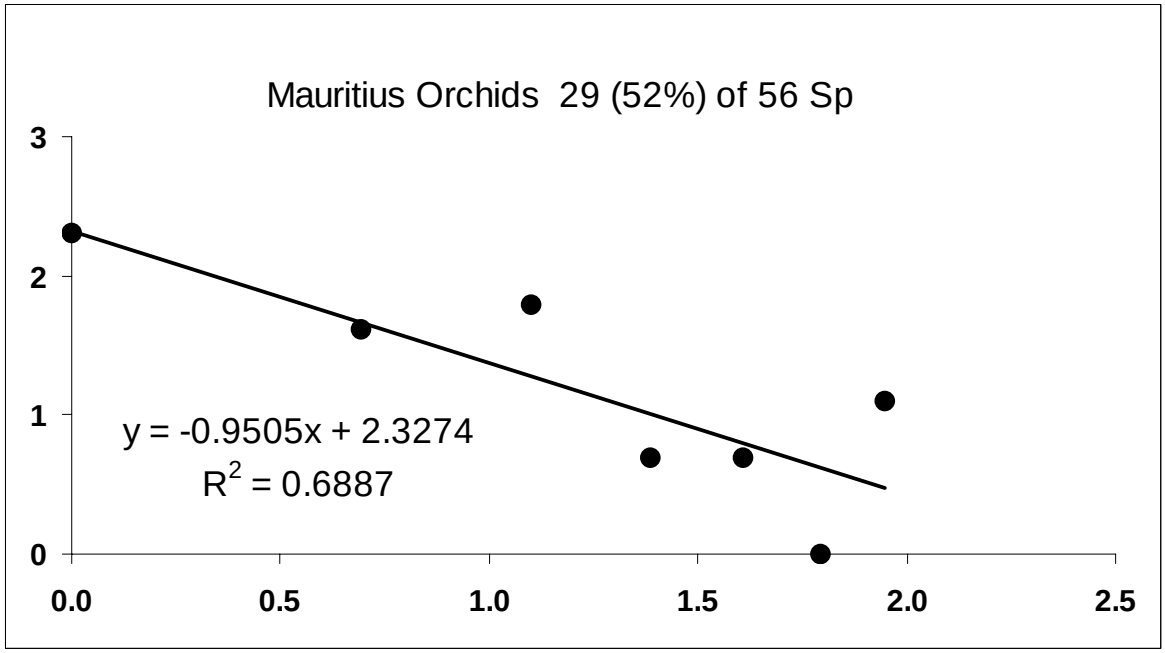

Irish Sea nematodes 161 (90%) of 178 Sp
y = -0.8821x + 4.0361
R2 = 0.9441
0
1
2
3
4
5
0.0
0.5
1.0
1.5
2.0

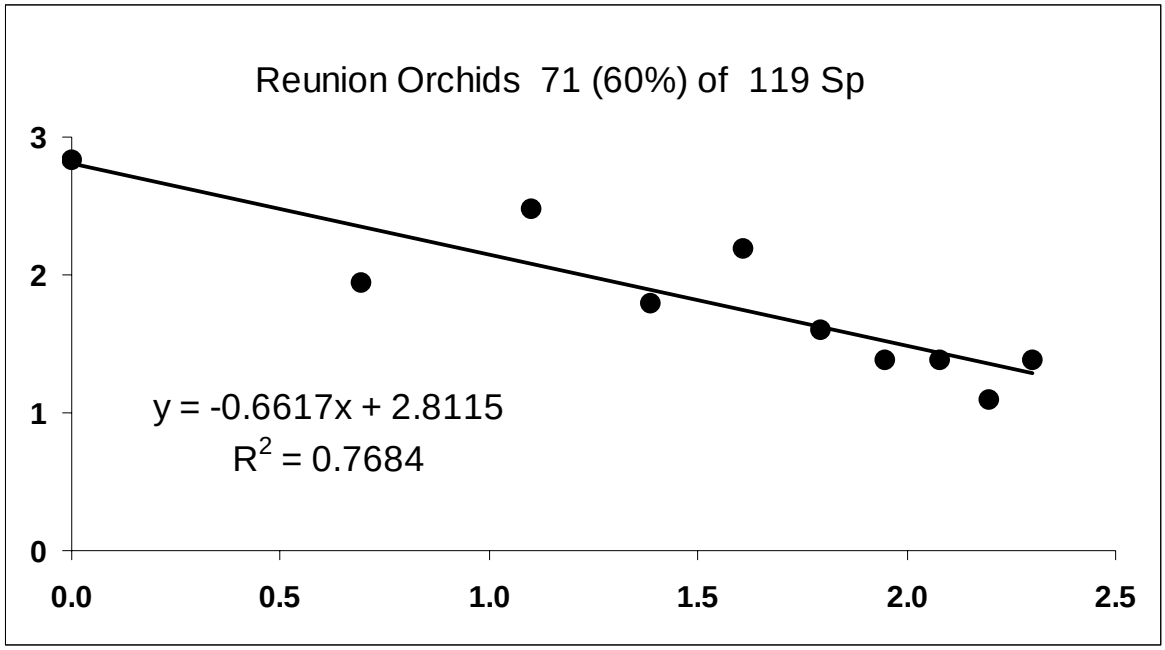

Eurorope rivers invertebrates 402 (90%) of 445 Sp
y = -1.0412x + 4.8341
R2 = 0.9277
0
1
2
3
4
5
6
0.0
0.5
1.0
1.5
2.0
2.5
3.0

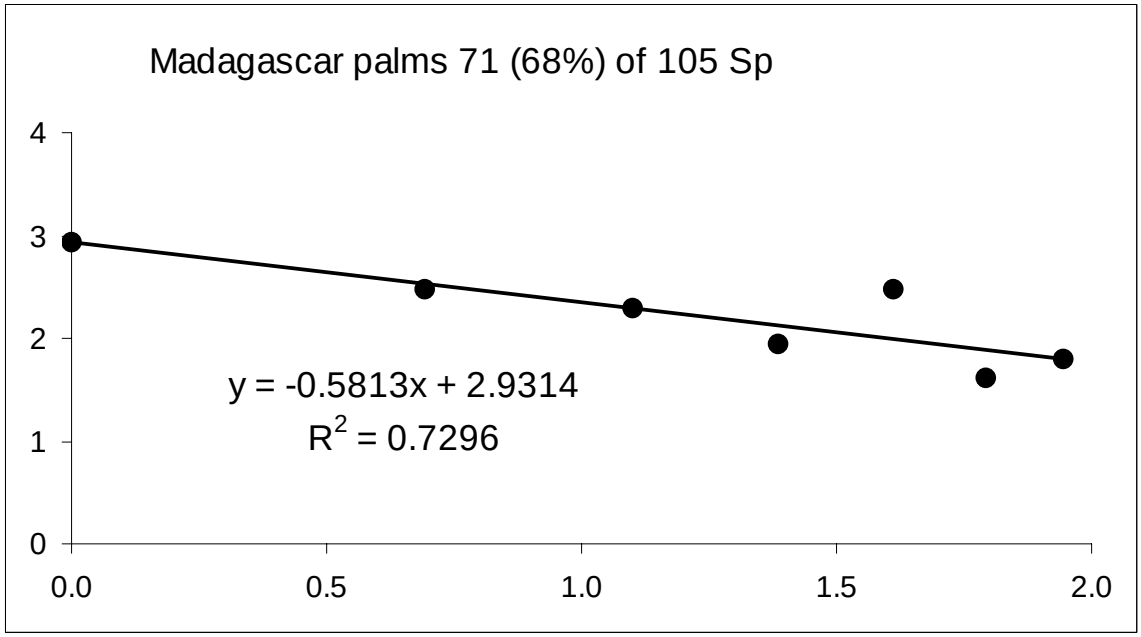

Malaysia freshwater fish 258 (89%) of 291 Sp
y = -1.5171x + 4.8641
R2 = 0.9826
0
1
2
3
4
5
6
0.0
0.5
1.0
1.5
2.0
2.5

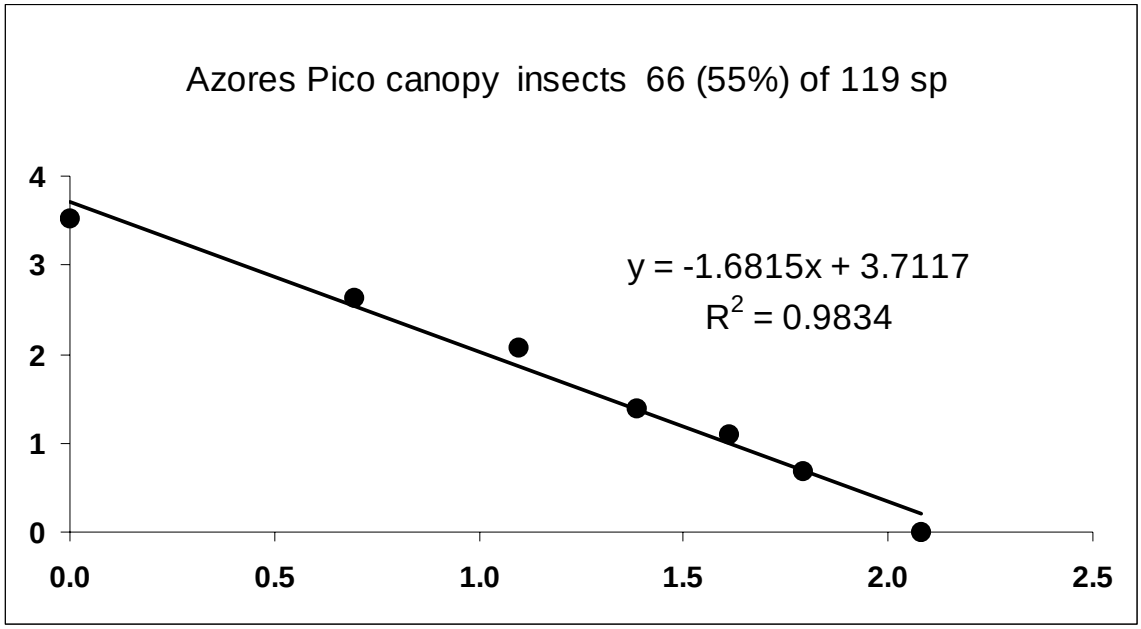

Tropical ants 138 (70%) of 197 Sp
y = -1.3343x + 4.2058
R2 = 0.884
0
1
2
3
4
5
0.0
0.5
1.0
1.5
2.0
2.5

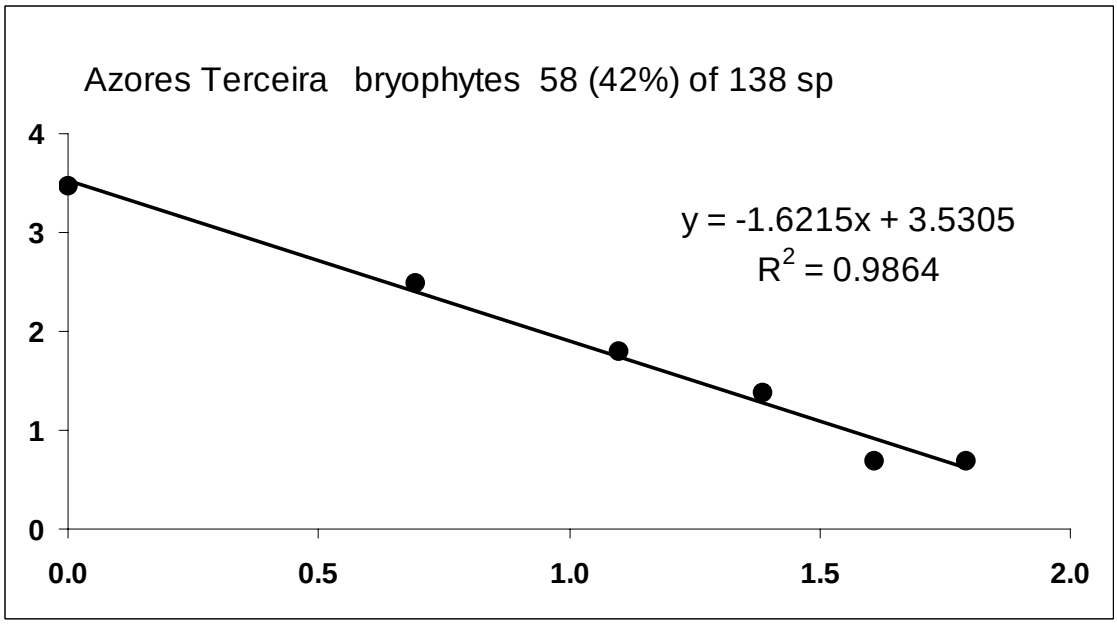

Pasoh tropical trees 689 (91%) of 759 Sp
y = -1.4227x + 5.6919
R2 = 0.9623
0
1
2
3
4
5
6
0.0
0.5
1.0
1.5
2.0
2.5
3.0

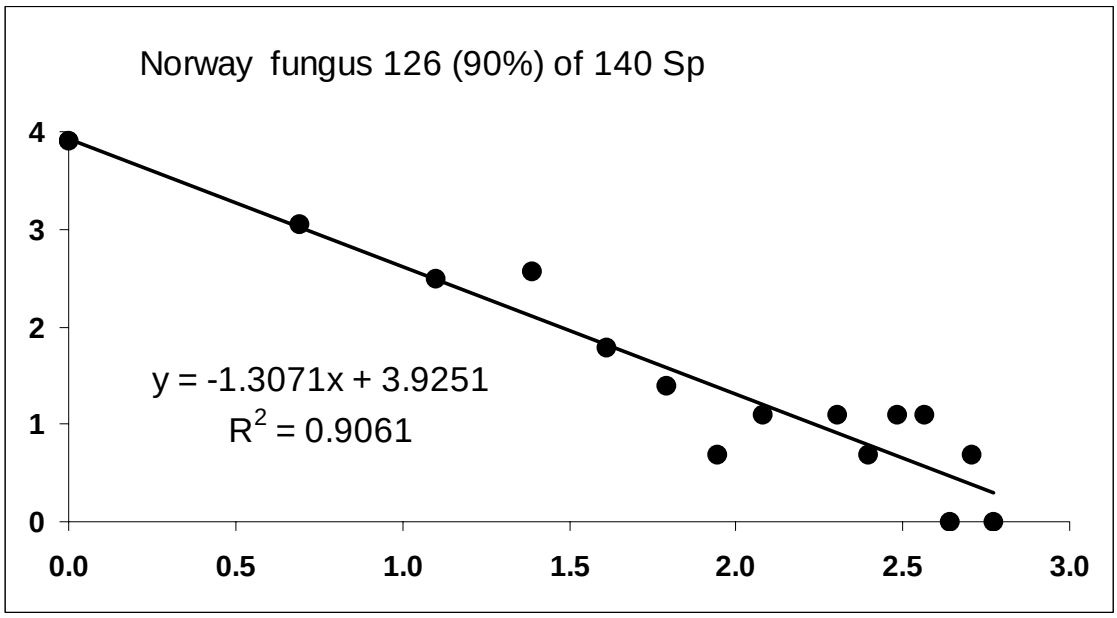

Mauritius Orchids 29 (52%) of 56 Sp
y = -0.9505x + 2.3274
R2 = 0.6887
3
2
1
0
0.0
0.5
1.0
1.5
2.0
2.5


Reunion Orchids 71 (60%) of 119 Sp
y = -0.6617x + 2.8115
R2 = 0.7684
3
2
1
0
0.0
0.5
1.0
1.5
2.0
2.5


Madagascar palms 71 (68%) of 105 Sp
y = -0.5813x + 2.9314
R2 = 0.7296
0
1
2
3
4
0.0
0.5
1.0
1.5
2.0


Azores Pico canopy insects 66 (55%) of 119 sp
y = -1.6815x + 3.7117
R2 = 0.9834
0
1
2
3
4
0.0
0.5
1.0
1.5
2.0
2.5


Azores Terceira bryophytes 58 (42%) of 138 sp
y = -1.6215x + 3.5305
R2 = 0.9864
4
3
2
1
0
0.0
0.5
1.0
1.5
2.0


Norway fungus 126 (90%) of 140 Sp
y = -1.3071x + 3.9251
R2 = 0.9061
4
3
2
1
0
0.0
0.5
1.0
1.5
2.0
2.5
3.0

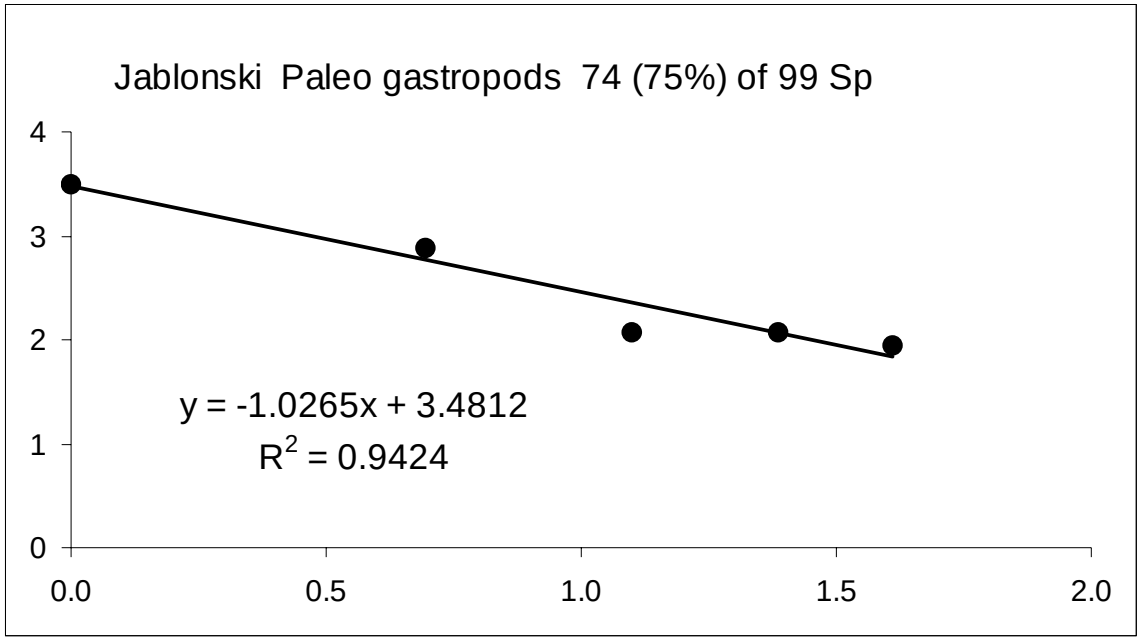


Figure 2.18

As mentioned earlier, Karev et al [KWR02] provide a simple explanation of power law behaviour in protein domains using a Birth, Death and Innovation Model (BDIM). Currently we are still researching the possible extension of this model to the above data. In the meantime one can propose some empirical heuristics from the data that has been collected so far. The long tail of rare species seems to be a fundamental characteristic of any stable ecosystem. Generally, we find that the communities with higher value for γ are from those ecosystems that have been less impacted by human intervention than those with a lower value of γ. The natural habitat of Mauritius, for example, is now highly fragmented. In addition, for the more degraded communities, we tend to see lower R2 values, indicating a loss in stability or degree of equilibrium in the associated ecosystem.

Further theoretical work is needed to justify the above statements. However, we can look at emerging fossil and geological evidence to see the dynamic importance of the long tail as a pool of innovation. In the next section we will see how evidence is emerging that significant long-term changes in one particularly important habitat, led to the extinction of one dominant group of species and the stimulation of a series of speciation events from this "pool of innovation" to fill the newly available niches. Just as a reminder, our interest in this is in terms of providing justification for the importance of SMEs in providing an analogous pool of open innovation in digital ecosystems.

2.3 The K-T Extinction and Speciation Events

The non-avian Dinosaurs and certain marine creatures were subject to a massive extinction event at the Cretaceous-Tertiary (K-T) boundary, about 65 Million Years ago. The current preferred theory seems to be that this was triggered by a massive meteorite impact at Chicxulub in what is now North-east Mexico. There is little doubt that this meteorite impact did cause widespread disruption to the global environment. However, it is less clear that this was the sole "impactor" on the inhabitants of the earth. Even more importantly, evidence is emerging that this did not even coincide with a period in which there was a net loss of biodiversity on the planet. Rather, the reductions in certain populations were being countered by ongoing speciation activities in other groups of species.

The first point to note is that the meteor impact was not the only major event that was effecting the physical environment of the time. In addition, the K-T boundary of 65 million years ago marked the final stages of the break up of the supercontinent Pangea into the broad continental structure that is now seen.

This was a gradual process that started some 225 million years ago. The period from 135 million years ago (Jurassic Park time!) to 65 million years ago saw the separation of the plates that form North America and Eurasia, and the plates that form South America, Antartica and Australasia. As well as this fragmentation of what were formerly massive continental areas, the continuing plate movements were beginning to form many of the topological features (mountains and high altitude plateaus) that are now homes to large pools of endemic species.

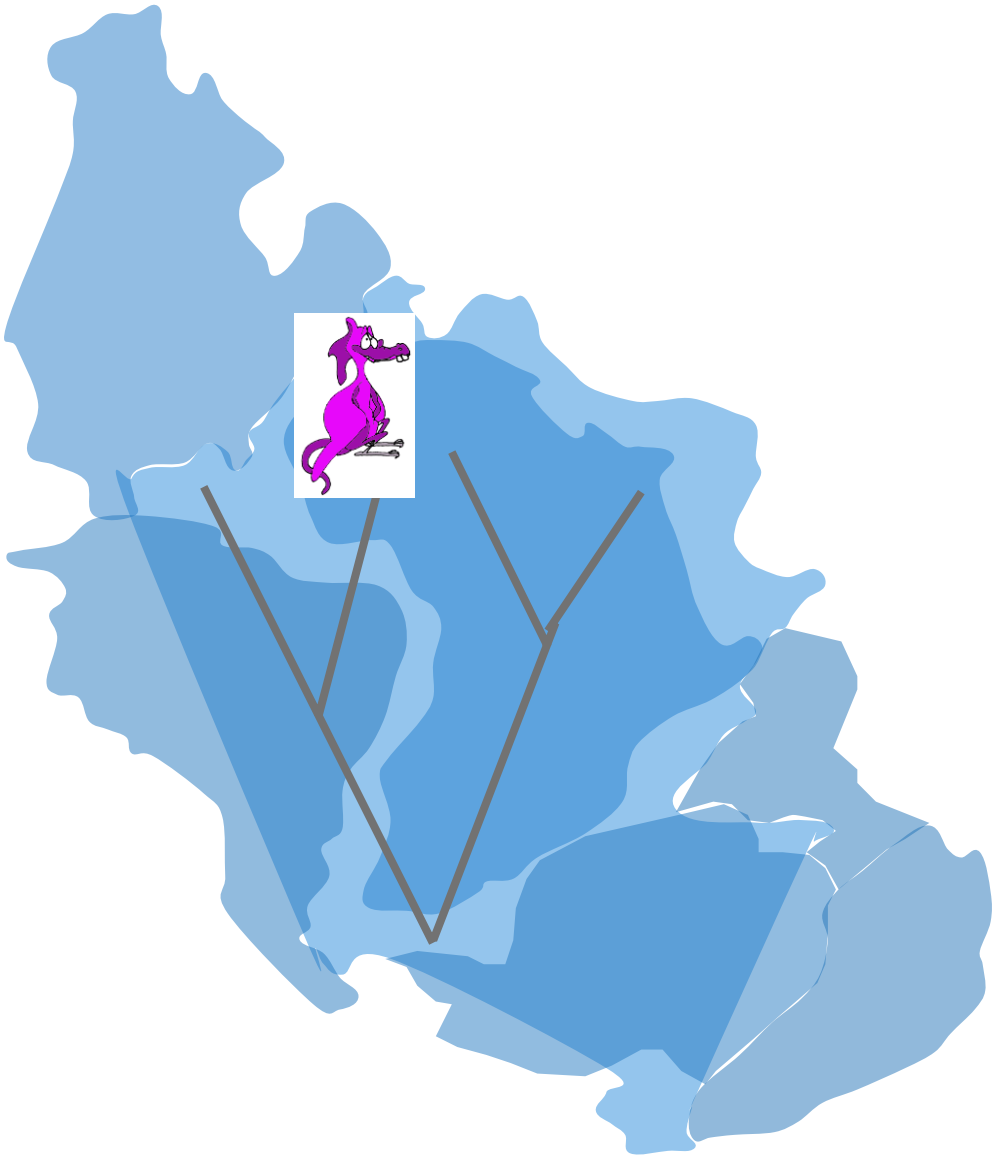


Figure 2.19: The dinosaurs emerged in a period when the Earth's land mass was joined as a single "supercontinent".

As an example, the huge sauropod footprints that were discovered in Croatia by a team led by Michael Caldwell of the University of Alberta (but published by Mezga et al, *Cretac. Res.* **27**, 735-742, 2006 – such is the cut-throat world of scientific research!) were laid down about 95 million years ago. As well as the important paleontological significance, these also represent the most recent signs of life on what is known as the Adriatic-Dinaric platform before is sunk below sea level 94 million years ago. This is clear evidence of niche removal for certain dinosaur species in one specific area.

Simultaneous to this niche destruction process that appears to have provided an adverse selection pressure on non-avian dinosaurs, recent evidence indicates that there was on going speciation amongst both mammals [BE07] and the avian fauna [BPM07]. Bininda-Emonds *et al*, [BE07] have constructed a near

complete species phylogeny for the extant mammals. This includes estimates of divergence times using a combination of gene alignment techniques and fossil calibration points. A key result was their identification of two periods of diversification. The first was from 100-85 Million years ago. Again note that this coincide with the final stages of the break up of the supercontinental land masses, although at the moment this is only a statement of coincidence and needs further theoretical investigation. The second appeared in the early Eocene, yet at approximately 50 millions years ago this was significantly *after* the K-T boundary. A key conclusion of [BE07] is that this "challenges the hypothesis that the end-Cretaceous mass extinction event had a major influence on the diversification of today's mammals".

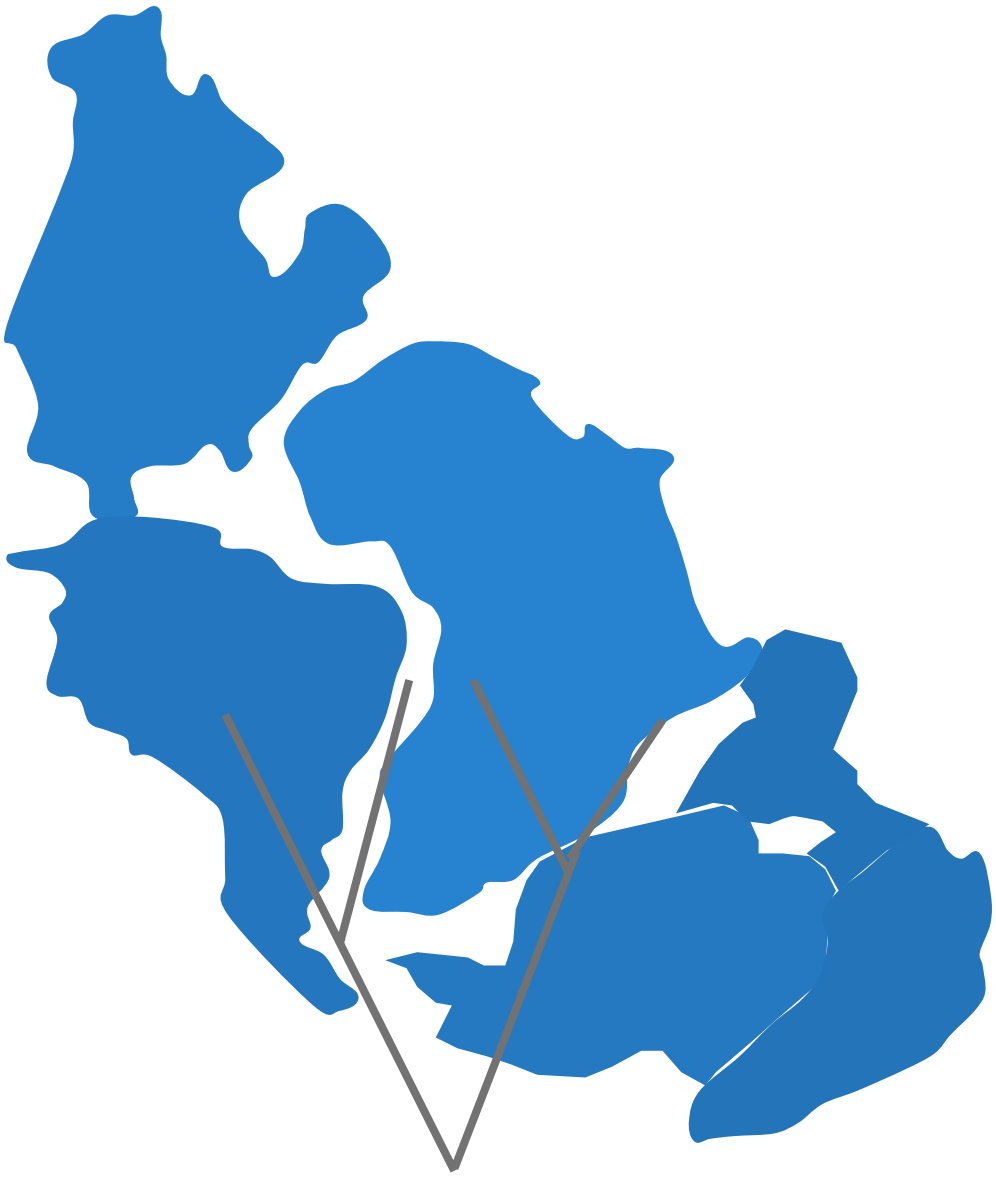


Figure 2.20: The later stages of the breakup of Pangea coincided with a major speciation event as well as the decline in dinosaur populations

Our proposal is that the K-T extinction event was a (final) perturbation in a long-term species innovation process that was driven by the continued break up and change to the topologies of the supercontinental land masses. Overall, this led to an acceleration in the Birth Death Innovation Model that underpins the evolution of biodiversity on this planet. Finally, and most importantly in terms of its relevance to our claims about the importance of support for SMEs in a DBE, the long tail of rare species played a critical role as a

pool of innovation during this adaptation of the natural ecosystem in response to this long-term period of niche creation and destruction.

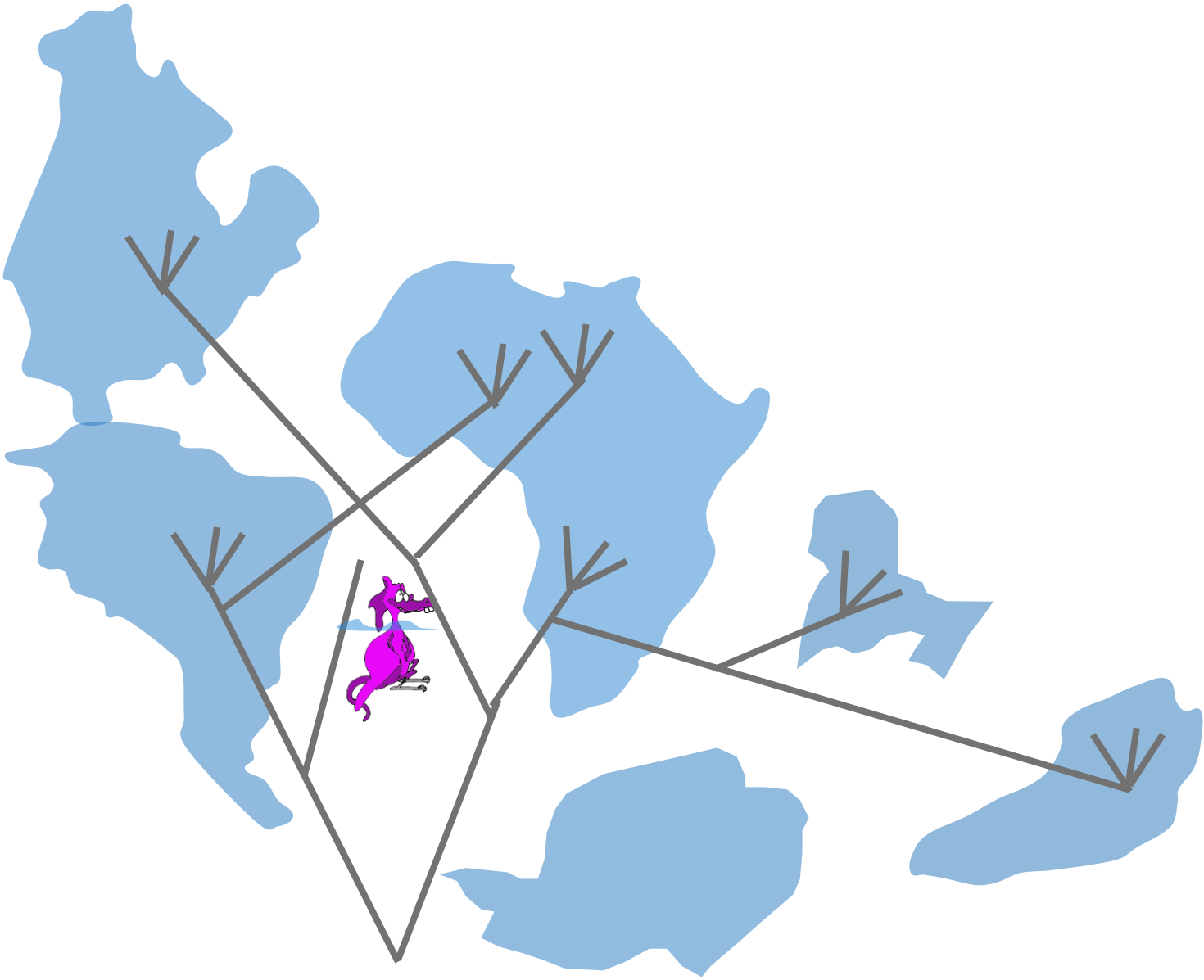


Figure 2.21: By the time of the final extinction of the dinosaurs, land masses were significantly fragmented, separated by large bodies of water and undergoing significant topological changes due to tectonic activity.

2.4 Relevance to WP3

In this section we have tried to take a global view of the dynamics of the largest-scale ecosystem that we currently have access to. Although further modeling work is needed, overall what we seem to be seeing is a need for a pool of scarce species as a reservoir of innovation that can respond to niche destruction, creation and evolution. As an ecosystem, the Earth has been able to respond positively and sustainably to long periods of physical change and stress, as well as short-term extreme events.

Our working hypothesis is that a digital ecosystem needs to have similar qualities in order to be stable against stress and change to its environment. The quality that our work is specifically targeted at protecting is the long tail of scarce species, and pool of innovation, for a DBE – the SMEs.

3 Modelling Distributed Transactions

As mentioned before, the main objective of work package 3 is the development of a P2P architecture to support distributed long-running transactions and facilitate open and trusted collaborations between SMEs in a digital ecosystem. Business transactions in this highly dynamic environment can become rather complex and a number of interrelated issues inherent in long-running multi-service transactions in a purely distributed setting arise, such as the exchange of data both within and between transactions (before their final commitment), compensation and recovery management when some transaction is aborted and ensuring consistency at all times. A transaction model is required to address such issues in a principled manner.

In this chapter we describe a distributed transaction model for OPAALS and examine the demands such a model makes on the P2P network that is intended to support it. We start with an outline of the basic characteristics of transactions in the context of digital ecosystems for business. We review current transaction models, which tend to be geared towards a centralised (or, at best, limited decentralised) architecture, and argue that these are not suitable for digital ecosystems as they do not consider a number of aspects that are critical for B2B transactions between open communities of SMEs. We then outline the proposed transaction model and highlight the features that make it particularly well-suited for long-running transactions in digital ecosystems. These include the local coordination of the underlying service compositions, the release of intermediate results within and between transactions, concurrency control based on a lock mechanism which also drives the compensation mechanism and allows for addressing aspects such as omitted results and forward recovery.

3.1 Multi-service long-running transactions in Digital Ecosystems

In Digital Ecosystems for business, networked organisations engage in complex transactions in order to perform some long-lived business activity. Our primary concern is thus with the support for long-running business transactions involving open communities of SMEs. A business transaction in this paradigm can be either a simple usage of a web service (rarely in B2B relationships) or a mixture of different levels of composition of several services from various service providers.

The conventional definition of a transaction [Dat96] is based on ACID properties: **A**tomicity – either all tasks in a transaction are performed, or none of them are; **C**onsistency – data is in a consistent state when the transaction begins, and when it ends; **I**solation – all operations in a transaction are isolated from operations outside the transaction; **D**urability – upon successful completion, the result of the transaction will persist. However, in advanced distributed applications these properties can present unacceptable limitations and reduce performance, a view also supported in [Elm94].

The necessity for change is derived from the very nature of business – as opposed to database – transactions. For example, the specification of a transaction may allow it to be completed over a period of hours or even days (a “long-lived” or “long-running” transaction). In the business environment many, possibly most, usage scenarios involve long-running transactions. In such cases, Atomicity provides an unacceptable constraint. Further, in long-running transactions, partial results may need to be shared between different transactions before their final termination (‘commit’). Failure to do this may at best lead to unacceptable delays in related transactions, and at worst leave a provider open to denial of service attacks (as data may be locked indefinitely in a non-terminating transaction). Hence, the Isolation property must be relaxed, and this then poses further challenges with regard to ensuring consistency of the underlying transaction model. In addition, the obligation for cooperation between transactions can be specified in a

business process rule (a requirement for availability of "partial results"). Finally, the instability of the internet environment can define a new requirement for keeping important results even when the connection between two platforms is lost ("omitted results"). These are challenging aspects that need to be addressed in defining a distributed transaction model for a digital ecosystem.

Current transaction models (targeted to web services) provide inadequate support for long-running distributed transactions [VZG+05], [FuG05]. We highlight such shortcomings in Section 3.2 but it is important to note that the current designs raise barriers to the adoption of the Service-Oriented Computing (SOC) paradigm. This is not perhaps as visible as it may be, due to the dominance of a relatively small number of highly visible service providers. The impact is much higher with regard to the restriction in adoption of SOC by small-to-medium sized enterprises (SMEs). Since SMEs contribute 50% of Europe's GNP, this is of significant concern.

The goal of SOC is to enable applications from different providers to be offered as services that can be used, composed, and coordinated in a loosely coupled manner. Web services in fact provide a realisation of SOC. Although recent years have seen significant growth in the use of Web Services, there are some very significant technological constraints that are stopping their full potential from being realised within a DBE environment.

The actual architectural approach of SOC is called SOA and is particularly applicable when multiple applications running on varied technologies and platforms need to communicate with each other. In this way, enterprises can mix and match services to perform business transactions with minimal programming effort. SOA is a way of reorganizing software applications and support infrastructure into an interconnected set of services, each accessible through standard interfaces and messaging protocols. The basic SOA is not only about architecture of services, it is a relationship of three kinds of participants: the *service provider*, the *service discovery agency*, and the *service requestor* (*client*). The interactions involve '*publish'*, '*find'* and '*bind'* operations (see Figure 3-1).

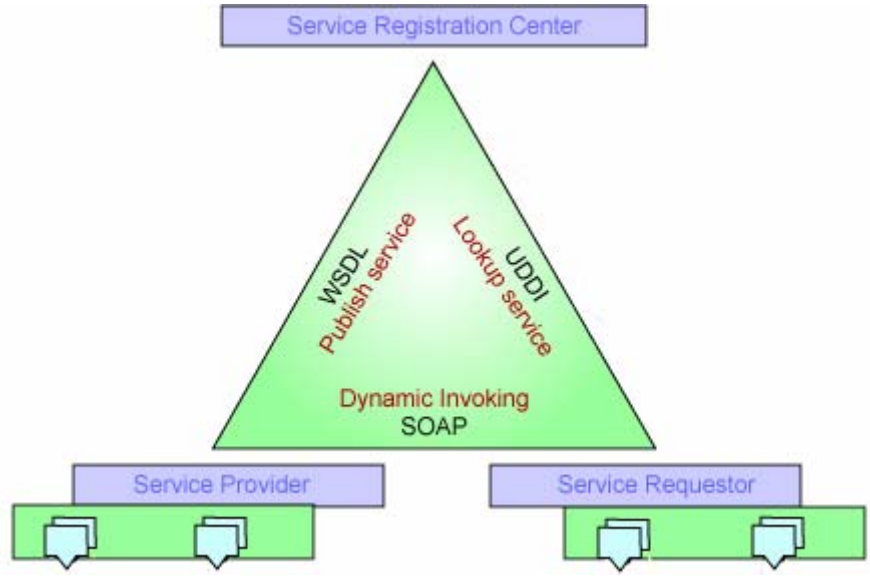


Figure 3-1 Service-Oriented Architecture (SOA)

In a typical service-based scenario a service provider hosts a network accessible software module (an implementation of a given service). The service provider defines a service description of the service and publishes it to a client or service discovery agency through which a service description is published and made discoverable. The service requestor uses a find operation to retrieve the service description typically

from a discovery agency, i.e., a registry or repository like UDDI, and uses the service description to bind with the service provider and invoke the service or interact with service implementation.

Furthermore, here is an important distinction between two broad aspects of services [Pap03]: *service deployment*, which is subjected to our transactional service composition, versus *service realization* (Figure 3-2). The service realization strategy involves choosing from an increasing diversity of different options for services, which may be mixed in various combinations.

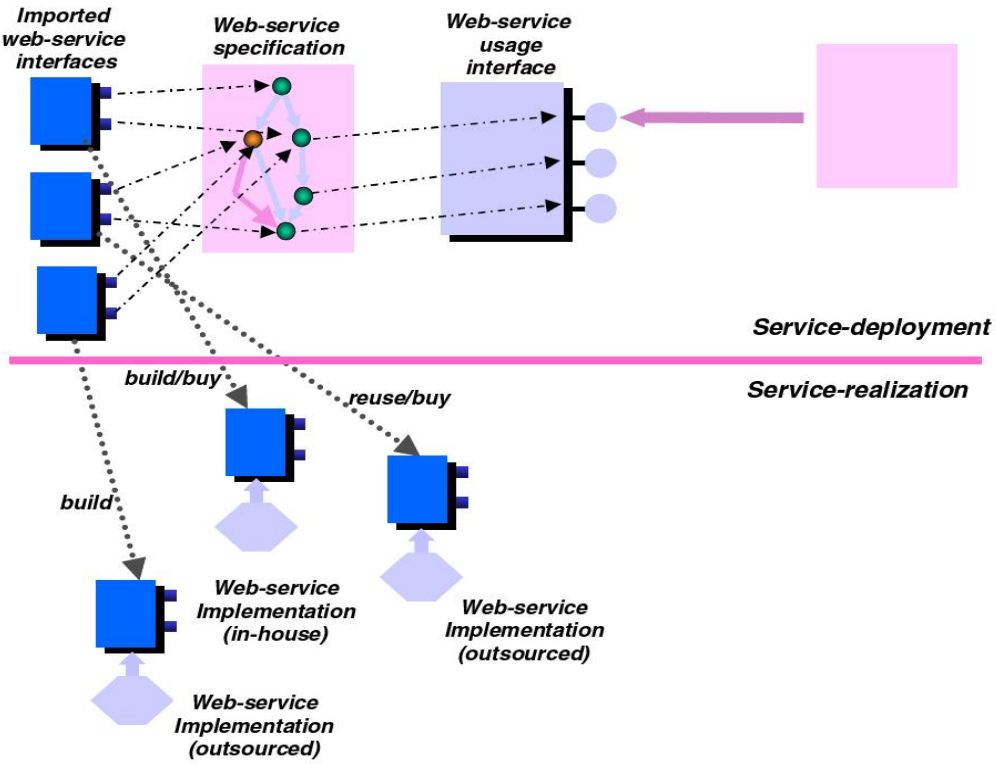


Figure 3-2 Service realization and service deployment in SOA

In the service realization terminology, *service implementation* can be very involved because on many occasions organisations rely on single monolithic programs to represent a single service or service method implementation. But very often in order to fulfil the functions of a service multiple programs are involved. Application composition and integration very often is required for fulfilling the service. At the logical level, what we consider is: 'there is a business function implemented in software somehow and this is the interface to it'.

The service in SOA is designed in such a way that it can be invoked by various service clients and is logically decoupled from any service caller (*loose coupling*). This means there are no assumptions of any kind in the service as to what kind of service a consumer is using and for what purpose and in what context. But the service callers are very much coupled with the service as they know what the services are, what they call and what they can accomplish. This can be considered as one of complexities of a service (transaction) composition manager, which has to encompass a polymorphous nature.

In service deployment, *service descriptions* are the most important part. They are used to advertise the service capabilities, interface, behaviour, and quality. Publication of such information about available services (on a service registry) provides the necessary means for discovery, selection, binding, and

composition of services. Later on, the importance and usage of service descriptions will be shown when services shall be used or composite services are created.

Based on this analysis, the research has to consider several issues such as transactional modelling, workflow manager, business models and service composition. According to the nature of web services architecture, taking into account a boundary for these issues is quite difficult and as a reflection of this fact most researches avoid to consider any boundaries. However, in practical situations somehow most transaction models are limited to a framework such as a workflow layer (typically reflected on different versions of BPEL such as BPEL4WS). These aspects will be addressed in various parts of the OPAALS project. In this report, we are mostly concerned with a distributed transaction model (with local coordination) and the preliminary design of a P2P network to support long-running multi-service transactions, and we touch upon the underlying service composition.

3.1.1 Service description

As the service composition will be performed at the service deployment level (see Figure 3-2 above), it requires the combination of service descriptions and specification of the ways in which they can be combined. Therefore, clear service specifications (descriptions) are needed to determine the possible service compositions. Based on [YPH02], a full service description can include the following.

- Service properties, i.e., service general information (e.g., identification, owner), service access information (e.g., service location -URL, the maximum time for a conversation between the service and a service requester, public key certificate), and contact information.
- Service ontology, i.e., what is the service about and the terminology that is needed to discover the service.
- Service cost, i.e., which is the estimated cost for using the service or the information provided by the service.
- Payment, i.e., the way the service receives the payment from the customers.
- Actors, i.e., which is the people or organizations who are using the service.
- Authorization/security/visibility, i.e., who can see/use what (service contents and functions).
- Service contents, which specifies the content and the structure of the underling service, e.g., the attributes, objects, the constraints on use of attributes/objects, etc.
- Service capability, which specifies the service structure/components, the conditions of using the service, and the order of component invocation.

Yang, Papazoglou and van den Heuvel introduce a service description language (SDL) which tries to cover all aspects of the service deployment needs [YPH02]. Its XML date type definition (DTD) can be found in [HYP01].

Based on a specification such as SDL advocated in [YPH02], service composition can be considered along the following dimensions: data, process, security, protocol. But by considering the main concerns of transaction model, our main apprehensions are data and process composition. We leave security and protocol compositions as an open discussion on the top of transactional layer. In particular, with process oriented service composition, we shall discuss the following aspects:

- ***Order***: indicates whether the composition of services is serial or parallel.
- ***Dependency***: indicates whether there is any data or function dependency among the composed services.
- ***Alternative service execution***: indicates whether there is any alternative service in the service composition that can be invoked - alternative services can be tried in either a sequential or a parallel manner.

3.1.2 Service Composition

Given that our distributed transaction model is to allow for open collaboration between SMEs within a digital ecosystem, despite what current technology (in particular) supports, we try to cover a various forms of service composition. The target is to cover a range of different forms of service composition that are required for setting an appropriate infrastructure for business modelling. Experience in the DBE project [DBE06] has shown that limiting the coverage (expressiveness) of service composition results in the need for more manual work in applying a business model to SOA, and this compromises the strategic role of transactions in building an aggregate service composition, described in [YPH02]

Inspired by the business-oriented view of SOA in [YPH02] we aim to cover the following forms of data- and process-oriented service composition.

a) **Data-oriented service composition**. The data generated from service realization level are released to cover different aspects service composition. According to transactional aspects this type of composition can be atomic or even considered as partial result for another transaction which force the necessity of compensability of this type of service composition. Typically Service composition in this case involves reorganization of the information and provides additional functions such as summarization and analysis of data.

b) **Sequential process-oriented service composition**. This type of service composition invokes services sequentially. The execution of a component service is dependent on its previous service, i.e. the service cannot begin unless the previous service commits. There are two possible scenarios for this composition:

   i. *Sequential with Commitment Dependency* (SCD). The relationship between component services is just sequential invoking orders without any interaction between them, which means one service has to wait for another service to commit before it can be executed. In the transactional sense only the order of execution should be considered in a coordinator.

   ii. *Sequential with Data Dependency* (SDD). This type of composition deals with more operational complexity. One component service relies on other component service's outputs as its inputs. As a result, in the transactional view the serializability of a data item has to be applied.

c) **Parallel process-oriented service composition**. In this service composition, all the component services can be executed parallel but different scenarios can be considered which can make different situations (implementations) in the transactional outlook:

i. *Parallel with Data Dependency* (PDD). Component services are executed parallel but one (or more) service(s) must wait for a specific data value from another service to arrive before it continues executing. On the other hand, the data serializability is an issue but there is not order execution in the transaction outlook.

ii. *Parallel with Commit Dependency* (PCD). This situation happens when two services may go in parallel to a certain extend, but one may not commit unless the other commits first.

iii. *Parallel with No Dependency* (PND). In this case, all the component services can run and commit in parallel without any interaction between them.

d) **Sequential Alternative composition** (SAt). This type of service composition indicates that there are alternative services to be combined, and they are ordered based on some criteria (e.g., cost, time, etc). Each will be attempted in succession until one service produces the desired outcome. In a transactional scenario, this can be considered as an important type for covering loosely coupled and very dynamic environment.

e) **Parallel Alternative composition** (PAt). Contrary to the previous case, alternative services are pursued in parallel. As soon as any one of the service succeeds the other parallel services are aborted (discarded).

3.1.3 High-level service composition

One of the motivations behind insisting on SOA as the enabling technology for distributed long-running multi-service transactions within a digital ecosystem is that applications from different providers are offered as services that can be used, composed, and coordinated in a loosely coupled manner. If the appropriate infrastructure is in place for open and trusted collaboration and business activities, participating SMEs have various opportunities for evolving their business and ensuring their sustainability. One such opportunity has to do with the levels of service composition that can be considered in such a setting.

In fact, the vision for service-oriented computing includes a much more ambitious agenda with regard to service composition [PTD+06]. This is mostly geared towards dynamic composition, but dynamicity comes not only from the service discovery and selection perspective, but also from the service requester's side in what is referred to as *explorative* or *semi-fixed* composition, depending on the level of involvement of the service requester and whether this is at run-time or while setting up the composition itself.

Early ideas on this more high-level type of service composition have appeared in [YPH02], even if only in a speculative manner. Current technology is not adequate for covering such forms of composition, but the fully distributed multi-service transaction model together with the supporting *autopoietic* P2P architecture being developed in OPAALS seems to be a step towards the right direction and can provide a solid foundation for realising this promise.

In what follows, we outline these types of high-level service composition and discuss them from the transactional modelling point of view.

Explorative composition

The main aspiration in *explorative* service composition is that his is generated on-the-fly based on the customer's request. The customer specifies the desired service and the specifications are then translated into the service specifications. The service discovery (broker) can then compare the translated specifications against the available services and come up with potential matches and propose feasible composition plans. These candidate service compositions are then evaluated by the customer and can be either accepted or the customer can add more requirements to his/her primary specification and produce more elaborate ones. The concept of explorative composition is described in Figure 3-3. Clearly, explorative service composition is particularly suitable for B2C relationships (where the customer is not fully aware of the detailed request from the outset), but also for B2B in which case the service requester is a small business (where the service requester does not know exact details of its request and/or does not have enough knowledge about service specifications or business modelling).

In this highly dynamic form of service composition, the customer or service requester can select from a set of candidate composition plans. Furthermore, a customer (in B2C) can even find out more about his/her desired usage scenario before committing to using a specific composite service. There is the additional possibility for giving ranked feedback on the returned candidate service compositions, and this is to be fed back to the system in devising the future candidate service composition plans. From a transaction point of view however, explorative composition requires some reconsideration of the concept of transaction to an extent, because this business action not only is interactive (and obviously can not be atomic) but also does not seem to fit in a conventional two-phased commit protocol.

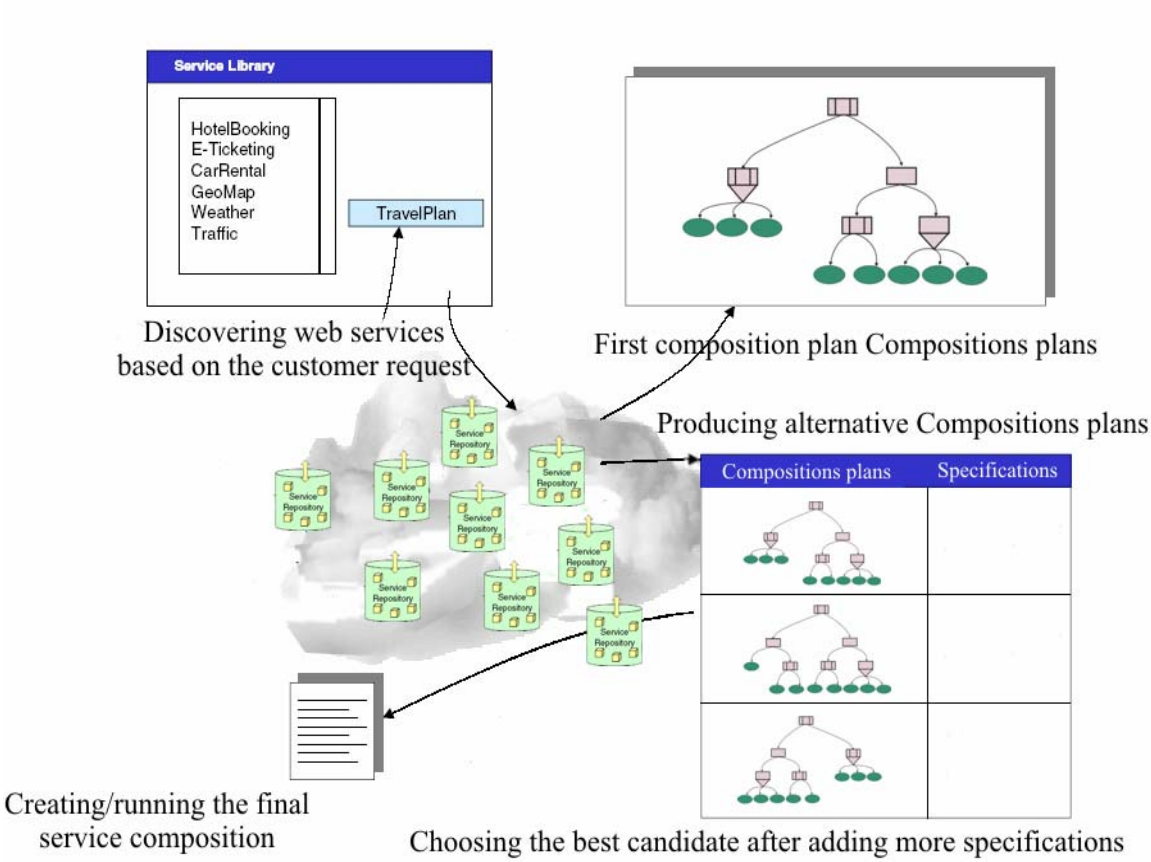


Figure 3-3 Explorative service composition – gradual refinement of specifications

Semi-fixed composition

In *semi-fixed* service composition, the assumption is that the service requester provides a formal composition definition (specification). But the actual service binding remains to be decided at run time and based on the availability of services (or, choosing alternative or equivalent services) at the time of the composition request. Not only does this composition cover the dynamic nature of web services on the internet but it is also a particularly useful solution for SMEs, especially since the availability of services is an important issue. Experience in the DBE [DBE06] project has shown that SMEs, who in this cse act as both service requesters but also service providers, are active depending on a variety of factors including geopolitical parameters and regional policies. In semi-fixed composition, the definition of the resulting composite service is registered in the marketplace, and it can be used as any other normal service, i.e. it can be searched, selected, and further combined with other services.

This type of service composition further motivates forward recovery and the provision for alternative scenarios. Also, it can considerably improve reusability in SOA (which is an aspect for which the SOC architecture sometimes comes under criticism). From a distributed multi-service transaction perspective, semi-fixed composition can be covered by the contingent transaction concept. However, one still has to consider the potential for a paradigm shift from the conventional view of a transaction as a predefined set of actions to a different view of a transaction as a dynamically instantiated set of actions.

Fixed composition

This type of service composition refers to the conventional model of service composition which is currently implemented and supported in most areas. In *fixed* service composition the composite service requires the component services to be integrated in a fixed way. The composition structure and the component services are statically bound. Requests to such composite services are performed by sending sub-requests to the component services. However, even in this well-studied case of service composition, a number of technical problems remain to be resolved. From a distributed transactions point of view, there are already constraints such as the need for centralised coordination, the tight-coupling of services and the limited depth of nested transactions, among others.

3.2 Current Transaction Models: Review

In this section we give a brief account of existing widely-used transaction models, namely Web Services Transactions and Business Transaction Protocol. It should be noted that there are a number of other transaction models, which are at a more conceptual level, but we have focused the discussion on the two most popular transaction models that have been designed with web services in mind. A more comprehensive review of existing transaction models can be found in [RKM06].

In 2001, a consortium of companies including Oracle, Sun Microsystems, Choreology Ltd, Hewlett-Packard Co., IPNet, SeeBeyond Inc. Sybase, Interwoven Inc., Systinet and BEA System, began work on the Organization for Advance Structured Information Systems (OASIS) Business Transaction Protocol (BTP), which was aimed at business-to-business (B2B) transactions in loosely-coupled domains such as Web Services. By April 2002 it had reached the point of a committee specification (see [CDF+03] and [FDF+04]).

At the same time, others in the industry, including Microsoft, Hitachi, IBM, IONA, Arjuna Technologies and BEA Systems, released their own specifications: Web Services Coordination (WS-Coordination) and Web Services Transactions (WS-AtomicTransactions and WS-BusinessActivities) [CCJ+04]. Recently, Choreology Ltd. started an effort for a joint protocol which attempts to cover both models. A number of

open problems with each protocol have been encountered and these are detailed in several reports by Choreology, e.g. see [FuG05].

In the following we outline the basic concepts behind these transaction models for web services.

3.2.1 Web Services Transactions (WS-Tx)

The Web Service Protocols (WS-Protocols) [CCJ+04] initiative on coordinating transactions in terms of their underlying web services comprises three protocols: WS-Coordination, WS-AtomicTransactions and WS-BusinessActivity. WS-Coordination [CCC+03] is the most important part of WS-Protocols as it defines the core coordination mechanism. It is defined as an extensible coordination framework and is intended to coordinate (sub)transactions at any level. WS-AtomicTransactions [CCC+04a] is intended to leverage the use of WS-Coordination in systems aware of ACID properties (systems in which transactions adhere to ACID properties).

Actually, it covers atomic transactions which are performed according to a conventional two-phase commit protocol – that is, a protocol defining two *atomic* phases, one for initialisation and one for final commitment. This is to be expected since transactions in WS-AtomicTransactions are understood to adhere to ACID properties, and thus can only be atomic. WS-BusinessActivity [CCC+04b] has been designed specifically to support long-lived activities. Hence, long-running transactions are considered in this protocol which tries to provide a sufficient foundation for applying long-term business activities in a transactional manner (hence, the name of the protocol).

It is worth stressing that both WS-AtomicTransactions and WS-BusinessActivity use WS-Coordination as the coordination framework. We describe WS-Coordination in more detail next.

WS –Coordination: outline

In the WS-Coordination specification, three roles are described for communicating parties: *Initiator*, *Participant* and *Coordinator*. These are shown in Figure 3-4. The Initiator is the entity aiming for consensus among multiple web services. The Participant is the entity offering some service(s) that need(s) to be coordinated during the interaction. The Coordinator is the entity coordinating the communicating parties in order to achieve the consensus.

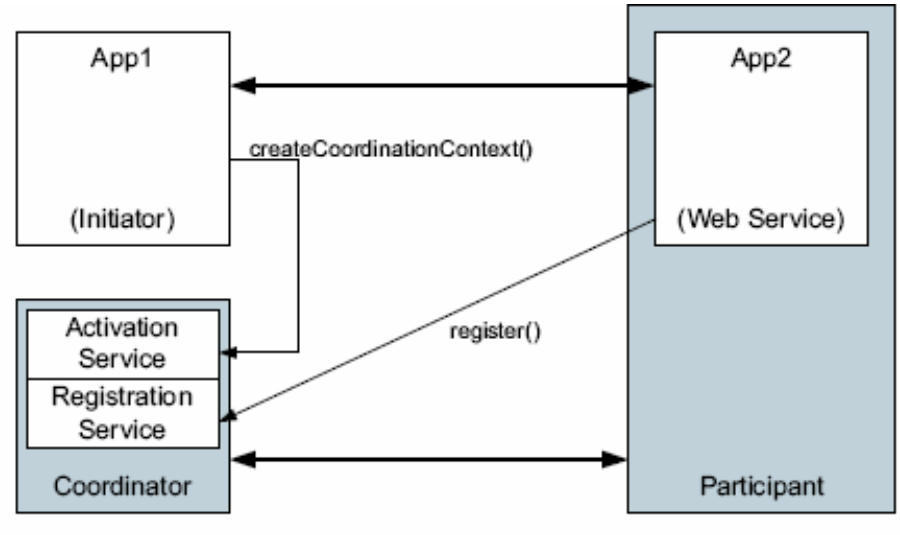


Figure 3-4 Initiator, Participant and Coordinator roles

The message exchanges between the different roles have to be introduced in the specification body, which requires the 'Activation' and 'Registration' of the participants. When a party wants to perform a transaction, it plays the Initiator role and, it does so by creating a CoordinationContext in the corresponding Coordinator, as shown in Figure 3-4. The CoordinationContext is acquired from the Coordinator's Activation Service in the activation phase. The CoordinationContext is attached to business messages being exchanged between the communicating parties, embedded in a SOAP (Simple Object Access Protocol) header.

The activation phase is started in the Coordinator once a CoordinationContext is created. All web services that need to be deployed in the transaction requested by the Initiator need to be registered with the Coordinator. This information allows the Coordinator to create (descriptions of) the logical steps required to execute the transaction.

The registering of the web services with the Coordinator is the actual registration phase. It should be noted that the two phases stand, in a sense, in a bidirectional relationship since the logical steps and the set of registered services are interdependent. Depending on what services are registered at that moment, the logical steps might need to be modified, and conversely, the logical steps need to take into account what services are currently registered.

The aim in the registration phase is to form a logical connection between the Coordinator and the Participant which is done by negotiation (through the bidirectional relationship mentioned above) and exchange of endpoint addresses between the Coordinator's and Participant's protocol services. (In this sense, the message flow over this logical connection depends on the coordination protocol being used and is not part of WS-Coordination specification.)

WS-AtomicTransactions and WS-BusinessActivity: outline

The WS-AtomicTransactions specification uses a conventional two phase commitment (2PC) protocol for coordination [BeN97]. This is to be expected since this protocol is concerned with atomic transactions, adhering to ACID properties, and hence a protocol which only consists of an initialisation phase and a final commitment phase seems to be appropriate.

On the contrary, the WS-BusinessActivity specification defines two different coordination protocols: the Business Activity with Coordinator Completion (BACC) (see Figure 3-5, upper half) and the Business Activity with Participant Completion (BAPC) (see Figure 3-5, lower half).

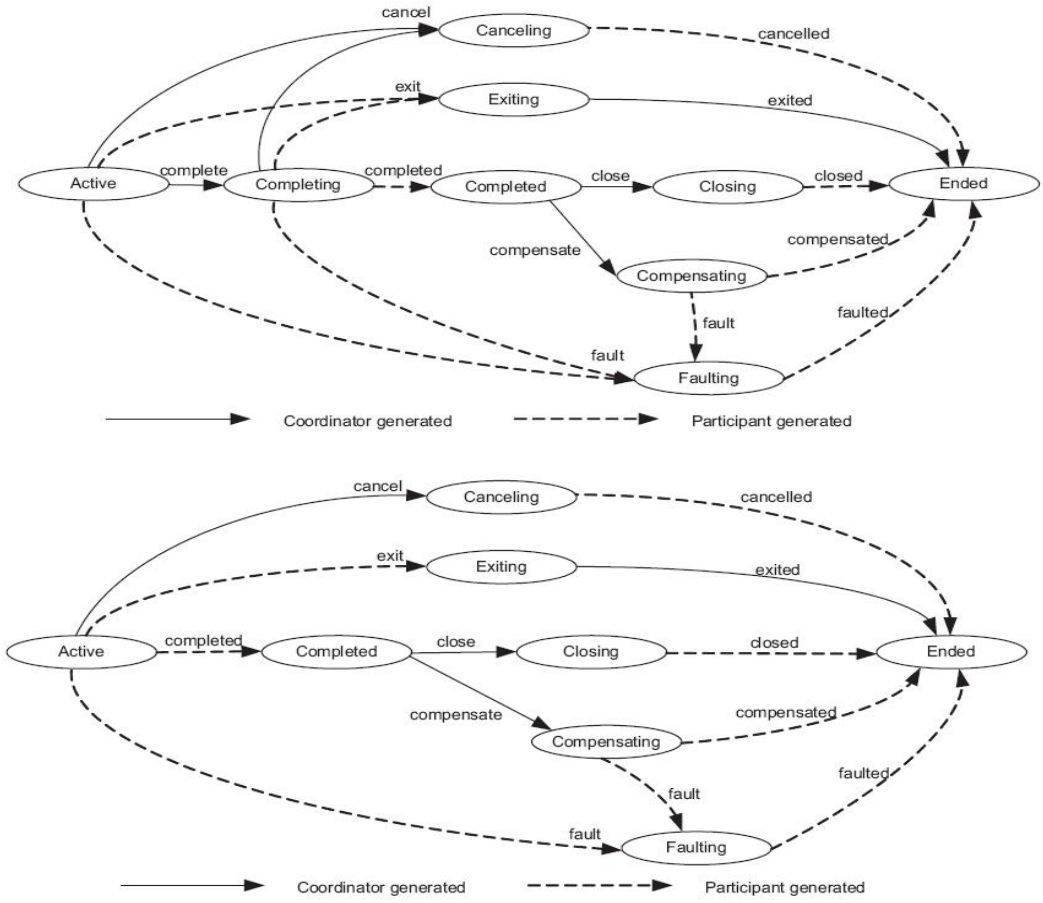


Figure 3-5 WS-BusinessActivity with Coordinator completion and Participant completion

The difference between the conventional 2PC protocol and BAPC / BACC mostly concerns the manner of their commitment phase. In BAPC and BACC this is not atomic. The first phase of both BACC and BAPC is used for the exchange of business messages between the parties. Then in the case of BAPC, the end of the first phase occurs when the completed message is sent from the Participant to the Coordinator, indicating that the Participant has completed its processing and all data has been stored persistently. The second phase is then used for confirmation or negation of any results achieved during the first phase. In the case of BACC, it is the Coordinator who can send the completed message after which the transaction transitions to the second phase (commit or abort).

In short, the difference between BAPC and BACC has to do with the procedures for business activities. The BAPC is designed for activities in which the decision about a transition from the first to the second phase can be made by the Participant whereas the BACC is designed for activities in which this decision is made by the Coordinator.

Discussion

Despite the intention for supporting both BACC and BAPC in WS-Tx, this is the case only for WS-AtomicTransactions in which transactions are atomic. WS-BusinessActivity only supports BACC in which the control is with the Coordinator. Further, and based on the analysis of Vogt and Zambrovski [VZG+05], WS-BusinessActivity can only support "do-compensate" behaviour patterns [FuG05] for the Participants. In the "do-compensate" pattern, the 'Confirmed' state has priority over the temporary states. But in the other behaviour patterns (such as "provisional-final" and "validate-do" [FuG05]), the Participant establishes a (temporary) 'Completing' state (see BACC in Figure 3-5) of the application data that will be changed to the 'Confirmed' state if and when the close message is received .

As a result of only supporting BACC and hence applying the "do-compensate" behaviour pattern, when a fault occurs, the transaction transitions to 'Faulting' state (see BACC in Figure 3-5), where only the Participant is involved. Since the control is with the Coordinator in this case (BACC), the Coordinator needs to have visibility of the Participant's states. Notice that after 'Completed', the 'Closing' and 'Compensating' states are controlled by the Coordinator. This has the implication of the Coordinator needing all details of the Participant to perform the compensating procedures. This limitation results in breaking the autonomy of the local platform as it forces to prescribe the internal behaviour of the realisation level of services, and this is in direct conflict with the primary foundation of the SOA paradigm (loosely-coupled services).

Additionally, the protocol authors made some pre-emptive design decisions about the internal build-up of the communicating parties, as described in [CCJ+04]. The tight-coupling between Coordinator and Initiator as well as the fact that the Participant encapsulates both business and transactional logic are examples of this. Such presumptions are against of basic SOA prerequisites (loose-coupling and highly dynamic composition [Pap03], [YPH02]), particularly with regard to the nature of web services. This should be of significant concern, especially when SMEs are involved in a transaction.

Another side effect of the identical roles of close in the "do-compensate" pattern behaviour, is the severe limitation on the possibility for alternative scenarios. These are necessary not only because they are a means of keeping a tight rein on the coverage of business requirements but also, considering the high dynamic nature of SMEs, because they cannot be a candidate for the Coordinator. In addition, following the design assumption (tightly-coupled) of WS-Transactions, and taking the necessity for stability into account, will force the system to continuously choose a few highly stable nodes as the Coordinators which, in the best case, will result in a decentralised system (and certainly *not* fully distributed as the designers promised [CCJ+04]). Further, it should be noted that SMEs cannot even be candidates for the Initiator as well as the Participant roles.

3.2.2 OASIS Business Transaction Protocol (BTP)

The Organization for the Advancement of Structured Information Standards (OASIS) has developed the *Business Transaction Protocol* (BTP) to provide a model for reaching the requirements of a SOA design with emphasis on long-lived collaborative business applications. BTP is designed to support long-running (long-lived) transactions and a transaction concept that goes beyond data-centric transactions [FDF+04].

BTP Structure

The BTP infrastructure is based on a nested transaction model and thus has a transaction tree structure. In contrast to other transaction models, BTP uses an open-top commit [DTL+03] protocol (sometimes

called a three phase protocol) which lets the application use an extra phase (as an intermediate phase) between the usual two phases of 2PC. This ability is provided by expanding the range of available verbs "prepare", "confirm", "cancel", etc to include explicit control over both phases.

BTP transactions types

The use of a tree structure for transactions in BTP allows for a mixture of different types of (sub)transactions. To address the specific needs of business transactions, BTP makes a separation in the categorisation of (sub)transactions by introducing two types of extended transactions. In the following, we briefly describe each.

Atom – the atom transaction type uses a classic two-phase commit (2PC) protocol and the transaction is considered as the "logical unit of work" which achieves a consistent result. In this context, a unit of work can be a number of business actions which is measured as an ACID transaction. An important feature of BTP,as described in its specification [FDF+04], is that these should either be completed together or be reversed to a state as if not executed at all. The *all-or-nothing* semantics associated with the Atom support precisely this style of interaction. In cases where a single party cannot fulfil its part in a business level agreement, or there is a technical failure that prevents that party from providing its service, the atom abstraction ensures that all parties involved in the interaction are freed from their obligations [DTL+03].

Cohesion – by relaxing Atomicity (in ACID properties), a Cohesion transaction tries to overcome the long-term (long-running) transaction problem. It permits the terminator of a transaction to confirm or cancel the selection of work based on the corresponding business model (high-level business rules). In fact, the cohesion concept at the architectural level aims to provide an abstraction for multi-party B2B interactions and it has a high reliance on the business model ontology. Based on this logic, and in contrast to a conventional transaction model, cohesion based on the business model can provide participants with different outcomes - which some might confirm while others cancel.

Cohesion uses a customised version of the two-phase commit protocol and, at a high level, specifies (in the intermediate phase) precisely which participants to prepare and which to cancel. During the life time of a long-running transaction, various conditions might be involved that require the cancellation or the preparation of some of its units, hence the cohesion might take several hours or days to arrive at its confirm-set (that is, the set of vital participants that must confirm for the successful termination of the transaction). When the confirm-set is determined, the cohesion collapses down to an atom, and at this final phase all members of the confirm-set see the same outcome. In this way, the action of cohesion means that the resulting transaction model is categorised as a conventional nested transactional model, which is slightly more flexible with regard to the commitment protocol but shares the same view of a long-running transaction in a tree structure.

BTP roles, actors and cast

A BTP role is specific to a particular relationship between the software agents (referred to as BTP *actors*) participating in a business transaction. Two types of relationships exist between different roles in the BTP model: *Inferior* and *Superior*, shown in Figure 3-6. The Inferior is applicable on Atom/Cohesion and is responsible for reporting to the Superior. It reports that it is prepared for the outcome, regardless of whether the associated operations' provisional effects can be confirmed or cancelled. The Superior is applicable on the coordinator/composer and is the party that coordinates the transaction. A Superior gathers reports from its Inferiors and must determine which to cancel or confirm. If the transaction is an atom, BTP's predefined rules require that every Inferior be confirmed unless at least one has signalled that it wishes not to, in which case all will be cancelled. If the transaction is of type cohesion, it will not rely on just the decision made by Inferiors and it has to be a combination of the decisions made by Inferiors and its own business rules - which are highly dependent on the corresponding business model.

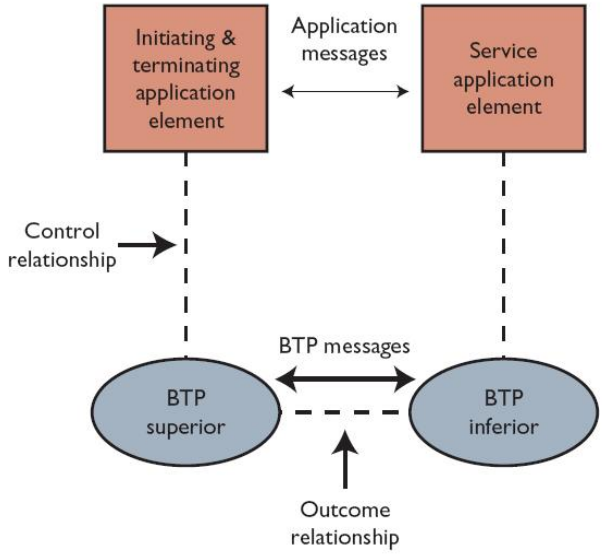


Figure 3-6 BTP roles and actors

BTP, as a nested transaction model, also stresses that each Inferior has only one Superior while a Superior can have multiple Inferiors. The tree of such relationships could thus be wide, deep, or both. A Superior coordinates an Inferior, as a rule of thumb, but the transaction's relative position in the transaction tree is important. This is because an actor's role could change during a transaction and infer different behaviour for the same logical entity.

Several other actors are introduced on BTP for handling more specific roles. One of the most important actors is a Superior for an atom and is called a *Coordinator*. It is responsible for the supervision of the atomic two-phase completion protocol. For cohesion, the most significant Superior is a *Composer*; (a sub-coordinator (atom) or sub-composer (cohesion)). Other important actors include the *Transaction factory* (which is typically a web service that creates the context for a business transaction and manages the coordinator associated with it), the *Initiator* (which starts a business transaction at an application element's request), the *Participant* (which is the entity that eventually does the real transaction work on behalf of the web service), and the *Terminator* (which is an application element that inculcates the Coordinator of a business transaction to either confirm or cancel the transaction).

Discussion

The first version of BTP was not specifically targeted to transactions for web services - the intention was to develop a more generic protocol that can also be used in other environments [CDF+03]. Actually, BTP defines the transactional XML protocol and must specify all the service dependencies within the specification. Recently Choreology Ltd. has tried to point out and modify the model and consider any common ground or possible connection with WS-Tx, for customising the BTP model for web services [FuG05]. (The reflection of these works can be seen in the latest version of BTP [FDF+04].)

As mentioned before, BTP is a conventional nested transaction model. This means that transactions can be nested and uncommitted results can be shared *internally*, but they are isolated from other transactions of the environment. That means that *partial results* (exchange of uncommitted results across transactions) can not be addressed in this model and this has serious ramifications with regard to the spectrum of business scenarios that can be covered.

In considering whether BTP is applicable in the context of a digital ecosystem, the other important point to consider has to do with the relationship between Superior (coordinator/composer) and Inferior. As there is no specific mechanism for alternative service composition in (specifically) Composers and (with less side effect) Coordinators, this relationship must be tightly-coupled (although this is not mentioned explicitly in the specification). If these roles are not tightly-coupled, then the whole transaction will face the risk of a regular cascading roll back. As a very simple example, consider the case of SMEs which by their very nature are offering services dynamically with loosely-coupled binding, which means the accessibility and characteristics of the provided services can be changed regularly. The immediate implication is that in the event of a failure or abortion the Composer/Coordinator has to be rolled back and in the case of the Composer, precisely because it is not isolated from other Composers of the same transaction, it is often the case that it might share some of its results with other Composers and consequently all dependent Composers have to be rolled back or be restarted too.

Another side effect of the last problem, the tight-coupling between Superior and Inferior or more generally between Coordinator and Participant, is the probability for deadlock. The reaction of the transaction model in the occasion of deadlock is not mentioned in the specification. In fact, there is no clear method / approach to detecting deadlock in the first place. Furthermore, even if there was provision in the internal structure for deadlock detection or prevention, the possibility for regular abortions (rollback/restart) make for a high probability that deadlock will indeed occur at some point. Further details can be found in [RKM06], which includes a discussion on a number of other open issues in BTP such as recovery management algorithm, fully distributed coordination and concurrency control.

3.3 Distributed Transaction Model for OPAALS

In this section we outline the proposed model for distributed long-running multi-service transactions in OPAALS. The autopoietic P2P network is to support a healthy and diverse socio-economic ecosystem that is the primary "business goal" of the OPAALS project. From that comes a specific focus on supporting and enabling a service-oriented business environment which facilitates business transactions between participating SMEs in a loosely coupled manner without relying on a centralised provider.

We start by describing the pre-conditions and motivations behind our model which is designed for transactions between open communities of SMEs in digital ecosystems for business.

Long-term transactions

A high range of B2B transactions (business activities [CCC+05] or business transactions [RKM06]), has a long execution time period. Strictly adhering to ACID properties for such transactions can be highly problematic and can reduce concurrency dramatically. The application of a traditional lock system (as the concurrency control mechanism [Dat96], [Gra93]) for ensuring Isolation, or capturing some version of serializability [Dat96], [Gra93], reduces concurrency and the general performance of the whole system – many transactions have to wait for a long-term transaction to commit and release its resources or results. As a side effect, the probability for deadlock is also increased since long-term holding locks directly increase the possibility for deadlock. Furthermore, the lack of centralised control in a distributed transactional environment such as DBE hinders the effective application of a deadlock correction algorithm.

Partial results

Releasing results from one transaction to another before either transaction commits is another challenge in the distributed transactional environment. According to conventional transaction models, releasing results before transaction commit is not legal, as it can misdirect the system to an inconsistent state when the transaction is aborted before final commit and all released results are not valid anymore.

It may be argued that this situation does not happen regularly, but it is a crucial requirement which has to be automated. This is because not only does it force the user to design a really complicated transaction but also he/she has to change the business model for defining some new transactions which do not match with the nature or reflect the reality of the business. Alternatively, an important part of the business requirements has to be executed manually.

Allowing for partial results and ensuring consistency are two aspects that clearly cannot fit within a conventional lock mechanism for concurrency control and log- or shadow-based recovery management [Dat96]. A wide range of business scenarios however demand partial results in specific circumstances and therefore we need to reconsider this primary limitation but at the same time also provide consistency for the system.

Recoverability and failures

In the dynamic environment of distributed business transactions there is a high probability for failure and thus recoverability of transactions is quite important. Recovering the system in the event of failure or abortion of a transaction needs to be addressed in a way that takes into account the loosely-coupled manner of connections. This makes a recoverability mechanism in this context even more challenging. As we can not interfere with the local state of the underlying services, the recovery has to be done at the deployment level [YPH02], [Pap03] and service realization (which includes the state of a service) has to be hidden during recovery. This is a point which current transactional models often fail to address as will be further discusses in the sequel.

Diversity and alternative scenarios

By integrating SMEs the DBE provides a rather diverse environment for business transactions. The provision for diversity has been discussed in the literature for service composition [YPH02], [Pap03]. When considered at the transaction model and/or business processes, it can provide a unique opportunity in not only covering a wider range of business processes but also in designing a corresponding recovery system [RMK07a], [RMK07b]. In conventional concurrency control and recovery management there is no technical consideration of using diversity for improving performance and reliability of the transactions.

Omitted results

One point of criticism for recovery systems often has to do with wasting intermediate results during the restart of a transaction after accruing a failure. The open question here is how much of these results can be saved (i.e. not being rolled back) and how. In other words, how we can preserve as much progress–to-date as possible. Raising to this challenge within a highly dynamic environment such as DBE can have significant direct benefits for SMEs in terms of saving time and resources.

Levels of centralisation

In different transactional models and workflow managers, there exist varying levels of centralisation by using centralised (limited level of decentralised) coordinator(s) (cf Chapter 4). Inherently this causes

limitations on composition, compromises the ability for automating different types of compositions (recall Section 3.1), reduces platform autonomy and the range of partial results (which is another important feature for covering a wide spectrum of business requirements). Therefore, with respect to additional architectural complications, we try to apply a fully distributed structure, which enables full autonomy of the local platforms.

Platforms failures and contingencies plan

Platform failures (even inaccessibility of platform in small and medium enterprises) are one of the natural events in web services that can easily cause a transaction to fail. But according to the user specification request, range of accessible alternative web services or even business model, which triggered the transaction, most of these incidents should not cause the transaction to fail (either can be covered by alternative web services or expected by user in a period of time).

For example, if one of the composed services is supposed to be a web service from UK's IBM platform and it is not available, the same web service can be deployed from US's IBM platform. Or if one of the composed services is buying a specific model of goods from a retailer and the retailer web services fails during the service composition (or transaction commitment), an alternative retailer which fulfilled the user description can provide an alternative web service for the transaction (service composition).

Another example is about the small and medium enterprises, which provide services only within specific periods of time per day (according to Sun Microsystems, this is more than 60% of SMEs in Spain and based on their study, much the same proportion in the rest of EU). Therefore in some period of time, any transaction requiring a web service from these SMEs will fail, even when this behaviour is expected and a user could wait for this specific service time period.

Another parameter that needs to be taken into account in this highly dynamic environment has to do with the lack of bandwidth, which may be due to some traffic bottleneck or simply a low bandwidth connection. We specifically introduce a delegation mechanism within our transaction model that transfers a transaction to another local coordinator and that includes giving full autonomy to the destination platform but also providing facilities for overcoming the variation in available bandwidths.

3.3.1 Transaction structure

The long-term nature of business transactions frames the concept of a transaction in a digital ecosystem for business and makes defining a consistent transaction model even more challenging. The conventional view of a transaction [Gra93] is based on the ACID (Atomicity, Consistency, Isolation, Durability) properties, which have been successfully applied to relational database management systems [Dat96]. However, in many new distributed applications such as CAD projects, e-commerce solutions and advanced simulators, these properties present unacceptable limitations. It can be argued that the conventional view of a transaction cannot capture the primary requirements of the era of DBEs.

From a business point of view, most usage scenarios in Digital Ecosystems involve long-term transactions and thus Atomicity is an unacceptable constraint. From a SOA for distributed transactions point of view, Isolation can lead to significant degradation of performance in the services offered (as critical data is locked until a transaction completes) or to increased probability of deadlock (as services may be locked into composite transactions that do not terminate). Additionally, in long-term transactions, partial results need to be shared between different transactions before their termination (commitment). This poses further challenges with regard to ensuring consistency of the underlying transaction model.

Typically, a two phase commit (2PC) protocol is recommended for nested transactions [Mos85]. The necessity of two phases is underlined by the long-running nature of business transactions. The first phase prepares the transaction and involves declaring dependencies, setting up the relationships, and indicating the boundaries and side-effects of updates (possibly using locks). The second phase is supposed to finalise or abort the transaction.

In the proposed model, we consider an additional intermediate phase which concerns potential failure of subtransactions (one or more coordinators or services). In brief, the introduction of an intermediate phase between 'preparation' and 'final commit' or 'abort' allows some further flexibility, in terms of recovery from failure, before the outcome of the transaction is determined. This adaptation of 2PC allows some leverage in attempting alternative subtransactions, or re-starting only the failed subtransaction, before re-starting the whole transaction. This will be further discussed in Section 3.4 of this report where we describe a forward recovery mechanism within our distributed transaction model.

2.3.2.1 Local coordination

In the Digital Ecosystem paradigm, networked organisations are expected to engage in complex transactions involving the coordination of a number of sub-transactions. In the case of SMEs, the participation of which already comes with strong demands for local autonomy, such open collaborations can lead to their sustainability within the digital ecosystem, but this certainly requires a fully distributed solution for transactional modelling.

The demand for local autonomy is in line with the concept of loosely-coupled services, a prerequisite for considering SOA as the enabling technology for the digital business ecosystem. This means that the multi-service long-running transactions must be coordinated locally, since in a fully distributed architecture there is no centralised control.

In fact, at the heart of our transaction model are the local coordinators. This is where most complexities of the model are handled such as coordination, service orchestration, keeping logs for managing dependencies and implementing a recovery mechanism, handling the low bandwidth, dealing with the low processing power of some nodes in the network, and so on.

Based on the different kinds of service composition within our model, we use different types of coordinators. We have opted for a nested transaction model in that a transaction is understood to comprise a nested group of sub-transactions. In our approach, a transaction is represented by a tree structure (see Figure 3-7). The root of the tree represents the main composition and is thought of as the initiator of the transaction. Each node is either a coordinator (a composition type) or a basic service (a leaf). There are five different coordinator types, drawing upon [YPH02], [Pap03], [PTD$^+$06[], plus a delegation coordinator for handling low bandwidth connections. We describe each below.

- **Data-oriented coordinator:** This coordinator is specifically working on data oriented service composition; including fully atomic and simple service oriented which is dealing with released data item inside of a transaction or using partial results, released by other transactions.

- **Sequential process-oriented coordinator:** This coordinator invokes its sub-transactions (services) sequentially. The execution of a sub-transaction is dependent on its previous service, i.e., one cannot begin unless the previous sub-transaction commits. In fact this coordinator handles sequential process-oriented service composition by covering both Sequential with Commitment Dependency (SCD) and Sequential with Data Dependency (SDD).

- **Parallel process-oriented coordinator:** This coordinator handles parallel process-oriented service composition. All the sub-transactions (component services) here are executed in parallel.

Different versions of the parallel coordinator can be considered which result in different implementations in the transactional outlook: Parallel with Data Dependency (PDD), Parallel with Commit Dependency (PCD) and Parallel without (or No) Dependency (PND).

- **Sequential alternative coordinator:** This coordinator indicates that there are alternative sub-transactions (services) to be combined. These services will be attempted in succession until one produces the desired outcome, according to some criterion (e.g., cost, time). In fact this coordinator is particularly useful in a highly dynamic environment as it can provide the basis for forward recovery (cf Section 3.4).

- **Parallel alternative coordinator:** This coordinator indicates that there are alternative sub-transactions (services) to be executed in parallel and once a service produces the desired outcome, the rest are aborted. Actually the parallel alternative coordinator handles Parallel Alternative composition (PAt).

- **Delegation coordinator:** this coordinator allows the whole transaction or a sub-transaction to be delegated to another platform, e.g. as a means of overcoming traffic bottlenecks or low bandwidth connections. Delegation can be by sending a request specification or service(s) description.

Figure 3-7 shows the symbolic view of the model. In the graphical description of a transaction, we have adopted the notation found in various works from the areas of both transactional modelling and service-oriented computing. In fact, apart from the notation for the data-oriented coordinator, the symbols used are standardised in [HPS93], [Hag97], [Hag95], [VeP98], [PDB+97], [PDH+96].

The tree structure representation of a transaction allows us to exemplify the local coordination that is required for the services involved to be performed in unison in accomplishing the goal prescribed by the transaction.

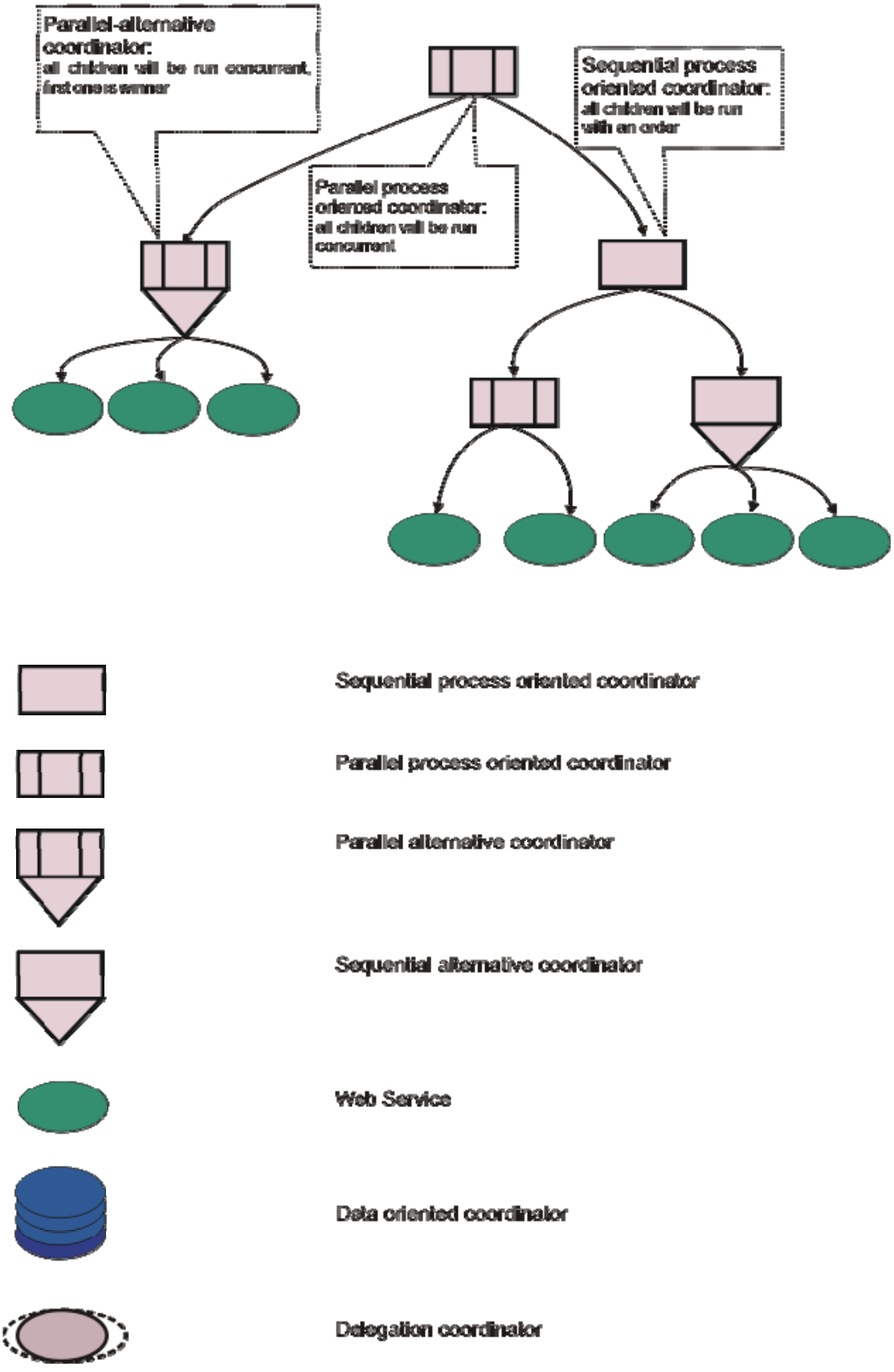


Figure 3-7 Long-running transactions in a tree structure

3.3.2 Managing dependencies: local consistency graphs

We have seen that in our approach transactions are understood as pertaining to SOC for B2B interactions. Hence, a transaction has structure, comprising a number of subtransactions which need to be coordinated accordingly (and locally), and execution is long-term in nature.

In order to relax the ACID properties, particularly Atomicity and Isolation without compromising Consistency, we need to consider some additional structure that will warranty the consistency of the transaction model. Maintaining consistency is critically important within a highly dynamic and purely distributed environment of a digital ecosystem. But at the same time, the model should defer from any tight-coupling between initiator and coordinator or between initiator and participant, as is the case with WS-Coordination [CCC+05] discussed in Section 3.2. In order to ensure that we can exploit the potential benefits of SOC, we want to respect its primary requirements such as loose-coupling. We therefore keep state information at the deployment level, rather than the realisation level, and defer from interfering with service execution as such as we wish not to break local autonomy of the realisation platform.

Note that by 'state' here we refer to the activations within a transaction, in terms of its local coordinators and/or subtransactions at any given point, and not to state as in the execution of the invoked services. The latter notion of state is related to the realisation level and is to be dealt with by the local platform on which these services run.

In what follows, we introduce two directed graphs that capture the dependencies between sub-transactions (coordinators or services) of a single transaction or belonging to different transactions. Keeping track of such dependencies is essential if the underlying distributed transaction model is to provide capabilities for taking reverse action (in case some subtransaction or even a whole transaction fails or is aborted), for applying deadlock control, for providing transparency during delegation. Furthermore these two graphs together with the transaction tree allow for releasing partial results (before 'commit') between different transactions in our approach. In fact, these graphs are very important system logs, which are stored *locally* on a coordinator and will be effected locally (in term of local faults, forward recovery and contingencies plan) and globally (abortion, restart, etc.).

3.3.2.1. Internal Dependency Graph (IDG)

In the conventional view of a transaction taken in databases, by using classic mechanism (such as lock-based concurrency control, Log-based/Shadow page Recovery management, etc), ACID properties are supported and consistency of the environment is guaranteed. Meanwhile for covering other problems such as deadlock, more structure (mostly graph-based) and additional mechanisms are used.

In the proposed transaction model, not only are the above concepts (consistency, durability, deadlock and etc) considered, but also a method is designed for releasing some of its modified/generated data even before transaction commit. In the concurrency control (cf Section 3.4), new Locks are used for full coverage of consistency and environment requirements. Furthermore, using two different graphs provides the possibility for applying data dependencies in a consistent way (which shows internal data structure of coordinators). Meanwhile, (taking into consideration the relationship between locks and these graphs) applying different anti-deadlock mechanisms is possible. In this section, the internal data dependencies and the designed graph for that will be analysed and subsequently, in Section 3.3.2.2, the other type of graph will be explored.

The *Internal Dependency Graph* (IDG) is a directed graph in which each node represents a coordinator and an arc indicates a dependency between the respective coordinators. The direction of the arc shows which node depends on which other. Its purpose is to keep logs on value dependencies in a transaction tree.

In further explanation, when a coordinator wants to use a data item belonging to another coordinator, two nodes have to be created in the IDG (if they do not already exist) and an arc generated between them (the direction of which shows the dependency between the two coordinators).

Consider a multi-service long-running transaction whose underlying service orchestration includes a Sequential with Data Dependency (SDD) coordinator. This means that (at least) one service or child of this coordinator relies on the data outputs of a previous service of the same coordinator. The children of the SDD coordinator will be executed one after the other and each child can make use of the data results of the previous ones. If the SDD has *n* children these can work with different data items so long as they data items used by subsequent children are released before their execution. So, if IDS_i and IDS_{i+1} are two children of the SDD and IDS_i comes before (is to the left of) IDS_{i+1} then IDS_{i+1} can use the data output of IDS_i.

This dependency between the children of the SDD coordinator has a further implication when it comes to recovery management, since if IDS_i is aborted then IDS_{i+1} must be aborted too (because it has used the data released by an aborted child). Thus, it is important to keep track of such dependencies and this is captured in the corresponding IDG, as shown in Figure 3-8, where a directed link is created between IDS_i and IDS_{i+1}. This is represented by the solid line directed arc in the figure. The hashed directed arcs denote other possible dependencies that might be inferred by the rest of the transaction tree (not depicted in Figure 3-8).

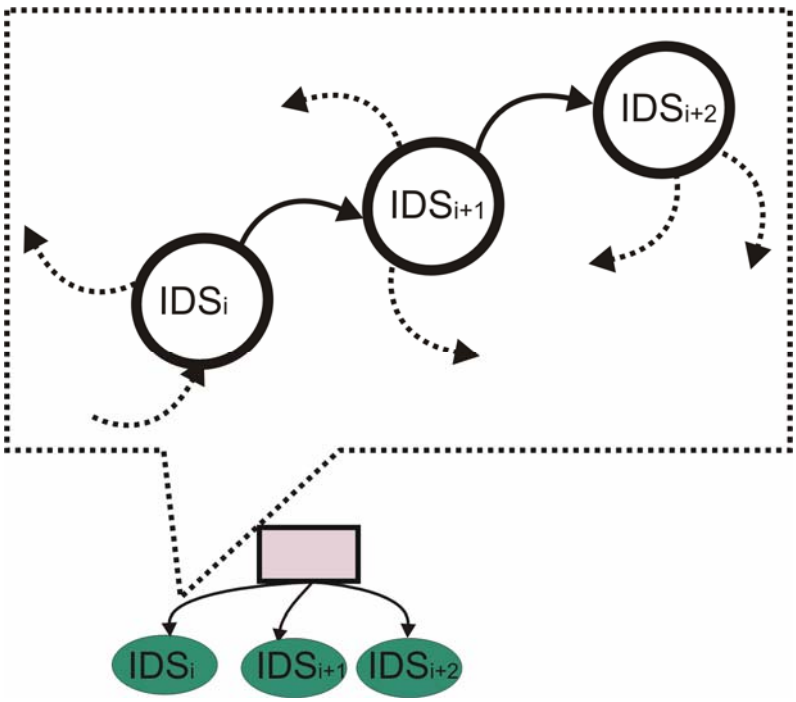


Figure 3-8 SDD coordinator and its associated Internal Dependency Graph

A similar approach is taken in the case of a Sequential with Commit Dependency (SCD) coordinator in a multi-service transaction. This means that one child has to wait for the previous to complete its 0065ectuion and commit, before proceeding with its own execution. For example, if there is a commit dependency between the children IDS_{i+1} and IDS_{i+2} of the SCD in Figure 3-8, then a directed link will be created between these two nodes in the corresponding IDG, also shown in Figure 3-8. As mentioned before, the direction of the arc is drawn from the dependent node to the other.

Next consider the case of a multi-service long-running transaction whose coordination includes a Parallel with Data Dependency (PDD) coordinator. This means that the component services of this coordinator are executed in parallel but (at least) one service or child of the PDD has to wait for a specific data output from some other service (another PDD child). In such occasions, if one of the children of the PDD is aborted (or re-started) the other children that have used its specific data results must also be aborted. It is the purpose of the corresponding IDG to capture such dependencies and, in the event of failure, allow for the transaction to be rolled back in recovery management.

Let IDS_i and IDS_j be two children of a PDD coordinator in a long-running transaction, and assume that IDS_i has to use the result released by IDS_j. If IDS_j fails or is aborted for some reason (platform failure, disconnection, low bandwidth and so on), then IDS_i must also be aborted because the data it has used is no longer valid. This dependency is captured in the corresponding IDG, as shown in Figure 3-9.

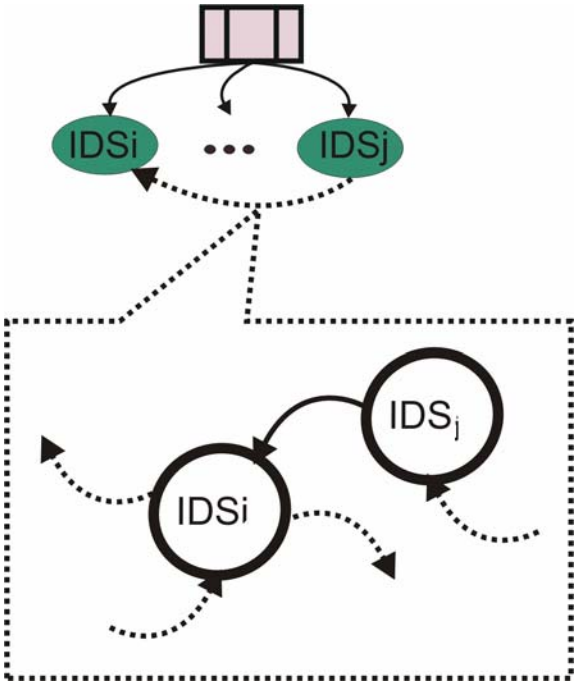


Figure 3-9 PDD coordinator and its associated Internal Dependency Graph

A similar approach is taken in the case of a Parallel with Commit Dependency (SCD) coordinator in a multi-service transaction. This means that (at least two of) the component services of this coordinator are executed in parallel, but only to a certain extent since one cannot commit unless the other does. For example, if there is a commit dependency between the children IDS_i and IDS_j of the PCD in Figure 3-9, then a directed link will be created between these two nodes in the corresponding IDG, included in Figure 3-9. As mentioned before, the direction of the arc is drawn from the dependent node to the other. Hashed directed links indicate dependencies on other parts of the long-running transaction, not shown here.

In what follows we demonstrate the role of the IDG in managing dependencies within a multi-service transaction by means of a small example – the example is primary but complicated enough to highlight the main concept behind employing the log-based structure of the IDG. Figure 3-10 shows a long-running transaction comprising four basic services whose order of execution is determined by the five local coordinators employed in the corresponding tree structure.

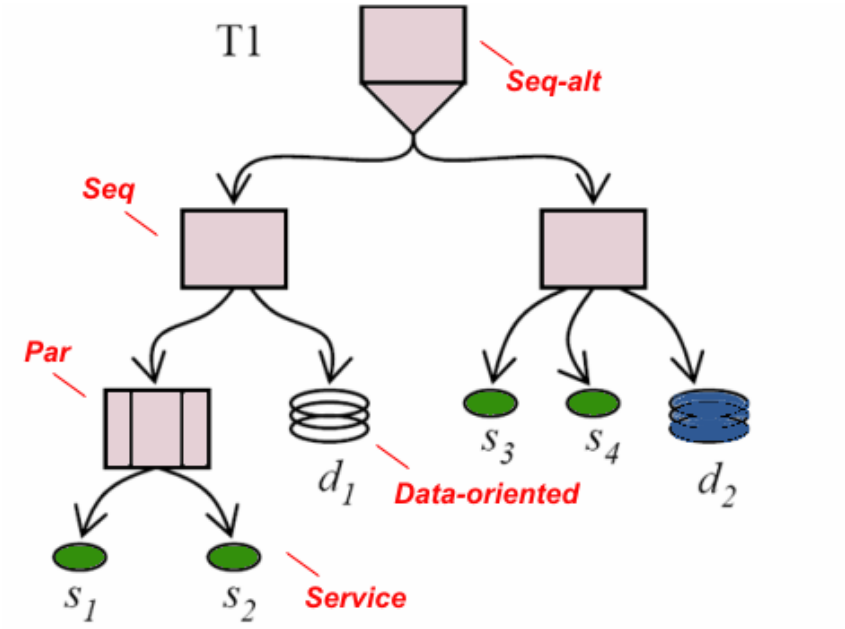


Figure 3-10 A long-term transaction with 4 basic services and 5 local coordinators

The tree structure of the transaction says that services s_1 and s_2 are to be executed in parallel (notice in the tree their parent is a parallel coordinator) and then the result is passed on to the data-oriented coordinator d_1, who can release it to another transaction before the transaction T1 commits. In fact, the mechanism for releasing partial results in a consistent manner is more complicated and will be explained further in Section 3.4 where we introduce a *conditional-commit* lock (cf Section 3.4.3) for precisely this reason. For now, it suffices to say that a result can be released, before T1 commits, to another transaction via a data-oriented coordinator, either d_1 or d_2 in this case.

In fact, in our example the data-oriented coordinator d_2 will only be used if the result of d_1 does not meet a pre-specified criterion or some failure occurs during the parallel execution of s_1 and s_2. This is determined by the use of the sequential alternative coordinator at the root of the transaction tree of T1. This means that if something goes wrong whilst performing the left branch of T1, then the right branch gets executed. The right branch says that execution will continue with s_3 followed by s_4 and the result is passed on to the data-oriented coordinator d_2, who can release it to another transaction (does not necessarily have to be the same transaction that was targeted by d_1).

It can be seen that in this transaction design each platform only needs to know about services dependent on its own. For instance, the service s_4 only needs to know that it gets executed once s_3 finishes its own execution. It may require some result from s_3, in which case their parent would more specifically be a sequential with data dependency (SDD) coordinator and so on.

Most importantly, it does not need to know why s_3 got executed, whether and for what reason the result of d_1 did not meet the criterion (and what criterion) and so on. It is shielded by all other information of the transaction. It only needs to know it depends on the execution of s_3 and it passes on its result to the data-oriented coordinator d_2. In other words, it only needs to know what happens immediately *before* and *after*, nothing else. This dependency, and assuming their parent is a SCD coordinator in particular, is shown in the corresponding (part of) the IDG of Figure 3-11(i).

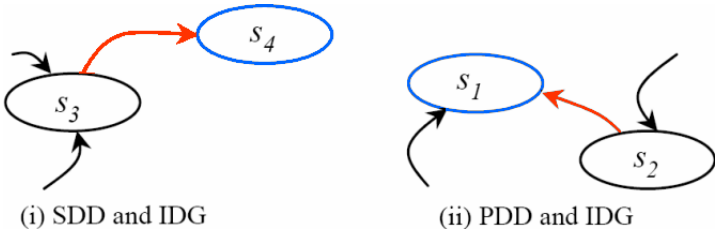


Figure 3-11 IDG for the SDD and PDD of transaction T1

Figure 3-11(ii) shows the dependency between the services s_1 and s_2 which are to be executed in parallel. This is captured in the (part of) the IDG of Figure 3-11 (on the right) where we have made the further assumption that the parent of s_1 and s_2 is a parallel with data dependency (PDD) coordinator – it is for this reason that the direction of the arc is from s_2 to s_1, denoting that s_2 depends on receiving a data item from s_1 at some point during their otherwise parallel execution.

3.3.2.2. External Dependency Graph (EDG)

In a highly dynamic and purely distributed environment such as a digital business ecosystem (DBE), it is often the case that a sub-transaction requires access to a data item released by a subtransaction belonging to a different transaction, before that transaction commits. This situation is often referred to as exchange of *partial results* between transactions.

In other words, dependencies may exist not only *within* a transaction but also *between* transactions (which may take place on different platforms). For example, consider the case of (compensable) subtransactions that release partial results whilst in a conditional-commit state [PDH+96], [PDB+97], [Hag97], [Raz99]. The support mechanism can be different depending on the environment and the corresponding data structure. Here, we aim to use a compensating mechanism for providing integrity *and* consistency. In this way, data integrity is maintained through a compensation routine when a failure occurs.

To capture such dependencies we introduce the *External Dependency Graph* (EDG). This keeps track of dependencies between (services or coordinators of) different transactions that have engaged in an exchange of partial result. The log structure this graph provides can be used in recovery management for running a compensating procedure, and this will be further discussed in Section 3.4.

When a sub-transaction wants to access a released data item which belongs to another long-running transaction (before 'commit' of either transaction), this dependency is shown by creating a directed arc between the corresponding nodes from the owner to the user of the data item in question.

For example, Figure 3-12 shows the release of partial results from two subtransactions of $IDHC_1$ and $IDHC_2$. As shown in the figure the two nodes appear linked in the corresponding EDG – notice the direction is towards the consumer of data thus indicating this data item usage.

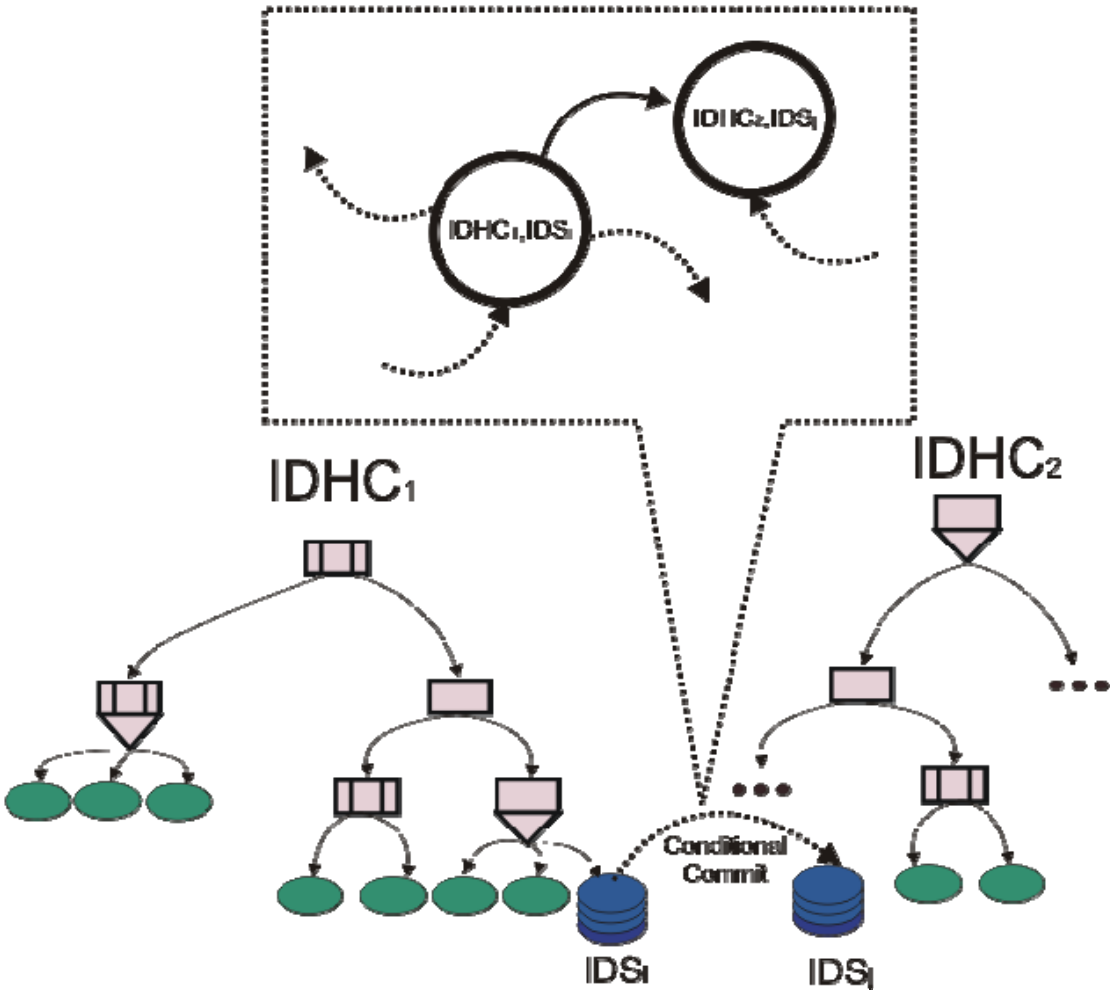


Figure 3-12 Partial results and EDG

If either of these nodes is absent in EDG they must be added and if the nodes and a connection between them already exist, there is no need for repetition. The most important usage of this graph is in the creation of compensatory transactions during a failure.

In what follows we demonstrate the role of the IDG in managing dependencies within a multi-service transaction by means of a small example – the example is continued from the previous section where we looked at the IDG. Figure 3-13 shows part of the EDG for the transaction trees T1 (of Figure 3-10, given earlier) and T2. The data-oriented coordinators d_1 and d_2 of transaction T1 release partial results which are required by transaction T2 and are received by T2 via the data-oriented coordinator d_3.

This means that there is a dependency between (part of) the two transactions. Indeed, transaction T2 uses data items released by T1, and if for some reason the results it received were inconsistent we need to keep track of what parts of T2 have used these results and take that into account in rolling back the system. Exactly how this is done, and what implications it has on concurrency control and the recovery routine will be described in detail in Section 3.4. At the moment, it suffices to understand that each time data is exchanged between transactions (before commit) an entry is added to the EDG which keeps the log structures necessary for applying recovery management.

The EDG of Figure 3-13 says that the data-oriented coordinator d_3 of transaction T2 expects some data from the d_1 of transaction T1, and failing that, it expects results from d_2 of the same transaction. That is all the data-oriented coordinator d_3 needs to know.

Now, if for some reason d_1 (or any other subtransactions on which d_1 depends, for that matter) was aborted, then d_3 should also be aborted along with any sub-transactions of T2 which depend on it. Based on the log information provided by the EDG and the corresponding transaction trees, we would like to recalculate d_3 based on the data items released by d_2 and defer from aborting (at least part of) transaction T2.

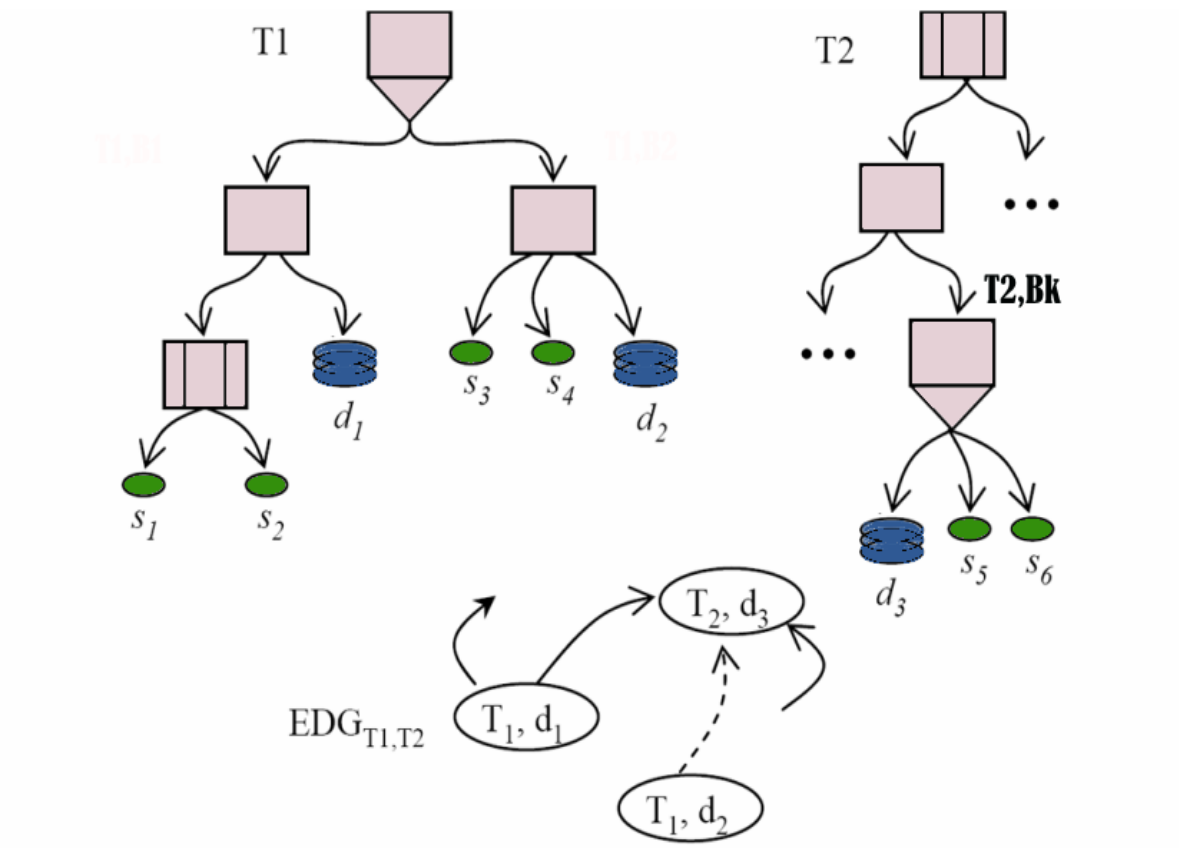


Figure 3-13 EDG for releasing partial results between transactions T1 and T2

Again, it is important to note that each platform or local coordinator only needs to know about services or data-oriented coordinators ('leaves' in general) that are dependent on its own children. It does not need to know about the rest of the transaction or the other transactions that may engage in partial results. Hence, the principle of *before* and *after* applies even in the case of external dependencies (across transactions).

For example, the data-oriented coordinator d_3 only needs to know that it expects data from d_1 of T1, or from d_2 of T2. Notice that in our example (see Figure 3-13), it does not even need to know about *after*, hence it does not need to know of s_5, since they are children of a sequential alternative coordinator.

We have seen that managing the dependencies within and between long-running transactions is an integral part of a transaction model if this is to provide consistency at all times and enhanced recovery management (these aspects will be further discussed in the following section). The thorough understanding of the behaviour exhibited by a transaction can ease the design, deployment and testing and formal analysis of this behaviour can increase the expectation of a successful outcome.

Work is in progress on laying the foundations for a formal model of long-running transactions describing how the sub-transactions (local coordinators and/or basic services) are orchestrated to achieve the goal of the transaction in question. Preliminary ideas on the formal description of a transaction within a service-oriented architecture can be found in [RMK07a]. The proposed formal model draws upon ideas on a truly-concurrent framework for describing interactions between communicating entities [Shi97], [Mos05] and is adopted to underpin the local coordination and compensation mechanism required for long-running multi-service transactions in a dynamic and highly concurrent environment such as a digital ecosystem. This work shall be reported in more detail in the next deliverable (D3.2) but we will have more to say about formal models for compensatable transactions in the concluding section.

3.3.3 Compensatory subtransactions

In a digital ecosystem for businesses (SMEs) a number of long-running multi-service transactions is expected to take place. It is also to be expected that some subtransaction may be aborted and this may be for a variety of reasons; to name a few, in a parallel alternative coordinator some subtransaction reaches the preset condition and the remaining alternative subtransactions need to be aborted, or the communication between two coordinators is lost during a long-running transaction, or due to platform failure some local coordinator does not respond, or a platform itself takes a decision to abort its subtransactions due to some internal problem such as deadlock of the underlying service compositions.

Considering the fully distributed nature of a digital ecosystem for business, different platforms or different services are offered by different service providers, and thus the abortion or failure in some part of the distributed long-running transaction may have financial implications. One of the important requirements for a transaction model is the ability to compensate, in other words to ‘undo’ or take reverse action, in cases where a subtransaction fails or is aborted.

Conventionally, when a subtransaction fails then the coordinator is aborted, which means that all other subtransactions must also be aborted. This normally happens before the second phase of commitment and the process goes on until every action on data is undone and all data used during that transaction is returned to the original values. This often results in a chain of roll backs which may be costly, in terms of time, processing power and unnecessary overheads. The situation gets more complicated when the transaction model allows for partial results, which create external dependencies on other transactions. This is because subtransactions of the other transactions that have used the partial results also need to be aborted. The process of roll back needs to be applied recursively across transactions. In addition, there may be intermediate results in those transactions that have not used partial results and these operations should not be performed again (to avoid huge overheads) in the event of re-starting the corresponding transaction.

It transpires that providing an efficient and consistent compensation mechanism is a major challenge in distributed long-running transactions. Starting from the dependencies both within and between transactions can give a handle for harnessing the complexity of this task.

Compensation within our distributed transaction model is given in terms of virtual *compensatory* transactions. These are generated using both the IDG and the EDG of the corresponding transactions and guide the required reverse operations on the local coordinators.

Creating a compensation tree by using IDG and EDG

A compensatory transaction is created whenever a subtransaction is aborted. We start by looking at the IDG of the failed subtransaction and any other subtransactions that appear within it are added as children of the root node. This covers subtransactions of the same transaction that are directly dependent on the failed one. At the same time, we look at the EDG, if any, associated with the failed subtransaction and add any

subtransaction that appears within it. This covers the subtransactions of other transactions that have used results from the failed subtransaction (partial results).

We then follow the IDGs of these subtransactions to identify their dependent subtransactions (other subtransactions that may have used partial results indirectly). These are added as children of the node of the corresponding compensatory subtransaction. Similarly, for the subtransactions of the same transaction (as the failed subtransaction) we trace their IDG to identify subtransactions that have used inconsistent results indirectly (and are thus also dependent). These are added as children of the node of the corresponding compensatory subtransaction. If a subtransaction occurs twice in creating the compensating tree, then it will be skipped from the second time onwards.

To sum up, we start with the aborted subtransaction as the root of the tree and its children are given by the the corresponding IDG (for dependent subtransactions within the same transaction) and the EDG (for dependent subtransactions in other transactions, via partial results). For each child of the root, we look at the corresponding IDG and EDG to identify subtransactions that are indirectly dependent on the aborted subtransaction. The process, depicted in Figure 3-14, is applied until we have gone through all associated IDGs and EDGs.

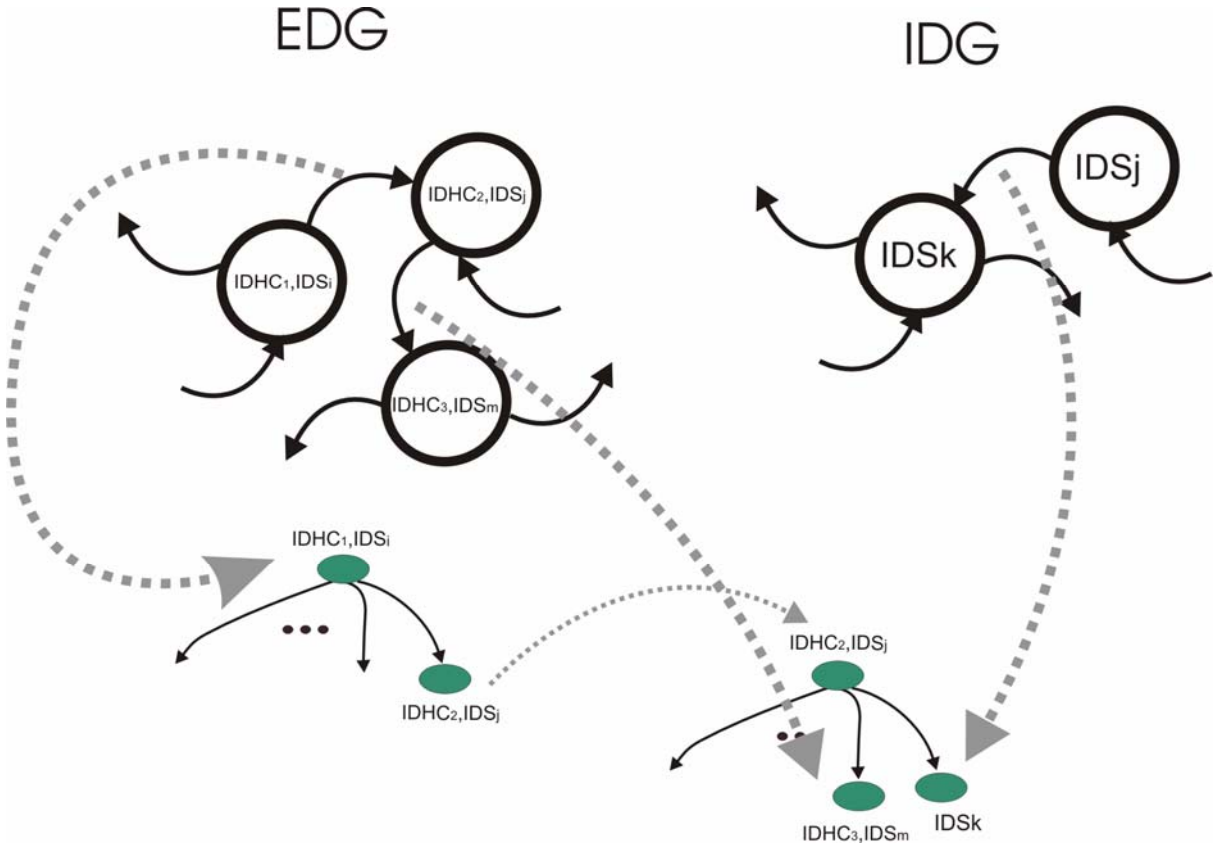


Figure 3-14 Creating compensating routines by using EDG and IDG

It can be seen that the IDG and EDG lie at the heart of recovery management within the proposed transaction model. This will be highlighted in the following section where we discuss an extended lock mechanism that drives the creation of these graphs.

The IDG and EDG allow for local coordination of transactions in a fully distributed manner as they provide a means of recording important system logs which can be stored locally, on the corresponding local

coordinator, but their effect is both local, in terms of local faults, forward recovery and contingency plans, and global, in terms of abortion, restarting, recalculating, and alternative execution. Some of these aspects are discussed in detail in the following section where we turn our attention to recovery management within our transaction model and show how this is interrelated with concurrency control within our overall framework for distributed long-running transactions.

3.4 Hybrid model for concurrency control and recovery management

In the previous section we have described a framework for the local coordination of distributed transactions. The log structures provided by the IDG and EDG keep track of dependencies between the local coordinators in a way that allows each platform to only know about what happens immediately before and after its services are executed. In a certain important sense, the IDG and EDG provide a distributed mechanism for coordination and log-based recovery management.

In this section we turn our attention to the internal structure of the local coordinators and examine the all important aspects of concurrency and recoverability within the distributed transaction framework. In particular, we describe a lock mechanism which extends the conventional S/X Lock model in order to allow for exchange of data and results both within and between transactions (partial results). Further, we will see that the proposed extension to the transaction model with locks allows for traceable transactions. This has the implication that we can guarantee consistency of the model at all times and work with a much more sophisticated recovery mechanism. This includes the preservation of as much progress-to-date as possible (omitted results) and adding diversity into the framework by means of alternative scenarios for completing a transaction in case of failure (forward recovery).

In conventional S/X locks models [Dat96], [Gra03], a data item that is locked by a Shared lock (S_Lock) can be accessed by other transactions or subtransactions of the same transaction. Whilst an operation is performed on a data item then this item is locked by an eXclusive lock (X_Lock) and access is restricted to the owner of the item. Following an S/X lock model for transactions, the X_lock is only released (converted to S_Lock) when the transaction commits. This does not allow for any exchange of data *during* a transaction (before 'commit'). We have seen (Section 3.1) that transactions within a digital ecosystem for business are *long-running* in nature and include the execution of a number of services (from different providers). This means that a transaction in this setting can take minutes or hours or even days to complete ('commit').

It transpires that applying a conventional S/X lock model for concurrency control and consistency in this transactional environment brings unacceptable limitations. It would not allow for exchange of results both within and between transactions, and this severely limits the range of business scenarios that can be covered with distributed transactions in the digital ecosystem. For instance, subtransactions of one transaction cannot access data produced by other subtransactions of the same transaction. Hence, releasing data within a transaction is prohibited.. Further, subtransactions of different transactions cannot exchange results before the transaction they belong to commits. This is often referred to as the problem of *partial results* and appears in most business scenarios in a B2B context.

In view of such limitations of conventional lock models for a transactional environment, we propose an extended lock system which has been designed with distributed transactions in a DE for business in mind. The idea is to relax the rigid eXclusive lock (X_Lock) by introducing intermediate locks that give leverage over access to data items within and between transactions. The additional locks drive the creation of the log-based graphs IDG and EDG, introduced in Section 3.3.2. The extended lock system allows for the concurrent execution inside a transaction but also across transactions and as a result maximises potential concurrency in our transaction model. Additionally, it has a significant role to play in ensuring consistency

of the transaction model at all times, especially when it comes to recovery management, where the locks are intrinsic to the forward recovery mechanism and the way we deal with omitted results.

In what follows we describe our extended lock mechanism for concurrency control and recovery management in distributed transactions. We start by considering the release of data within a transaction and then address the issue of partial results (releasing results between different transactions before 'commit'). Next, we describe the overall jest of recovery management in our transactional framework. We discuss the issue of isolation in the proposed recovery routine and the way we deal with failure propagation on the local coordinators. We show how in some cases we can provide for optimised recovery of data items within the proposed framework. This brings about the issue of omitted results and we show our approach towards preserving as much progress-to-date as possible. Finally, we discuss the additional locks in view of forward recovery and show their importance in adding diversity to the system.

3.4.1 Releasing data within a transaction: Internal lock

In most B2B scenarios, subtransactions of a long-running multi-service transaction need to access and share results before the transactions commits. The use of X_Lock on data items inside a transaction implies that these can only be accessed by the owner of the lock. In our approach, we use an *Internal* lock, denoted by I_Lock, to relax X_Lock and allow subtransactions to access and/or modify data items inside a transaction before 'commit'.

The introduction of this intermediate lock, I_Lock, in between S_Lock and X_Lock as shown in Figure 3-15 makes deployed data items available to other subtransactions of the same transaction.

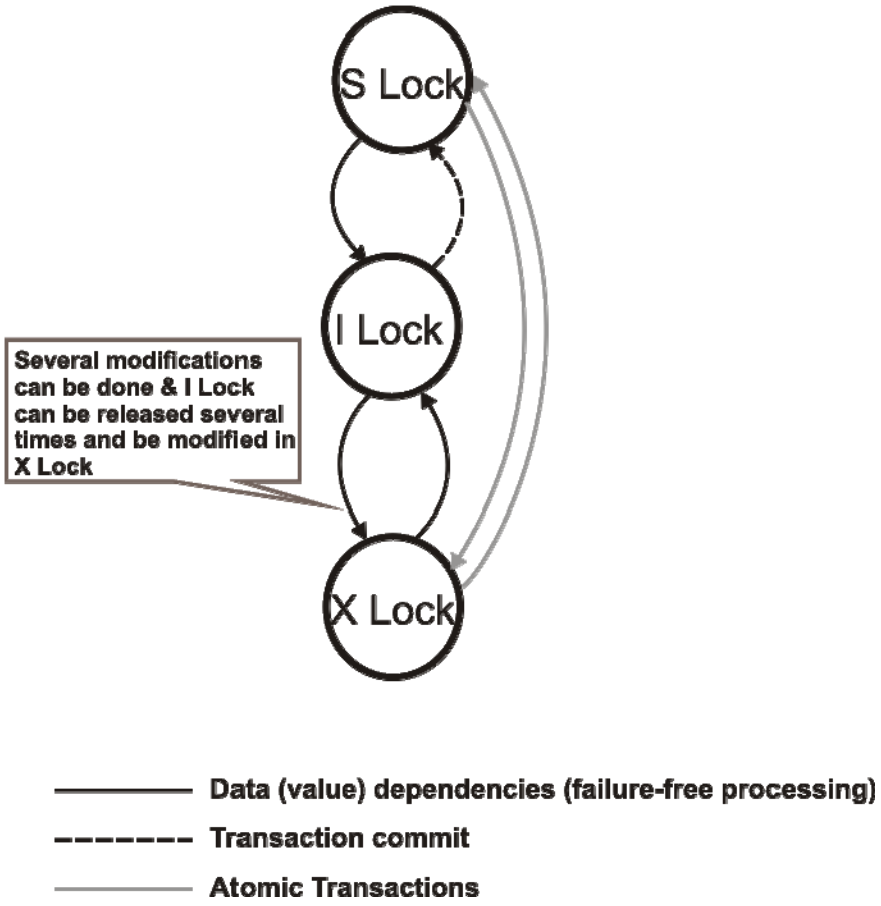


Figure 3-15 Internal lock (I_Lock) schema

When a subtransaction needs to release a result before the transaction it belongs to commits, then it uses I_Lock as a relaxed version of X_Lock. An item that is locked by the internal lock I_Lock can be accessed and modified by other subtransactions of the same transaction. Any subtransaction that wants to use it can do so, provided it adheres to the concurrency control principle and converts I_Lock to X_Lock while it does so (this is necessary to ensure consistency at all times), and then converts it back to I_Lock. The data item is then again available to other subtransactions that might require it.

Every time a subtransaction makes use of the internal lock I_Lock on a data item or uncommitted result, an entry is added to the corresponding IDG which keeps track of dependencies within a transaction. The information required from the subtransaction using I_Lock is the id of the main transaction (IDT), the id of the parent coordinator (IDSh) and the id of the subtransaction itself (IDS). This is necessary in order to keep track of which subtransaction and local coordinator has used what, and from what other coordinator of the transaction, in case a subtransaction fails or is aborted. Recovery in such cases will be discussed in the sequel, but it is worth pointing out that such dependencies are critical for implementing an effective recovery routine.

In this way, the internal lock in combination with the corresponding IDG, allow for uncommitted results inside a transaction to be accessed and modified a number of times and for the duration of the transaction. This mechanism facilitates the concurrent execution of different parts within a transaction. Following a conventional S/X lock model, each subtransaction (in fact, it would only be a transaction as there it would not make sense to have internal structure) would have to execute in turn and commit before releasing tis result to other subtransactions. On the contrary, in our approach the various subtransactions can be potentially executed in parallel and only need to be synchronised on releasing/sharing data items before commit. All data items locked by I_Lock in our model are converted back to S_Lock only when the transaction commits. This means that the results of the transaction can then be used by other (different) transactions that require them.

3.4.2 Releasing data across transactions (before commit): Conditional-commit lock

In distributed transactions within a B2B context of a digital ecosystem, it is often the case that transactions need to release results to other transactions before the transactions involved commit. Following a conventional S/X lock model such cases cannot be addressed as in these models a transaction can only release results after ‘commit’. This limitation not only prohibits transactions from being executed concurrently (since, at best, the second has to wait for the first transaction to commit), but also can prevent a number of different transactions from reaching their target and thus also abruptly stop the corresponding long-lived business activity.

We have seen in Section 3.3 that our transaction model for digital business ecosystems includes a data-oriented coordinator which has been considered particularly for such cases, i.e. to allow for data from a transaction to be released to a subtransaction of another transaction before either transaction commits. This situation is known as the problem of *partial results*. An additional mechanism in concurrency control within our framework for distributed transactions is required in order to address this issue in a consistent manner.

The approach taken for partial results in our framework is similar to that of introducing the internal lock for releasing results within a transaction, described in the previous section. In particular, we use a *Conditional-commit* lock, denoted by C_Lock, which in combination with the corresponding EDG can

provides a safe mechanism for releasing partial results to a subtransaction of another transaction before commit. The lock for this purpose is termed conditional-commit lock to reflect the fact that it is to be applied to interim results of a long-running transaction, in the sense that these results are produced by parts of a transaction (subtransaction as defined by a local coordinator) and are to be released outside the transaction before the transaction as a whole commits.

When a transaction wants to release a data item before commit it uses C_Lock on it. In fact, it converts the item in question from X_Lock to C_lock as shown in Figure 3-16.The C_Lock-ed data item is available to subtransactions of another transaction, but anything that uses it must update the corresponding EDG. The information required for adding an entry to the EDG is the id of the transaction (IDT) and the id of the subtransaction (IDS). Notice that there is no need to keep information of the parent explicitly since we are now concerned with dependencies between transactions. The dependencies within a transaction, for which we would need the parent coordinator, is kept in the corresponding IDG as discussed in the previous section. The data item that is locked by C_Lock is released from a data-oriented coordinator of one transaction to a data-oriented coordinator of the other. In case of failure, anything else that used it has to be rolled back too. This information is given by the corresponding EDG (for moving across transactions) and then following the corresponding IDG of the transaction we arrived at (for tracking the internal dependencies within the transaction that used a partial result).

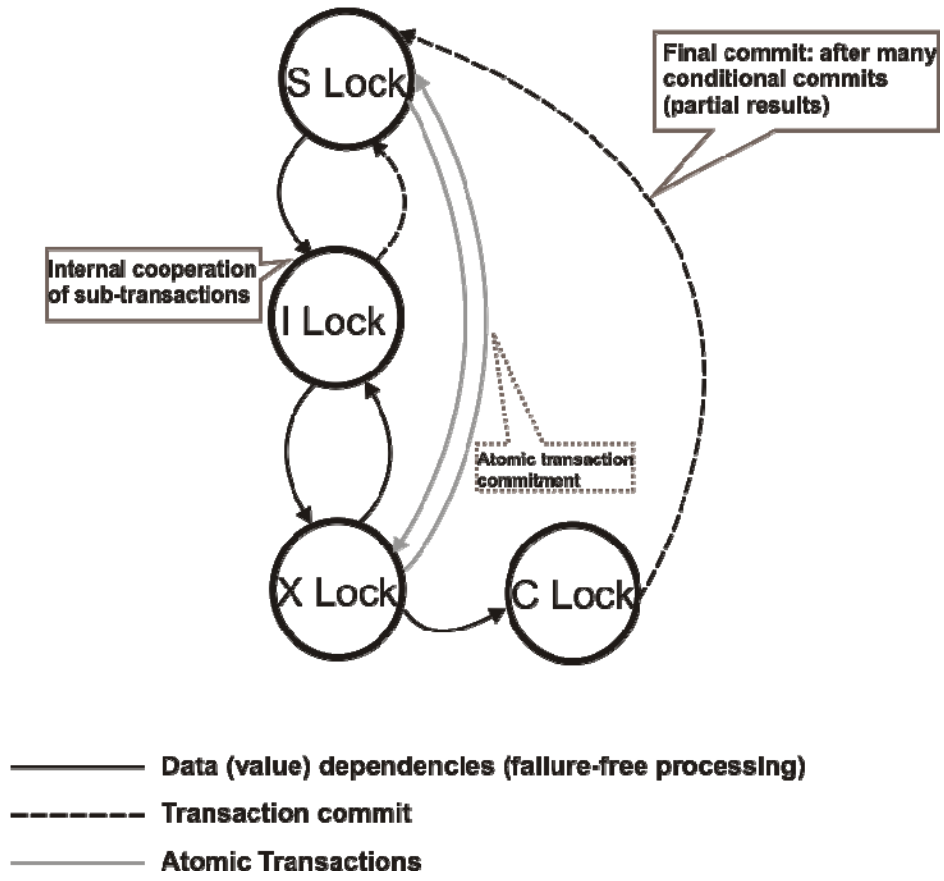


Figure 3-16 Conditional-commit lock (C_Lock) schema

In further explanation, a data item that is to be released before the transaction commits is converted from X_Lock to C_Lock. (Note that in case the data item is to be used internally only, within the same transaction, then it is converted to I_Lock as described in the previous section.) While the data item is in

C_Lock, other transactions can use it, provided they update the corresponding EDG, as explained before. When the transaction commits, the C_Lock is converted back to S_Lock. This has the immediate effect that its dependent transactions, that is, other transactions that have used partial results of this transaction that just committed, are now free to proceed with their own commit.

It can be seen that the introduction of the conditional-commit lock allows for the concurrent execution of different transactions (and their respective subtransactions). Indeed, a transaction that expects some result from another transaction does not have to wait until that transaction commits. It is given access to the C_Lock-ed items and can proceed with its execution while the other transaction that provided these results has not yet committed. Considering that transactions are long-term in nature in this context, the provision for concurrency can have significant benefits.

It might also be worth noting that the mechanism given earlier for releasing data within a transaction is still in use while the C_Lock is applied to data items that are to be released to other transactions before the transaction commits and everything goes back to S_Lock. In other words, the sharing of results internally continues until some result is released outside the transaction, in which case it is C_Lock-ed. Notice that there is no arrow from C_Lock back to I_Lock in the lock schema of Figure 3-16. Results released to other transactions cannot be further modified as this would compromise consistency and integrity of data within the model. It is still possible for other results produced before the transaction commits however, to be released to other transactions or other data items to be accessed/modified by subtransactions of the same transaction.

3.4.3 Recovery without failure propagation: Recovery lock

Recovery management deals with the possibility of failure of (some part of) a transaction and is an integral part of any transaction model. By large, the objective is to ensure consistency at all times, even in the face of failure, and then attempt to avoid abortion of the whole transaction as this may have a knock-on effect that may ground the corresponding long-lived business activity. We have seen that a transaction has structure which comprises a number of subtransactions which are in turn realised by the underlying service compositions. This makes recovery management in a transactional environment rather challenging.

A DE for business comes with specific challenges of its own which may not be so critical in other transactional environments, but have to be taken into account in designing effective recovery management in this setting. One of the most important differences has to do with the purely distributed nature of a DE and the participation of SMEs. The implication is that we cannot afford a central point for managing the recovery procedure and thus the model is driven towards a distributed mechanism which should not only handle but also predict, wherever possible, failures.

For example, consider the case that communication is lost between two local coordinators. This leads to a failure of the underlying service executions. A proper recovery routine needs to be in place to ensure that any effects of the transaction (up to the moment that failure occurred) on the environment are rolled back (are effectively undone). A side-effect of such a situation is that a large part of the system may have already used or is about to use inconsistent results due to the high-speed of failure propagation in such a highly dynamic and purely distributed environment.

There are two well-known methods for designing recovery management [Dat96]: the shadow-paged approach and the log-based approach. As mentioned before, the proposed model is fully distributed and this makes the shadow-paged approach unsuitable due to the range of global overheads incurred [vdMDD+03]. Therefore, we have opted for a log-based approach. In fact, the structure of the proposed recovery management system has strong similarities with a log-based model.

We have seen that two types of information may be released before a transaction commits, which make different demands on the recovery manager. The first type is released results within a transaction (shared between its subtransactions) while the second type is released results between different transactions (partial results). For the former, we have introduced an internal log with a graph structure, given by the Internal Dependency graph (IDG), that determines the internal dependencies for the recovery routine. For the latter, we have introduced an external (though it is kept locally) log with a graph structure, given by the External Dependency Graph (EDG), that indicates the dependencies between different transactions in performing a long-lived business activity. These will be discussed in greater detail in the following sections.

Conventionally, the creation of the appropriate graphs, the order of execution of the recovery manager as well as the routines for releasing results both within and between transactions is considered to be the responsibility of the concurrency control mechanism of a transaction model. However, the analysis of the requirements of a Digital Business Ecosystem [RKM07], and especially dynamicity and its highly unpredictable nature, has shown that merging the two can provide significant benefits with regard to adding diversity (forward recovery) and addressing omitted results. Such aspects will also be discussed in the sequel. Therefore, we have opted for a hybrid model for concurrency control and recovery management within the proposed transaction model for OPAALS.

The design of the recovery routine, together with concurrency control, has been inspired by common strategies of public health authorities for containing the spread of a contagious disease which rely on quarantine, isolation and vaccination [USD07]. A similar mechanism can be seen at work in the isolation of enzymes from an infected tissue which provides the basis for the biochemical diagnosis of an infectious disease. The design of our recovery routine and concurrency control mechanism in our framework reflects this basic concept and is run in three phases:

1. Stop / Halt the system – prohibit any further progress of the transactions ("**quarantine**")

2. Preparation phase: Determine the damaged part – the failed transaction and all its dependent transactions ("**isolate**")

3. Atomic Recovery Transaction Routine: rollback the system to a consistent virtual check point – undo up to that part of the transaction tree in which the local coordinator remains affected as that coordinator may take the transaction to the next step ("**vaccinate**")

In the first phase, a message (abort/restart) is sent to all subtransactions that puts them and their data in *quarantine*. In the second phase or Preparation phase, the part of the system that has been involved in failure is localised; this includes the transaction in which failure occurred as well as any other transactions that have used results from it (via exchange of partial results). This phase puts the system in recovery mode and targets the avoidance of any propagation of inconsistent data (before the rollback procedure is applied). The third phase or Atomic Transaction Recovery Routine, is where the recovery routine is run as an atomic procedure to rollback the system to a consistent virtual checkpoint. Note that we talk about 'virtual' checkpoint as our system does not use actual checkpoints as such, but rather uses the log structures (as given by the IDG, EDG which were created by the locks applied during the transaction) of the local coordinators – this allows to rollback the system as far as necessary but no more. Each subtransaction is rolled back or simply by-passed, depending on whether it has used inconsistent results or not, and this is determined by the corresponding EDG (which takes us across transactions) and then following the corresponding IDG (for identifying any further dependencies within a given transaction).

Isolation in recovery

In previous sections we have introduced locks that allow the release of data within a transaction (I_Lock) and between different transactions (C_Lock). We now discuss these locks in view of the recovery routine within our framework for distributed transaction in DEs.

Data items that have been locked by I_Lock can only be used internally, can only be accessed/modified by subtransactions of the same transaction, and this is recorded in the corresponding IDG as explained in Section 3.4.1. So these items can be considered atomic and can thus be rolled back (wherever necessary) based on the corresponding IDG. Recall that every subtransaction that has used/accessed such an item has added an entry to the IDG.

Data items that have been locked by C_Lock can be used externally, by other transactions, and this is recorded in the log structure given by the corresponding EDG as explained in Section 3.4.2. So subtransactions of other transactions that have used these items (locked by C_Lock) can be rolled back following the corresponding EDG. Recall that every transaction that has used a partial result from another transaction has done so whilst adding an entry to the EDG. Hence, in case of failure, for C_Lock-ed items we look at the EDG which takes across to other transactions that have used these items, and within that transaction we then follow the IDG to see what other part (subtransaction) of this transaction has made use of the inconsistent data items. This ensures that the whole spectrum of dependent subtransactions is identified by the recovery routine.

It transpires that the part of the transaction in which failure occurred as well as all related or dependent subtransactions of other transactions need to be marked as soon as possible and isolated so as to avoid costly failure propagation. In particular, the rollback for C_Lock items, due to the external dependencies incurred, can result in chains of rollback which is costly in terms of time, but also in terms of distributed accounting which another part of OPAALS (WP4) is currently looking at.

In order to ensure that the damaged part of a long-running transaction is isolated as soon as possible we introduce a *Recovery* lock, denoted by R_Lock, whose immediate effect is to restrict data to the recovery routine only. The idea is that, in the first instance, all locks are converted to R_Lock until further notice, as shown in Figure 3-17.

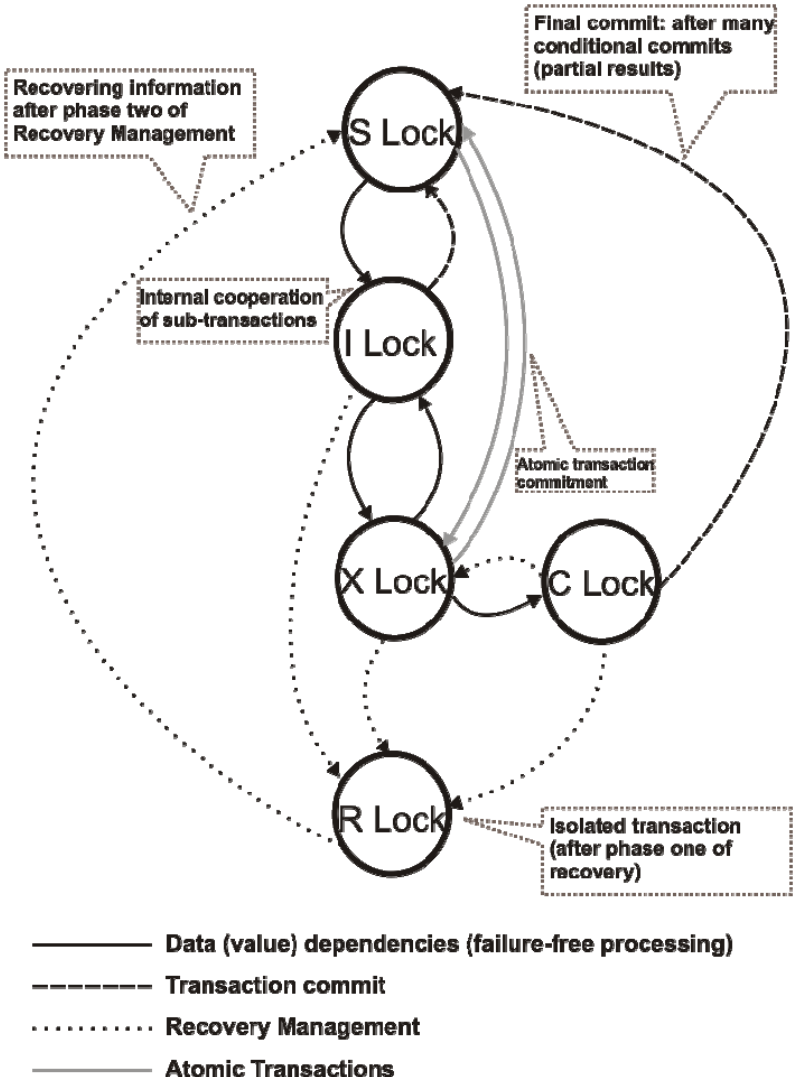


Figure 3-17 Recovery lock (R_Lock) schema

In particular, the items locked by C_Lock are converted to R_Lock based on the EDG (of the damaged part) and then the IDGs (of the affected parts) found in each of the transactions indicated by the EDG. This is done without processing any data. As mentioned before, R_Lock restricts access to data to the recovery routine only. Once the recovery routine applies the rollback, wherever necessary, then the data is returned to the initiator of the transaction, which in the lock schema of Figure 3-17 amounts to converting R_Lock back to S_Lock.

3.4.4 Optimised recovery: Time-out lock

We have seen that data items (results) used internally by other subtransactions of the same transaction are locked by I_Lock. We have also seen that in recovery mode, these items are rolled back as if they were atomic transactions. This is done following the corresponding IDG. In a digital ecosystem for business interactions entail a large number of highly concurrent transactions whose latency may be critical for conducting a business scenario or performing a business activity.

In this section, we are concerned with optimising the recovery routine of data items locked internally using I_Lock, in an attempt to address the dynamicity of the environment more effectively. In particular, we introduce a *Time-out* lock, denoted by T_Lock, which introduces further flexibility in our lock system for distributed transactions.

For optimising the recovery of (I_Lock-ed) data items results we use T_Lock on internally locked items so as to allow for a time-out before rolling them back. The T_Lock restricts access to data to the recovery routine only - just as R_Lock does. But the T_Lock also sets a time-out after which the I_Lock-ed data item will be rolled back automatically. The way T_Lock functions within the context of the overall locking scheme is depicted in the extended lock schema of Figure 3-18.

We have seen that I_Locked items are addressed by the recovery routine in a binary fashion; either rolled back or not, depending on the corresponding IDG. It turns out [RKM07] that in some cases rollback is not necessary whereas in other cases we are certain about failed subtransactions. The diea behind introducing the T_Lock is to give the recovery routine an opportunity to reconvert T_Lock to I_Lock if rollback is not deemed necessary. If no decision is possible within a certain amount of time, then the data item is rolled back automatically.

In this way, safe results, those not affected by the damaged part of the transaction, do not need to be rolled back or re-computed in case the transaction needs to be re-started. Clearly, there is a relation between T_Lock and the handling of omitted results and this should become more clear in the following section.

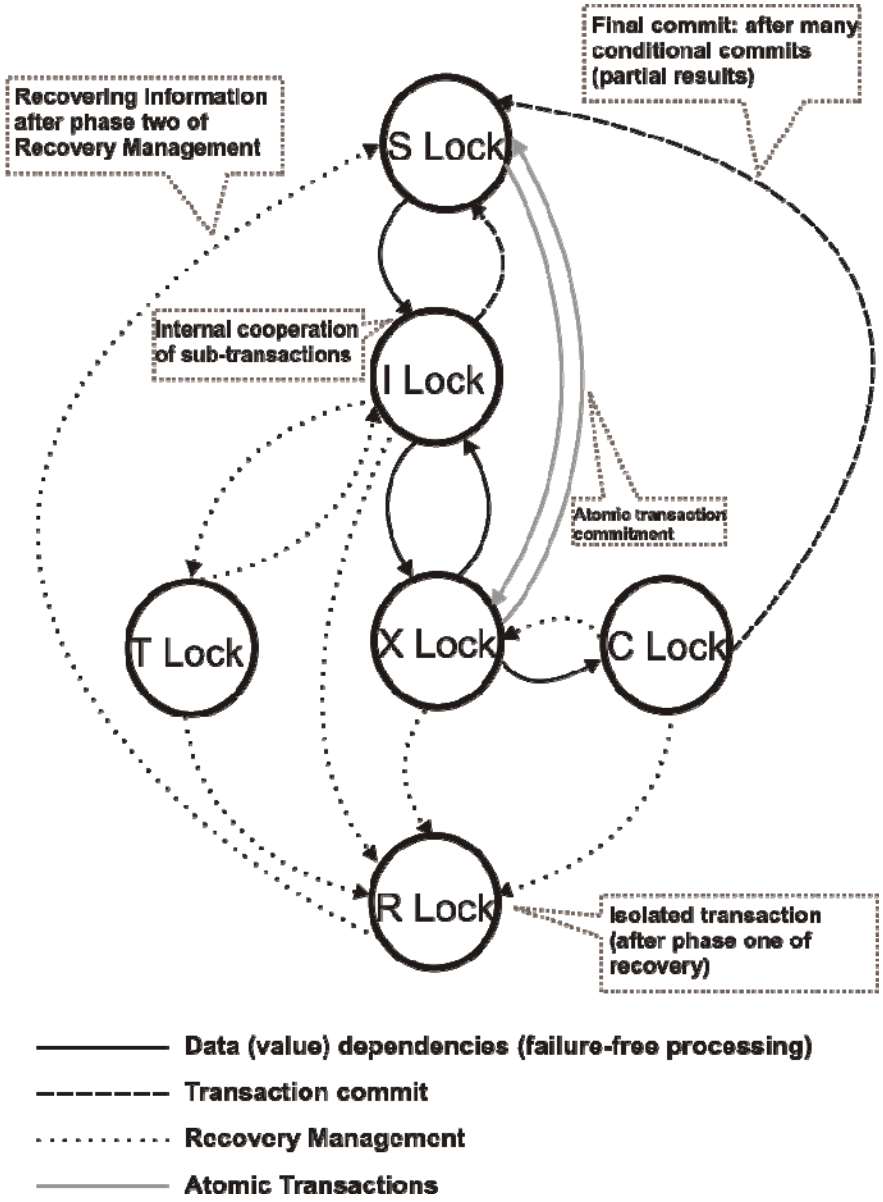


Figure 3-18 Time-out lock (T_Lock) schema

3.4.5 T_Lock and omitted results in recovery management

The framework we have described so far is intended for purely distributed transactions within a digital ecosystem for business, in a B2B context. A service-oriented architecture (SOA) is envisaged as the enabling technology since this computing paradigm considers loosely-couple services that are distributed across several platforms over a P2P network (which is discussed in Chapter 5). We have also described an extended lock system that introduces flexibility and meets the demands of a transaction model for business transactions between SMEs. In this section, we demonstrate the use of the lock system in recovery management and highlight the relation to the P2P network topology required to support this transactional environment.

Let us assume that during a long-running transaction, the connection between the local coordinators is lost. The situation is depicted in Figure 3-19.

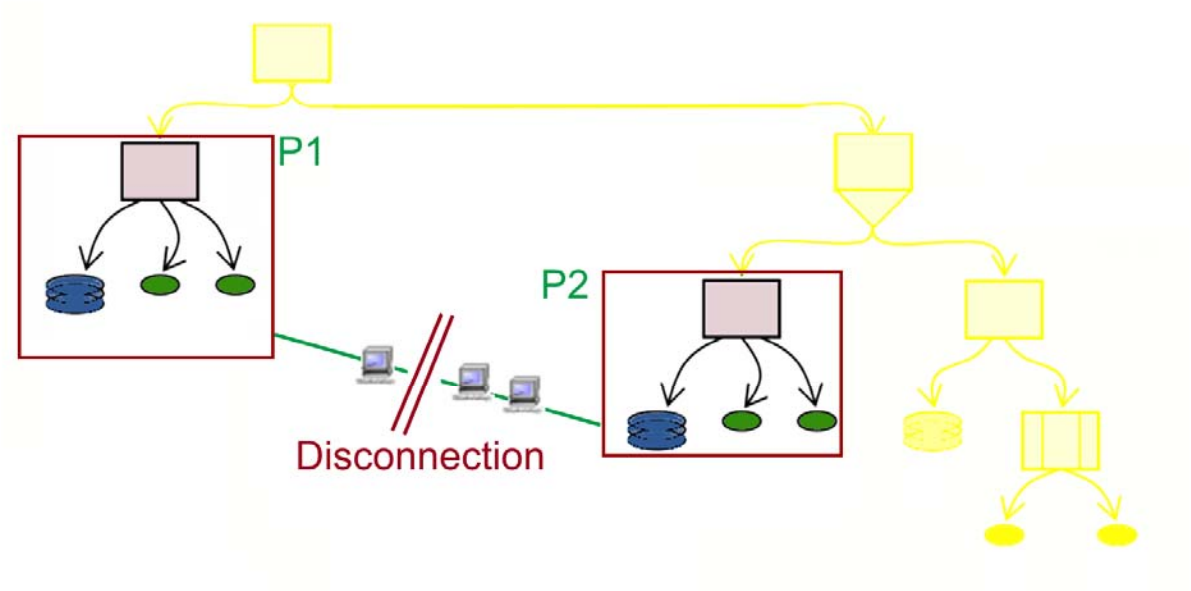


Figure 3-19 Failure due to loss of communication during a long-running transaction

Further assume that within the local platforms P1 and P2, some results need to be shared (between subtransactions). In our framework, this is possible by using I_Lock on such data items.

Now in the event of disconnection, say, the link between platform P1 and P2 is lost, the long-running transaction does not necessarily have to be aborted as a whole in our framework, as would be the case with conventional S/X lock models (and current transaction models). The objective in this case is to ensure that costly (in terms of time, network usage, execution of underlying services, etc.) results within each part of the long-running transactions are not lost. The use of the time-out lock T_Lock proves useful in this respect.

Results that have been used internally (within each platform or local coordinator) have been locked using the internal lock I_Lock. Once the communication between the local coordinators is lost, the recovery routine is applied to (the whole of the) transaction. This means that any data items locked by I_Lock are converted to T_Lock, which restricts access to I_Lock-ed items to the recovery routine only, and sets a time-out after which the items will be rolled back automatically. If within the time-out, the recovery routine determines that the items (or some of them) have not used inconsistent data or have not been affected as such by the failure in some other part of the transaction, then the items are converted back to I_Lock. This means that in case the transaction is re-started these data items do not have to be re-computed.

For instance, the connection between the two local coordinators may be established again, within the time-out, or there might be an alternative path through the supporting network for P1 to regain connection with P2. This is described in Figure 3-20 which shows that an alternative path through various other

platforms is readily available and can be used to reconnect P1 and P2. Or, it might be the case that there is another platform, say P3, which is connected to P1 over the P2P network and also has connection to P2.It is important to note that this touches upon the very characteristics of the underlying P2P network such as replication and duplication, which are discussed in Chapter 5.

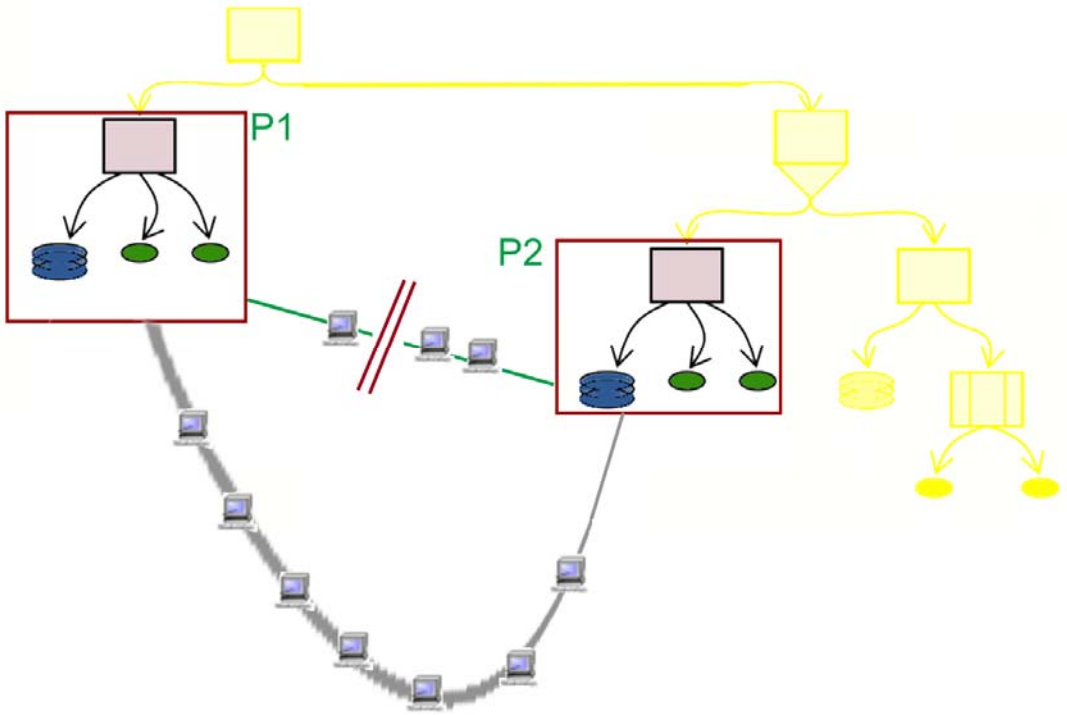


Figure 3-20 Reconnection between local coordinators in a long-running transaction

It can be seen that in case the communication between P1 and P2 is restored (there exists an alternative path and this can be identified) within the pre-set time-out of T_Lock, the long-running transaction does not need to be aborted. The results in each of P1 and P2 will not be lost, in fact hey will be converted back to I_Lock, since the whole transaction will not be aborted.

It is in this sense that the extended lock system for concurrency control and recovery management in our framework for distributed transactions over a P2P network provides additional flexibility and can be used to cover a wider range of B2B scenarios. Furthermore, the design of the transaction model has been based on the principle of preserving local autonomy and thus is well-suited for business transactions between SMEs.

4 Current P2P Designs

The popularity of Peer-to-Peer (P2P) networks grew rapidly within a few months of Napster's introduction in 1999. The system started spreading widely and was quickly recognised as the first sign of a new revolution in the computer science community. P2P network designs received much attention for computer networks and several measurements of data suggest that P2P applications are having a very significant and rapidly growing impact on Internet traffic [GaW01], [Plo00].

Different expectations of the usage of the network itself, limitations of resources, safety, security and other challenges have led designers to introduce various topologies and designs. In this chapter, we try to review the general views and some practical implementations of these networks.

4.1 Network basics

At a fundamental level, following the definition of Paul Baran [Baran and Boehm, 1964], there are three main categories of network: centralised; decentralised; and, distributed. We will review the requirements for OPAALS's P2P network in Chapter 5. However, there are two critically important requirements that must be borne in mind as we review network technologies in this section. Firstly, the need to support long-term transactions imposes strong requirements for consistency in the network. Secondly, we need a network topology that is free of bottlenecks in the messaging traffic. Note also that network traffic will involve both search and maintenance (growth and repair) activity.

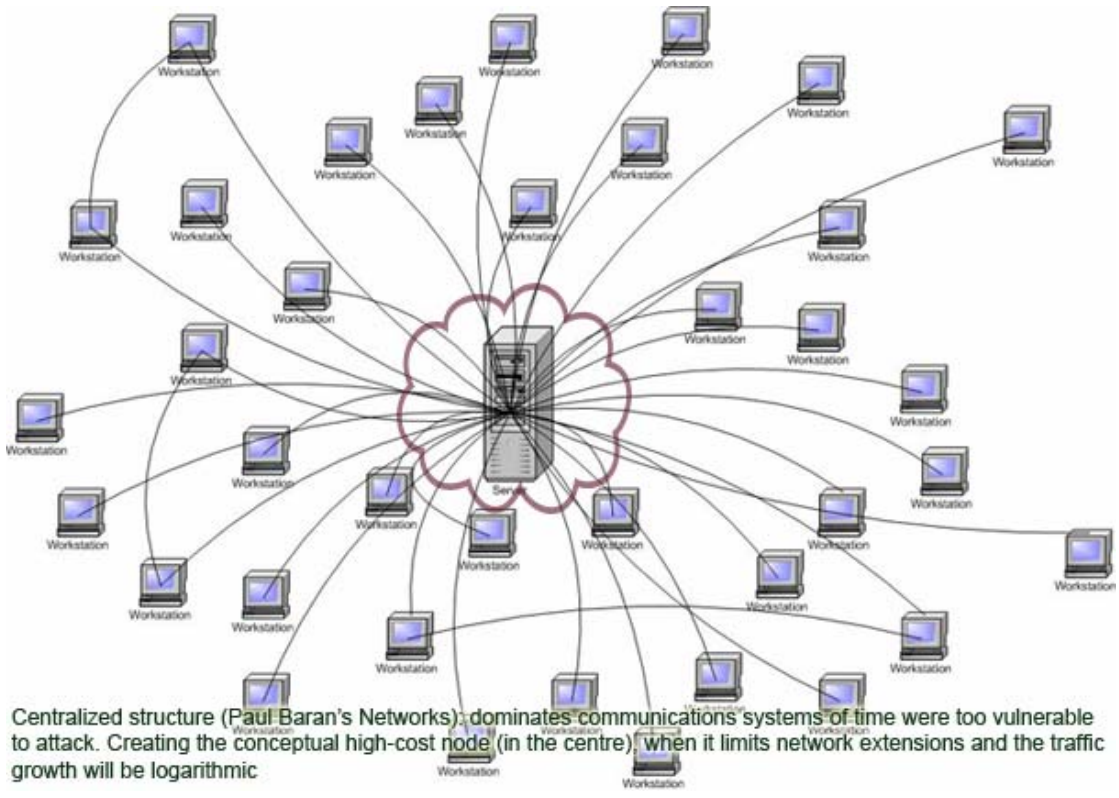


Figure 4-1 Centralised network – single central node

In the case of a *centralised* network (Figure 4-1) there is a central node to which every other node is connected. In this topology, as the network grows the central node has to increase its bandwidth. This not only increases the cost of that node, but also places an upper bound on the performance of the whole network. Thus the whole network, and consistency of its data, is dependent on a single highly strewed point of failure. This applies whatever kind of service the central server provides, even if it is "just" a central registry.

A centralised network is the result of applying very restrictive rules and limitations to the control of, and services provided by, the network. These rules and limitations come from the network's topology and restricted resources. The first step towards solving some of the problems arising was the move to a *decentralised* network.

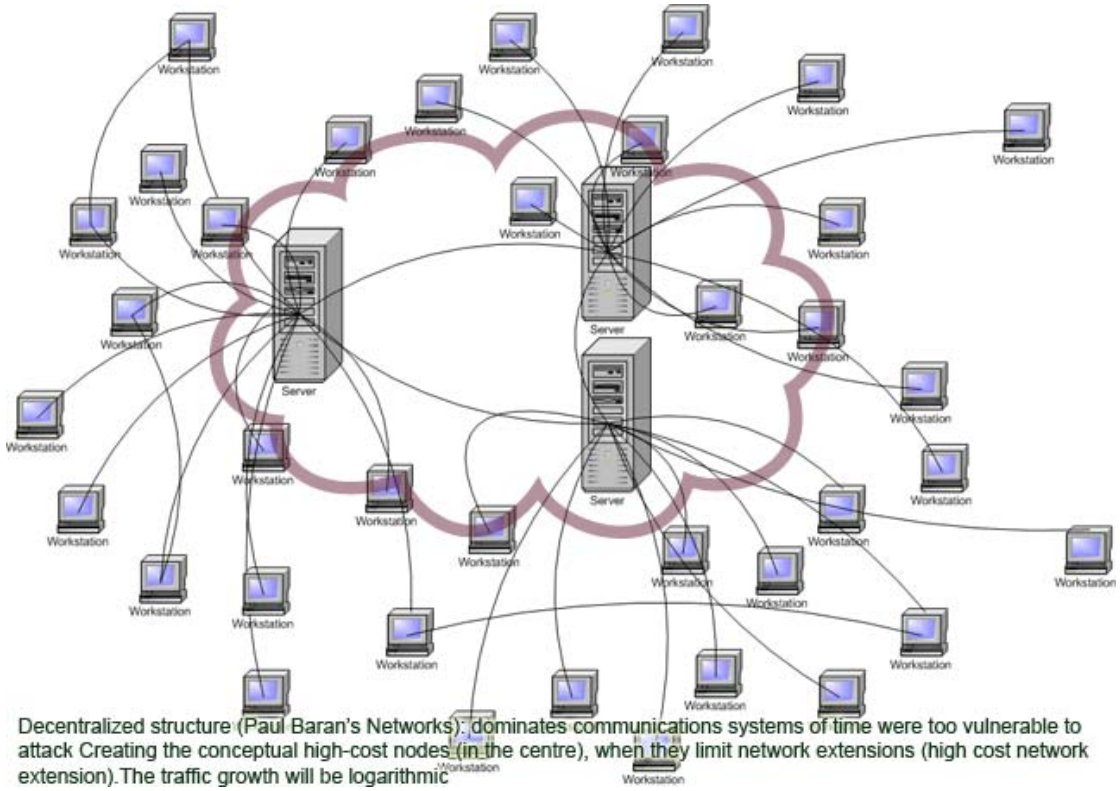


Figure 4-2 Decentralised network – limited number of central nodes

A decentralised network tries to divide the services and responsibilities across several nodes (in contrast of a central node). But a decentralised network can only improve the situation slightly, because the server nodes together generate conceptually a centralized network with essentially the same traffic problems. Pressure to increase the number of server nodes (and/or bandwidth) as the network grows is the main bottleneck. In addition, in the context of OPAALS and DBE, both centralised and decentralised networks raise concerns over the ownership of the server nodes, *even if the responsibility of the server nodes is limited to registry services.*

This historical problem has been known since 1964 [Baran and Boehm1964]. As part of a solution, Baran envisioned a network of unmanned nodes that would act as switches, routing information from one node to another onward to their final destinations. The nodes would use a scheme Baran called "hot-potato routing" as a method of distributed communication (see Figure 4-3). Their 'Cost estimation of network' is

considered as an oracle for improving mathematical models of network topologies (e.g. see [Bar05] which contains links to online copies of Baran's work).

On a different level, the movement from decentralised networks to *distributed* networks was one of the main benefits from the creation of the Internet [GiC00]. Companies such as Yahoo, Google, Amazon, and Ebay very quickly responded to the potential of the internet [Bar03], and tried to exploit it by designing and implementing their own distributed network models.

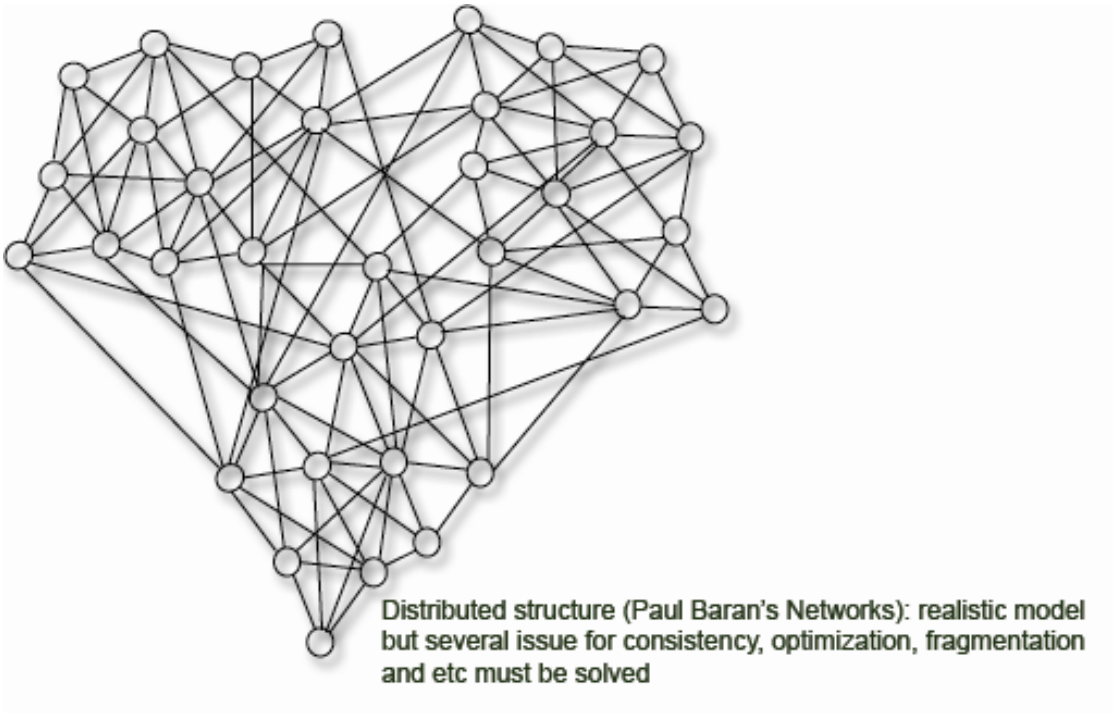


Figure 4-3 Distributed network – no central node

Several issues must be discussed and solved before a distributed network can be used effectively. These include: how to grow the network; how to guard against or repair fragmentations during growth; consistency maintenance; and, performance. In addition, and most importantly for OPAALS, the level and models of distributed transactions which can and must be supported on the network also need to be identified.

The application of specific topologies, rules and restrictions for the customisation of a network to a specific area (for example, meeting requirements for distributed e-Business), the effect of these restrictions on the design of a network, and the protocols and optimisation routines supported by the network are other areas that need to be considered.

4.2 Current P2P network implementations

The Intel P2P working group gave the definition of P2P as

"*The sharing of computer resources and services by direct exchange between systems*"

This thus gives P2P systems two main key characteristics:

- *Scalability*: there is no algorithmic or technical limitation to the size of the system; for example, the complexity of the system should be somewhat constant regardless of the number of nodes in the system.

- *Reliability*: the malfunction on any given node will not affect the whole system (or maybe even any other nodes).

Based on the research of Schmidt and Parashar [ScP03], P2P can be categorized at least into two groups by the type of model: *pure* P2P and *hybrid* P2P. A pure P2P model, such as Gnutella and Freenet, does not have a central server. The hybrid P2P model, such as Napster, Groove, and Magi, employs a central server to obtain meta-information such as the identity of the peer on which the information is stored or to verify security credentials. In a hybrid model, peers always contact a central server before they directly contact other peers. We will refer back to these two categories in the following review.

4.2.1 Napster

In May 1999, Shawn Fanning and Sean Parker created Napster Inc., thus beginning an unforeseen revolution. Napster devised a technology for using peer-to-peer connections to exchange compressed music files (MP3s). Due to the fact that Napster is not an open source application, it was only possible to build up a similar application to reveal the Napster protocol through reverse engineering (OpenNap project; Turcan, 2002; [DNB03]). Most analyses have been based on a reverse engineering exercise and so cannot guarantee that they capture the exact architecture of Napster.

In respect of its underlying centralized directory model, the early Napster (Napster, 2000 [DNB03]) can be viewed as a nearly perfect example of a hybrid P2P system in which a part of the infrastructure functionality, in this case the index service, is provided centrally by a coordinating entity.

The moment a peer logs into the Napster network, the files that the peer has available are registered by the Napster server. When a search request is issued, the Napster server delivers a list of peers that have the desired files available for download. The user can obtain the respective files directly from the peer offering them.

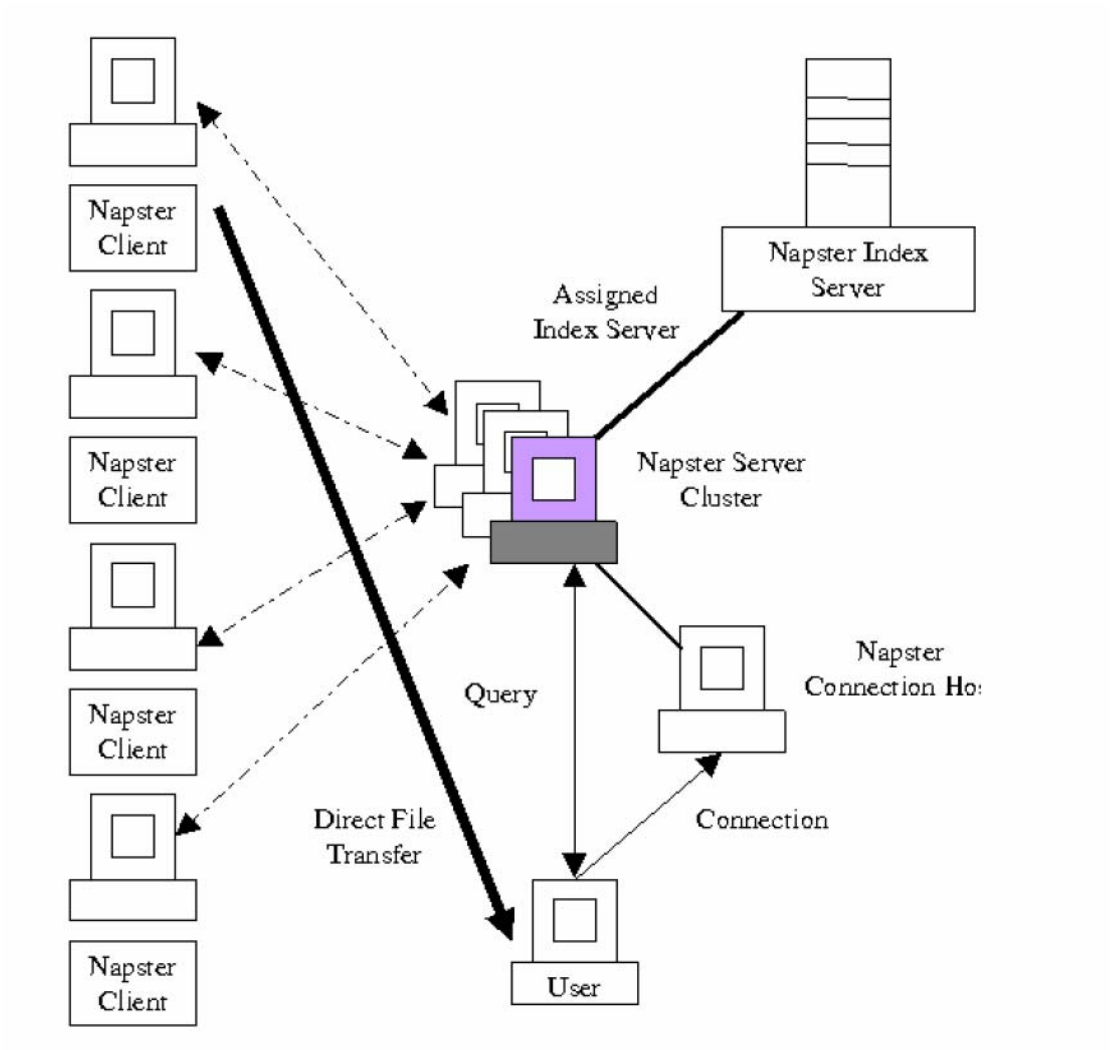


Figure 4-4 High-level design of Napster

In an efficient "Napster" design (Figure 4-4), there is a server-client structure where there is a central server system, which directs traffic between individual registered users. The central servers maintain directories of the shared files stored on the respective PCs of registered users of the network.

These directories are updated every time a user logs on or off the Napster server network. Clients connect automatically to an internally designated "metaserver" that acts as common connection arbiter. This metaserver assigns at random an available, lightly loaded server from one of the clusters.

Servers appear to be clustered about five to a geographical site and Internet feed, and able to handle up to 15,000 users each. The client then registers with the assigned server, providing identity and shared file information for the server's local database. In turn, the client receives information about connected users and available files from the server. Although formally organized around a user directory design, the Napster implementation is very data-centric. The primary directory of users connected to a particular server is only used indirectly, to create file lists of content reported as shared by each node [Bar01].

4.2.2 Gnutella

In early March 2000, Justin Frankel and Tom Pepper, who were both, working under Gnullsoft, created Gnutella which is one of AOL's subsidiaries. AOL halted distribution shortly after it was published, but the short duration where Gnutella was made online was enough to allow interested programmers to download and later reverse engineer Gnutella's communication protocol. As a result, a number of Gnutella clones with improvements were introduced (for example, LimeWire, BearShear, Gnucleus, XoloX, and Shareaza).

P2P networks that are based on the Gnutella protocol function without a central coordination authority. All peers have equal rights within the network. Search requests are routed through the network according to the flooded request model, which means that a search request is passed on to a predetermined number of peers. If they cannot answer the request, they pass it on to other nodes until a predetermined search depth (TTL = time-to-live) has been reached or the requested file has been located. Positive search results are then sent to the requesting entity, which can then download the desired file directly from the entity that is offering it. A detailed description of searches in Gnutella networks, as well as an analysis of the protocol, (can be found in [RLF02] and [Rip01]).

Due to the fact that the effort for the search, measured in messages, increases exponentially with the depth of the search, the inefficiency of simple implementations of this search principle is obvious. In addition, there is no guarantee that a resource will actually be located. Operatingsubject to certain prerequisites (such as non-randomly structured networks), numerous prototypical implementations (for example, [CrP02], [RoD01], [RFH+01], [LCC+02], [ALH02]) demonstrate how searches can be effected more "intelligently" (see, in particular, [DKR02], and also [AbH02] for a brief overview).

4.2.3 FastTrack

The FastTrack network as a hybrid architecture is a cross between centralized and decentralized topologies ([YaG02]). Very little is known of the actual protocol used. Many attempts have been made to reverse engineer the FastTrack protocol. The most well known to date would be the giFT project as they were the closest to finally cracking the protocol. Our review is based on this reverse engineering research ([Ber03]). This technology uses two tiers of control in its network. The first tier is made up of clusters of ordinary nodes that log onto Super Nodes (ordinary machines with high-speed connection). As discussed previously, this sort of connection mimics the centralized topology.

The second tier consists of only Super Nodes that are connected to one another in a decentralized fashion. These Super Nodes are just ordinary nodes which can and will join or leave the network as they please. In order to ensure the constant availability of the network, there exists a need for a dedicated peer (or several of these peers) that will monitor and keep track of the network. Such a peer is called a bootstrapping node ([KuR03]) and it should always be available online.

Resource discovery is accomplished through the act of broadcasting between Super Nodes. When a node from the second tier makes a query, it is first directed to its own Super Node, which will in turn broadcast that same query to all other Super Nodes to which it is currently connected.

This is repeated until the TTL of that query reaches zero. So, if for example, the TTL of a query is set to 7 and the average amount of nodes per Super Node is 10, a FastTrack client is able to search 11 times more nodes on a FastTrack network as compared to Gnutella ([ABC+01]). This provides FastTrack clients with a

much greater coverage and better search results. However, there is an important drawback to such a broadcasting method, and that is the daunting amount of data that needs to be transferred from Super Node to Super Node. This is the very same problem that has been plaguing the Gnutella network.

But by defining Super Nodes as nodes that are guaranteed to have fast connections, it tries to overcome to problem. Each of the Super Nodes that received the query performs a search through its indexed database that contains information of all the files shared by its connected nodes. Once a match is found, a reply will be sent back following the same path the search query was propagated through until it reaches the original node that issued the query.

This method of routing replies is similar to Gnutella, and hence runs the risk of facing the same problem of losing replies as it is routed back through the network. This is due to the fact that the Gnutella network backbone, as mentioned previously, is made up of peers that connect and disconnect from the network very sporadically (One of the most famous P2P networks with FastTrack structure is KaZaa).

4.2.4 OpenFT

Like FastTrack, the OpenFT protocol also classifies the nodes in its network into different roles, but instead of a two-tier control architecture, OpenFT added an extra tier, making it a three-tier control architecture. The classification of nodes is based on the speed of its network connection, its processing power, its memory consumption, and its availability ([ABC+01]).

The first tier is made up of clusters of ordinary machines, which we refer to as User Nodes. These nodes maintain connections to a large set of Search Nodes (ordinary machines with high speed connection). The user nodes then update a subset of the search nodes to which it is connected with information regarding files that are being shared ([DNB03]). The second tier is made up of machines that are referred to as Search Nodes. These nodes are the actual servers in the OpenFT network.

These servers have the responsibility to maintain indices of files that are shared by all the User Nodes under them. On default, a Search Node can manage information about files stored at 500 User Nodes. The third tier is made up of a group that it much smaller, because the requirements to qualify for this group are much more stringent. One has to be a very reliable host that has to be up and available most of the time. These nodes are referred to as Index Nodes as their main purpose is to maintain indices of existing Search Nodes. They also perform tasks such as collecting statistics and monitoring network structure.

4.2.5 Freenet

Freenet ([CSW+00]) was designed to prevent the censorship of documents and to provide anonymity to users. Unlike Gnutella, which uses a breadth-first search (BFS) with depth TTL, Freenet uses a depth-first search (DFS) with a specified depth. Each node forwards the query to a single neighbour and waits for a response from the neighbour before forwarding the query to another neighbour (if the query was unsuccessful) or forwarding the results back to the query source (if the query was successful).

Searching for and storing files within the Freenet network ([CMH+02]) takes place via the so-called document routing model. A significant difference to the models that have been introduced so far is that files are not stored on the hard disk of the peers providing them, but are intentionally stored at other locations in the network.

The reason behind this is that Freenet was developed with the aim of creating a network in which information can be stored and accessed anonymously. Among other things, this requires that the owner of a network node does not know what documents are stored on his/her local hard disk. For this reason, files and peers are allocated unambiguous identification numbers.

When a file is created, it is transmitted, via neighbouring peers, to the peer with the identification number that is numerically closest to the identification number of the file and is stored there. The peers that participate in forwarding the file save the identification number of the file and also note the neighbouring peer to which they have transferred it in a routing table to be used for subsequent search requests.

4.2.6 Groove

The Groove platform provides system services that are required as a foundation for implementing P2P applications. A well-known sample application that utilizes this platform is the P2P Groupware Groove Virtual Office. The platform provides storage, synchronization, connection, security and awareness services.

In addition, it includes a development environment that can be used to create applications or to expand or adapt them. This facilitates the integration of existing infrastructures and applications (such as Web Services or .NET Framework).

Groove Virtual Office is the current best-known application for collaborative work based on the principles of P2P networks. This system offers standard collaboration functions (instant messaging, file sharing, notification, co-browsing, whiteboards, voice conferences, and databases with real-time synchronization) to those of the widely used client/server based Lotus products, Notes, Quickplace, and Sametime, but does not require central data management.

All of the data created are stored on each peer and are synchronized automatically. If peers cannot reach each other directly, there is the option of asynchronous synchronization via a directory and relay server. Groove Virtual Offce offers users the opportunity to set up so-called shared spaces that provide a shared working environment for virtual teams formed on an ad hoc basis, as well as to invite other users to work in these teams.

Groove Virtual Office can be expanded by system developers. A development environment, the Groove Development Kit, is available for this purpose ([Edw02]). Similar system, IBM Sametime Server-based e-Meetings (Sametime) is server oriented. However, the process of knowledge sharing in such systems regardless of their physical architecture is often P2P.

4.2.7 Jabber

The Jabber Open Source Project (http://www.jabber.org/) is aimed at providing added value for users of instant messaging systems. Jabber functions as a converter, providing compatibility between the most frequently used and incompatible instant messaging systems of providers, such as Yahoo, AOL, and MSN. This enables users of the Jabber network to exchange messages and present information with other peers, regardless of which proprietary instant messaging network they actually use. Within the framework of the Jabber-as-Middleware-Initiative, Jabber developers are currently working on a protocol that is aimed at extending the existing person-to-person functionality to person-to-machine and machine-to-machine communication ([Mil01]).

4.2.8 JXTA

JXTA is an open platform that aims at creating a virtual network of various digital devices that can communicate via heterogeneous P2P networks and communities. The specification includes protocols for locating, coordinating, monitoring and the communication between peers (Project JXTA; [Gon01]).

5 A Peer-to-Peer Network for Digital Ecosystems

In Chapter 3 we outlined the transactional model the *autopoietic* P2P network (as a infrastructure of digital business ecosystem network) is intended to support. The full potential of such a transaction model cannot be realised without significant development of the underlying P2P architecture. In this chapter we highlight the areas that need to be addressed in the development of a next generation architecture that supports distributed long-running transactions and evolves to reflect the dynamicity of this service-oriented business environment. We identify the key attributes that are needed for digital ecosystems and then analyse where current P2P models fail in meeting such requirements. We then report on work towards addressing the concerns raised with respect to developing a P2P network architecture that meets digital ecosystems requirements which is currently the focus of our attention.

5.1 Towards a P2P network to support business transactions

We have described a transactional model for supporting critical requirements of an open community of SMEs in digital ecosystems in Chapter 3 of this report. However, fulfilling all these requirements also puts constraints on the design of the underlying Peer-To-Peer network structure. The central problem is that the P2P network itself *must* respect the local autonomy of the participating SMEs. Unfortunately, any dependency on large enterprises for providing hubs on the network and introducing a centralised (or limited decentralised) discovery system such as UDDI, (even if supported by distributed coordination frameworks), introduces two blocks on achieving the goals of a digital ecosystem.

Firstly, this seriously impacts on the ability for small and medium businesses to advertise their services in a fair and computational environment. Secondly, such dependencies fail to provide transparent loosely-coupled binding with large enterprises that is needed to preserve the local autonomy of SMEs. In addition, the current state of technology comes with serious technical problems for the implementation of explorative and semi-fixed service composition. This is simply because these can cause huge amounts of traffic on any centralised registry and as a result the whole structure will rely on few critical hubs.

A key driver behind the *autopoietic* P2P network is that the whole computational infrastructure must serve the needs of the digital ecosystem community as a whole and facilitate the sustainability of the businesses involved. The autopoietic P2P environment must be provided by and for its whole community of users. There must be no opportunity for one organisation, or even a small number of organisations, to be in a position to benefit from, or (intentionally or otherwise) create risks to, the routine business activities of other members of the network.

Consequently:

- there must be no critical dependencies on single organisations
- there must be no critical points of failure
- hence, the environment should be *fully distributed*

The categorised requirements for the *autopoietic* P2P network can be considered as below.

Consistency

As has been mentioned already, a key aspect of the autopoietic P2P environment is that it must be optimized for the needs of business, and not just the sharing of information within a community. Within a single business transaction, there may be a set of sub-transactions involving different organizations. Each sub-transaction may involve a degree of financial risk should the transaction as a whole fail (e.g. if a sub-transaction involves purchase of a flight ticket, it may not be possible to recover part or even all of the ticket cost should a later stage in the transaction lead to failure and roll back).

Hence, the *autopoietic* P2P network must

- support a highly transactional environment which can cause huge amount of traffic
- support long-running transactions which need replication of critical system logs

Latency

One of the most disputable characteristics of the autopoietic P2P can be latency; different views on latency can be confusing and sometimes contradictory on the final design of autopoietic P2P network. But this is a critical impact factor on the performance of the infrastructure in supporting long-running business transactions. On 4.3.3 we briefly try to analyze the latency and its effect on the network.

Local autonomy

Although the autopoietic P2P network as a whole must belong to the community to enable their services to be freely published and discovered, the autonomy of individual nodes within the network must be respected.

Hence:

- There must be no limitation to the ownership or decision making of a single node's services (local node has full control over its own data and services).

Reliability

- Address/manage failure risks for any infrastructural services

Security

- Primary structure must fully support development of a secure transaction model

Availability

- Consistent replication

Boundaries

- In principle there should be no boundaries to the growth of the network and must be a scale-free network
- Hence requires a dynamic structure to handle bandwidth + load balancing issues (maybe not just traffic but also local conditions)

Limitations
- Infrastructure must not restrict business solutions

Technical Requirements
A fully distributed infrastructure is required.
- Not centralised (including clusters with strong dependencies)
- Not limited decentralisation

Bootstrapping
- Accessibility (ability for new members to register with and join into the network)
- Performance
- Security

Growth
- Algorithm for growth must ensure critical properties are preserved as the network grows

Maximum Distance
- Specifying the diameter according to the lookup algorithm.

Durability
- High reliability of the whole network
- Needs a reliability model for the network to understand what guarantees can be made

De-fragmentation algorithm

One of the serious dangers for a distributed network is that of fragmentation of the network into separate sub-networks (islands), (see Figure 5-11, in Section 5.4). In this situation, peer-to-peer connections between different nodes, running different transactions and queries, may fail. This could lead to critical failures in long-term transactions, failure to provide optimal responses to queries, and ultimately a collapse of the community that the network aims to support. We discuss the problem of fragmentation and possible ways round it in the sequel (Section 5.4).

Recovery model
- Self-healing and ability for recovering after failure happened (chapter 3, we have discussed this)
- Needs a good theoretical understanding of the boundaries for recovery

Replication Model
- Replication should be consistent
- Which information is replicated?
- Supports Delegation – to enable roles or responsibilities to be rapidly delegated should a node be partially or wholly unable to fulfill them for some period of time.

Overall, we need to understand the importance of autopoietic P2P network as an environment that must be designed so that critical concepts stay alive. We can make a very useful analogy with the Gaia hypothesis,

which proposes that the Earth as a dynamic system which will react to perturbations in such a way that the system as a whole continues to survive.

5.2 'Small world' and 'scale-free' concepts in P2P networks

The purpose of the *autopoietic* P2P network is to enable connectedness of the community whether this is for conducting business interactions or sharing knowledge and research practices. Social scientists have long been concerned with the connections found between individuals in social networks; when and how they are formed and what the underlying structure of such connections is. Interestingly, computer science, and computer networks in particular, may have something to learn from social science in this respect.

Social Network Analysis (SNA) is, by large, the study of the structure of connections among individuals, organisations and nations. As discussed in [Hey06], social scientists working on SNA often directly observe or interview people ("nodes") as they go about their daily routines, graph their connections ("paths"), and then analyse the data to find unsuspected structure.

In a famous experiment, Professor Milgram of Harvard University asked Midwestern volunteers to send packages to strangers in Boston. The volunteers were not allowed to send the packages directly but rather had to use their own personal contacts to relay the package. Milgram found that the average number of personal contacts ("hops") used in each case was five which meant that a six-linked chain was formed – the bases for the popular phrase "*six degrees of separation*".

What this simple experiment shows is that networks cannot be completely clustered or totally random. On one hand, random sampling would not produce the tight clusters, e.g. of friends that most of us belong to. On the other hand, a strict division into clusters contradicts the six degrees of separation principle – if networks are completely clustered, how can any two nodes be "six degrees" apart?

Watts and Strogatz [WaS98] examined what mathematical conditions underlie the formation of a *small world* like Milgram's – that is, a web of interpersonal connections that link people into a community. Allegedly [Buc02], the two mathematicians started by drawing a series of dots on a piece of paper. They then connected the dots together with lines to produce a (simple) pattern that in mathematics is called a *graph*. If people are taken to be the dots and links of acquaintances to be the lines connecting them, then the social world becomes a graph.

In experimenting with different patterns, Watts and Strogatz identified a peculiar graph that seemed to capture the characteristics of real social networks. What they discovered was a subtle way of connecting the dots that was neither orderly nor random but somewhere in between. They concluded that networks must exist in a sliding scale between clustered and random, producing tightly knit group of friends but also 'short paths' that reach through-out the whole network.

In their famous 1998 paper [WaS98], Watts and Strogatz introduced a mathematical way of modelling the "six degrees of separation" principle. This is based on the *clustering coefficient* which precisely models the friend-of-a-friend effect. If George knows Sue and Sue knows Anne, the clustering coefficient weighs the likelihood of George also knowing Anne. In the same paper, the authors showed that this mathematical technique could model a variety of networks – from neural networks to a film actor's collaborators. The clustering coefficient has been widely used in biological network analysis from neuroscience to bioinformatics, for example, it is used as an objective measure of whether tow genes are coexpressed [Hey06].

What Watts and Strogatz failed to understand in their early version of the "*small world*" model [WaS98] was that searchability was important. This was pointed out by Kleinberg in [Klei00] who argued

that the small world concept could model how Milgram's volunteers got near their targets, but not how they eventually found them.

This lead Watts and his colleagues Newman and Dodds to refine the model accordingly: the key was to take 'multiple interests' into account. If each network cluster is formed based on some common interest, then searchability is facilitated by (relatively few) nodes with multiple interests that allow for cross-cutting links to other network clusters. The refinement of the model, subsequently published in [WDN02], advocates a multidimensional lattice rather than what was initially perceived as a one-dimensional ring. On a multidimensional lattice, cross-cutting links down through anther grid provide the 'short paths' necessary to get "six degrees of separation".

The study of network properties in systems as diverse as genetic networks or the World Wide Web revealed that their complex topologies have something in common. In particular, the investigation of the structure of the network of computers on the Internet [BAJ99] as well as the network of molecules in the cell, lead Barabasi and Albert in 1999 [BaA99] to conclude that a common property of many large networks is that the vertex connections follow a *scale-free* power-law distribution (see Figure 5-1). Unlike the more familiar Gaussian 'belle curve', a power-law distribution is far more skewed. In terms of networks, this means that there are a few highly connected nodes (often referred to as *hubs*) that get the majority of the connections. And this does not change while the network grows, so is irrespective of the scale of the network (hence, the term *scale-free*).

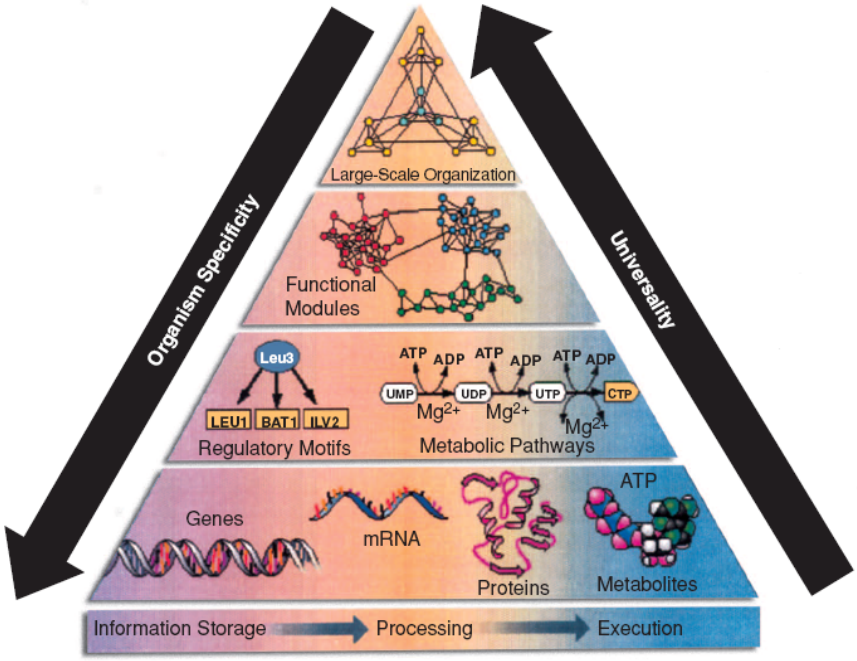


Figure 5-1 Complexity Pyramid [BaA99]

This feature, according to [BaA99], [Bar03], is attributed to two generic mechanisms. One is that networks expand continuously through the addition of new vertices. The second is that new vertices tend to attach to nodes that are already highly connected, i.e. that already have a large number of connections. Hence, in a dynamic, growing network governed by power-laws there is a notion of 'preferential attachment' – that is, one is more likely to attach where there already many links. A thorough account of the inception of *scale-free* networks and their characteristics can be found in [Bar03].

We have outlined the most basic features of networks, namely *small world* and *scale-free* and traced their inception back to social network analysis and bioinformatics. In the following, we revisit these two central network properties this time coming from a different angle, that of P2P networks and topologies.

Further computational analyses have reinforced that networks that are strongly adhere to the "six degrees of separation" theory are networks that have the 'small world' and 'scale-free' properties. In this part, we try to find a global categorisation for P2P networks, especially in terms of network theory. We use different views and models from the classical Erdös and Rényi model to Watts and Strogatz's 'small world' [WaS98], to Barabasi's scale-free network [BaA99] to evaluation frameworks and analyses from Adamic (HP Labs) [AdH00a], [AdH00b], [ALH02] and [ALH01]. The idea is to investigate the effect of these generic models with respect to primary network properties and functions such as resource discovery, bootstrapping, query response time, network growth limitations, fragmentation possibility and even probability of transaction abortion and ways to handle it.

For a long time, the modelling of physical as well as non-physical systems and processes has been performed under an implicit assumption that the interaction patterns among the constituents of the underlying system or process can be embedded into a regular, and perhaps universal, structure such as a Euclidean lattice (this is the case for a regular network). The side effect of this design became apparent in the early days of Napster where a centralised server was used to keep an index which was used for client registration, responding to queries and creating peer-to-peer connections between nodes (client and users in Napster's case). The system was admittedly efficient for search and look-up, but as the demand on the server was ever increasing the model became unsustainable (huge server costs). Note that the design of Napster was not entirely centralised, although it is categorised as such, since the server would eventually establish P2P connections between clients. Even so, the demand on the server would continuously cause traffic bottlenecks. However, this model is still popular in other areas of P2P networking, especially for aspects such as bootstrapping or keeping the dynamic structure of rings of servers.

Two mathematicians, Erdös and Rényi (ER), made a breakthrough in the classical mathematical graph theory. They described a network with complex topology by a random graph. Fundamentally, Gnutella uses this model. Many real-life complex networks are neither completely regular nor completely random, and in peer-to-peer networks even in just file sharing system, these two models had lots of problems which became evident in practical implementations (e.g Napster).

A key model that motivated the development of several P2P networks (especially discovery systems) is the 'small world' model. A conceptual definition of this model was introduced by Watts and Strogatz [WaS98], as discussed earlier. A prominent common feature of the ER random graph and the small world model is that the connectivity distribution of a network peaks at an average value and decays exponentially. Such networks are called "exponential networks" or "homogeneous networks," because each node has about the same number of link connections.

Scale-free networks ([BAJ99], [BaA99] and [Bar03]) or "power-law networks", differ fundamentally in their connectivity distribution from small worlds and random graphs. They are scale-free in the sense that their connectivity distributions are in a power-law form that is independent of the network scale [Bar03]. In contrast to an exponential network, a scale-free network is inhomogeneous in nature: most nodes have very few links and a few nodes have many links.

5.2.1 Key attributes for the assessment of alternative architectures

Given the significant volume of theoretical analyses of network structures, it is not surprising that a number of attributes have been identified that can be used to characterise certain aspects of network topologies. This section provides a glossary of the main ones.

Average path

The *average path length L* of the network is defined as the mean distance between two nodes, averaged over all pairs of nodes. We clearly need to minimise the average path length, in terms of number of hops needed to find the answer to a query. In addition, we must do this in a way so that the *optimal* solution to a query is found (for example, a search for the cheapest flight, should be guaranteed to find the cheapest flight available in the *whole* network, and not just a local minimum).

Clustering coefficient

One can define a *clustering coefficient C* as the average fraction of pairs of neighbours of a node that are also neighbours of each other. Suppose that a node *i* in the network has *ki* edges and they connect this node to *ki* other nodes. These nodes are all neighbours of node *i*. Clearly, at most $ki\ (ki - 1)/2$ edges can exist among them, and this occurs when every neighbour of node *i* is also connected to every other neighbour of node *i*. The clustering coefficient *Ci* of node *i* is then defined as the ratio between the number *Ei* of edges that actually exist among these *ki* nodes and the total possible number $ki(ki - 1)/2$, namely,

$$Ci = 2Ei/(ki(ki - 1)).$$

The clustering coefficient *C* of the whole network is the average of *Ci* over all *i*. Clearly, $C \leq 1$; and $C = 1$ if and only if the network is globally coupled; that is, every node in the network connects to every other node. In a completely random network consisting of *N* nodes, we have $C \sim 1/N$, which is very small as compared to most real networks. It has been found that most large-scale real networks have a tendency toward clustering, in the sense that their clustering coefficients are much greater than $O(1/N)$, although they are still significantly less than one (i.e., far away from being globally connected). This, in turn, means that most real complex networks are not completely random. Therefore they should not be treated as completely random and fully coupled lattices alike.

In many cases the nodes in a network are clustered according to some common domain. Indeed, we typically see this in the internet, where there is a high degree of connectivity amongst nodes concerning a common subject matter. This is already pointing us towards the general network topology we need for DBE. That is, it needs to be scale-free on a global basis, but with local small-worlds of highly connected clustered nodes.

Degree of a node

The degree k_i of a node *i* is usually defined to be the total number of its connections. The average of k_i over all *i* is called the *average degree* of the network, and is denoted by $< k >$.

Distribution degree (function) P(k)

The spread of node degrees over a network is characterized by a distribution function *P(k)*, which is the probability that a randomly selected node has exactly *k* edges. A regular lattice has a simple degree sequence because all the nodes have the same number of edges; and so a plot of the degree distribution contains a single sharp spike (delta distribution). Any randomness in the network will broaden the shape of this peak. In the limiting case of a completely random network, the degree sequence obeys the familiar Poisson distribution; and the shape of the Poisson distribution falls off exponentially, away from the peak value $< k >$. Because of this exponential decline, the probability of finding a node with *k* edges becomes negligibly small for $k >> < k >$.

Many empirical results showed that for most large-scale real networks the degree distribution deviates significantly from the Poisson distribution. In particular, for a number of networks, the degree distribution can be better described by a power law of the form

$$P(k) \sim k^{-\gamma}$$

The power-law distribution falls off more gradually than an exponential one, allowing for a few nodes of very large degree to exist. Because these power-laws are free of any characteristic scale, such a network with a power-law degree distribution is called a *scale-free network* (most real networks are not fully connected and their number of edges is generally of order *N* rather than N^2)

ER (Erdös and Rényi) model (random graph) versus 'small world'

Erdös and Rényi showed that, if the probability *p (probability for having an edge between any two nodes)* is greater than a certain threshold *pc ~ (*ln *N)/N*, then almost every random graph is connected, and ER random graph with *N* nodes has about *pN(N* − 1*)/*2 edges. The average degree of the random graph is $< k >= p(N-1) \cong pN$.

If L_{rand} be the average path length of a random network. Intuitively, about $< k >^{L_{rand}}$ nodes of the random network are at a distance L_{rand} or very close to it. Hence $N \sim < k >^{L_{rand}}$, which means $L_{rand} \sim \ln N / < k >$.

This logarithmic increase in average path length with the size of the network is a typical small-world effect. Because *LnN* increases slowly with *N*, it allows the average path length to be quite small even in a fairly large network.

The clustering coefficient of the ER model is *C = p =< k> /N << 1*. This means that a large-scale random network does not show clustering in general. In fact, for a large *N*, the ER algorithm generates a homogeneous network, where the connectivity approximately follows a Poisson distribution.

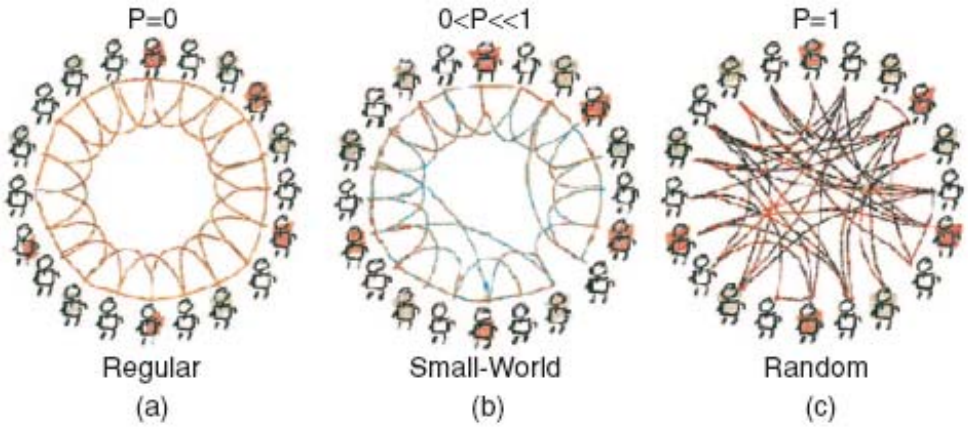


Figure 5-2 Classic comparison between regular, small-world and random network

Simple classic algorithm for WS 'small world' model

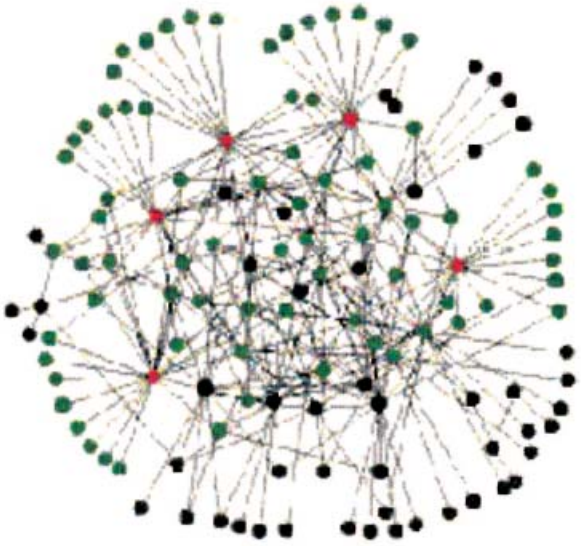
1) Start with order: Begin with a nearest-neighbour coupled network consisting of N nodes arranged in a ring, where each node i is adjacent to its neighbour nodes, i = 1, 2, … , K/2, with K being even.

2) Randomization: Randomly rewire each edge of the network with probability p; varying p in such a way that the transition between order (p = 0) and randomness (p = 1) can be closely monitored.

Simple classic algorithm for scale-free model

1) Growth: Start with a small number (m_0) of nodes; at every time step, a new node is introduced and is connected to $m \leq m_0$ already-existing nodes.

2) Preferential Attachment: The probability $\prod_i$ that a new node will be connected to node *i* (one of the *m* already-existing nodes) depends on the degree k_i of node *i*, in such a way that $\prod_i = k_i / \sum_j k_j$.



A scale-free network of 130 nodes, generated by the BA scale-free model. The five biggest nodes are shown in red, and they are in contact with 60% of other nodes (green).

Figure 5-3 Scale-free network

Using the attributes described so far in this section, together with the requirements for supporting long-life transactions, we can now provide an evaluation of various network structures. This highlights their respective strengths and weaknesses in the context of supporting a digital ecosystem for business. In what follows we give a brief account of the global information structure of current network structures.

a) Centralised design:

- Reasonable average path and diameter; in centralised average path is minimum (accessing to server and reaching the target) and even diameter is reasonably low. This can be considered as a reason for using a centralized system but other problems can affect the performance of such a system

dramatically; indeed manipulation of huge amounts of data can be impossible in a centralised system.

- Certainty in resource/service discovery; theoretically all answers to queries can be reached (if the network cost does not affect this performance).

- High cost growths after reaching the traffic limitation; if the system is highly service oriented or resource (especially file) sharing is part of system, the network traffic will increase quickly and the central point will have to be improved regularly. Otherwise the rate of failure will affect the system consistency.

- Long-term (long-running) transactions problem: the effect of long-term (long-running) transactions (as explained in Chapter 3), can be more than other designs (lack of replication causes/increases the problem and long waiting for other transactions, when a long-term (long-running) transaction work on a data item).

- High risk in critical situations (attacking/failure on server, exalted traffic period and etc); attack or any failure on server may destroy the whole network, server as a centre of network is critical point of network.

- High cost recovery system; recovery management will effect the performance of server and with this the whole network performance will decrease (because this network is highly dependent to its server).

- Single point of failure

b) Scale-free network with power-law structure:

- Reasonable search performance (average path and diameter) in power-law distribution degree; one of the proven attribute of scale-free network is low average path (actually it was the point of discovering this network in the nature). This useful attribute comes from power-law distribution degree, which can side effect in different situation.

- Reasonable clustering coefficient; another proven property of scale-free networks, as discussed earlier. This is one of the important useful attributes of this type of network.

- Flat growth; dead-lock and exponential traffic generation; as we know the growth in scale-free networks is two dimensional and because of power-law nature of distribution degree, it is focused on few nodes. Then based on different scenarios (different transactions) and dependency to those few nodes (with very high degree), the probability for dead-lock increases and network traffic can increase based on same pattern of distribution degree.

- High degrees nodes drop down with the growth of degree; this is one of the most important problems in this architecture, because first of all dependency on a few nodes cause a need for improvement of bandwidth for those specific nodes (improving the bandwidth of other nodes can not help), otherwise network performance can reduce. On the other hand, based on the dynamic nature of this network if for any reason the importance of any this nodes reduces (see Adamic's statistical study in HP labs [ALH02] and [ALH01]), not only this cost (for improving the bandwidth) is wasted, but also we have to improve the bandwidth of other high degree nodes for compensating the lack of cooperation of that specific node.

- Self-attack as an exponential growth based on age of platform ([BAJ99], [BaA99] and [Bar03]); based on Barabasi's research, the degree of highest nodes will increase based on their age by power-law.

- Performance reduction; Long term (long-running) transactions, uncertainty in resource/service discovery (result of high traffic on a high degree nodes and no warranty for reaching all nodes for reaching a certain answer in a query)

- High probability for Fragmentation

c) 'Small world' network (different versions of Chord)

- Reasonable average path and diameter (as discussed earlier in the current chapter).

- Reasonable clustering coefficient (as discussed earlier in the current chapter).

- Network growth problem; it is the most important problem, because the nature of WS small-world is static and if we allow a highly dynamic growth, the network will start to change to scale-free network (as a solution some P2P networks try to mix a hybrid solution with centralized system and used centralized node for some specific tasks, such as bootstrapping and system registry which again we have to deal with centralized systems problems).

- Very low performance on critical transactions (on popular services in the network); specifically in long-term transactions, this problem can reduce the performance of system and possibilities for deadlock will increase.

5.3 Birth and growth model

The primary requirement for an *autopoietic* P2P network is that there is no centralised directory as there is no precise control over the network topology or the placement of content/services on the network. It is formed by nodes joining the network following some loose rules. The resulting topology has certain properties but, in contrast to structured designs, the placement of contents is not based on any knowledge of this topology. To find contents/services, a node queries its neighbours. The most common query method is *flooding* – the query is propagated to all neighbouring nodes within a certain radius. Unstructured designs are extremely resilient to nodes entering and leaving the system. We refer to the policy (model) which is applied when a node enters and/or leaves the network as the *birth* and *growth model*.

5.3.1 Network reliability

The term 'reliability' is often used in different ways, with different connotations and as a result has different conceptual impacts. It is important to clarify the priorities and then discuss reliability according to these priorities. In this way, we might be able to reach some level of formalisation or clear measurement of

reliability within the network. In different scenarios/models, these priorities can be in conflict with each other and therefore, as a primary solution, we may need to engage in a trade-off (decrease the expectation in certain circumstances) between levels of reliability in view of different scenarios.

One of the major challenges in P2P network has to do with *fragmentation* – the situation where the network is divided to a number of smaller isolated networks ('islands'). Undoubtedly, the design of the *autopoietic* P2P network for OPAALS should take into account fragmentation as a top priority for reliability, but also for recoverability (most important in a business transactional environment; recall Chapter 3), and latency (especially with regard to contents sharing), which are also considered to be high priorities.

5.3.2 Network connectivity

Analysing the distribution degree function alone, can often mislead the reliability discussion. As a simple example, we may consider reliability in a typical scale-free network. In the first instance, the network failure tolerance and recoverability may indicate high resistance to fragmentation. However, a smart attack on hubs can have the directly opposite results, which may be missed in the first analysis of the network reliability. The situation is described in Figure 5-4.

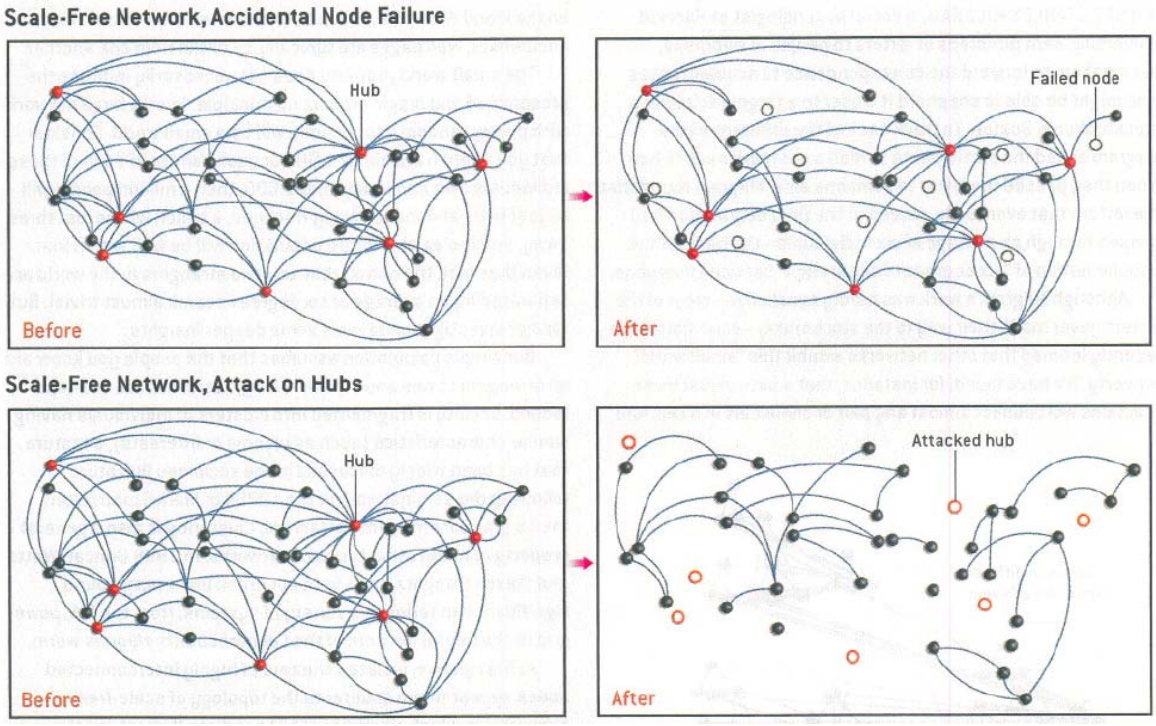


Figure 5-4 Scale-free network – accidental node failure and smart attack

In this case, parameters such as network connectivity, edge propagation and minimum number of edges per node, can play an important role for resistance against fragmentation. Additionally, replication at the level of network edges (that is, replication of edges instead of contents replication) can be considered as another important factor.

5.3.3 Latency, concurrency and recoverability

Another mystifying concept in the autopoietic P2P network is latency. Traditionally, there has been a dispute regarding the role of concurrency in relation to latency (see Figure 5-5). But this argument can be valid when we are dealing with any level of centralised control/processing in which any preceding transaction (request) has to be followed and checked by some centralised authority. This may be a simple identity check or the application of a complicated centralised concurrency control mechanism for ensuring consistency.

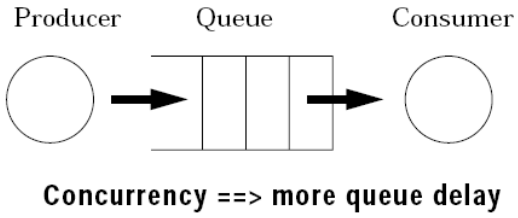


Figure 5-5 Latency and concurrency

In contrast with this view, the autopoietic P2P network neither relies on a centralised authority, nor suffers of global topological control. Furthermore, the autopoietic P2P network is supposed to be fully distributed and this implies that each node can act as a distributed unit for processing requests and improving latency. Consequently, allowing for more transactions to execute concurrently not only does not deteriorate latency but also seems necessary for recuperating the performance (latency) and exploiting the potential processing power of the system (recall Chapter 3, especially the discussion in Section 3.4.2).

If the *autopoietic* P2P network for OPAALS is to be considered for contents/knowledge sharing too, and not only as a P2P network which provides a high performance transactional environment for SMEs, then another factor may play a crucial role too: the handling of data packages. The loss or delay of data packages will have a direct impact on the response time, as shown in Figure 5-6, which means the middle-ware nodes for routing (or even data processing) have a significant role to play with regard to latency. Of course, any traffic bottlenecks and unreliable nodes at the heart of the network have the foremost impact on the latency.

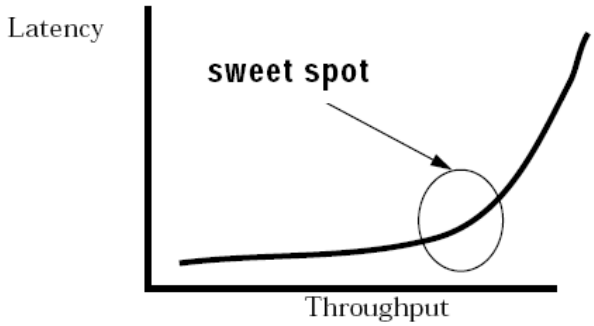


Figure 5-6 Latency and throughput of data packages

5.3.4 Evolutionary growth of metabolic networks: overview

Considering the above requirements, we propose to draw upon the evolutionary growth of metabolic (signal transduction) networks in designing the birth and growth model (Rzhetsky and Gomez 2001)[RzG01], [GLR01] for the auotpoietic P2P network in OPAALS. The analysis by Rzhetsky and Gomez shows that the evolutionary growth of metabolic networks has scale-free characteristics while it also has some interesting properties with respect to network connectivity.

The frequency of vertices connected to exactly k other vertices in metabolic (signal transduction) networks follows a power-law distribution. On the other hand, the distribution function degree is equivalent to

$$P(k) \sim k^{-\gamma}$$

or, precisely

$$P(k) = c.k^{-\gamma}$$

when c is a normalizing constant and γ diverges across networks (but usually has a value between 1 and 3) [BaA99].

The network follows a fractal model, as the shape of this distribution remains invariant to changes in network scale. This means that a small sub-graph has the same distribution as the complete graph from which it is derived. As the total number of different DNA and protein domains in a genome with both the total number of genes and the overall network topology, make an equation [KWR+02], it is statistically possible to analyse the relationship between the edge duplication/creation and the network topology.

Overview of the proposed model

Based on the Rzhetsky and Gomez hypothesis [RzG01], [KWR+02], [GNR03] in molecular networks each edge is implemented as a pair of mutually specific molecular structures. For example, in the case of an edge corresponding to phosphorylation of protein B by protein A, the beginning of an edge in protein A is encoded as a kinase domain, whereas the ending of the same edge in protein B is encoded as a specific protein domain recognized by the kinase domain of protein A. In another example, the beginning of an edge is encoded as a DNA-binding domain of a transcription factor, whereas the ending of the same edge at a gene regulated by the transcription factor is encoded by a DNA motif specifically recognized by the DNA-binding domain.

As a result, Rzhetsky and Gomez, have introduced a directed graph in which up the direction of the an edge (towards the node or opposite) is given in ‘upstream domain’ and ‘downstream domain’ terms. Each vertex of a molecular network is assumed to have either both upstream and downstream domains, or just one of the domains. Furthermore, the authors advocate that upstream and downstream domains experience independent duplication and that after duplication, each new domain copy randomly picks a domain from the available pool of domains of the opposite type (upstream or downstream) to form a two-domain protein. If there is no domain of the opposite type available, a one-domain protein is formed.

In the Rzhetsky and Gomez model, a domain[1] is defined as a functional unit that provides a specific interaction between two molecules. Note that, in the case of downstream domains, their definition lumps together protein and DNA domains so as to include network edges that correspond to transcription activation/inhibition events. In this analysis they use InterPro and Pfam databases to identify protein domains.

5.3.5 Domain duplication or link replication

In this model, the numbers of upstream and downstream domains in each class[2] has been monitored (without analysing how these domains are combined into genes or proteins). As a general approach for modelling stochastic processes, the primary assumption is that evolution in the network is governed by a homogeneous continuous-time Markov process, in which individual changes in the network arrive spontaneously but at constant rates. We note that this assumption may need to be changed/refined to model more realistic chains of events in the autopoietic P2P network, but nevertheless provides a valuable starting point.

One of the major evolutionary events that affect network growth is the duplication of genes and proteins (Figure 5-7). We can model domain duplication as some level of replication, the so-called *link replication*, which mostly inherits the nodes' links of the original copy. As a result of applying link replication, the connectivity of the network will be increased. The question remains as to how this is done and what formulae it follows, but this should be addressed in a subsequent report on the formal analysis of the network (D3.2).

[1] There are at least two frequently used definitions of a protein domain. Structural biologists define domains based on visual analysis of three-dimensional protein structures by human experts. Computational biologists define domains mainly through similarities in the primary sequences of several proteins. Despite considerable ideological differences between these definitions, both definitions are used in practice and the currently popular databases, such as InterPro (Apweiler et al., 2001) and Pfam (Bateman et al., 2000) incorporate domains defined in both ways.

[2] For simplification, classes of edges are defined in this model. As long as neither upstream nor downstream domains, across all vertices in the network, are required to be unique, a molecular network can use multiple copies of the same domain to encode multiple edges of the same kind. To distinguish between types of edges and different edges of the same type, they assume that all edges of a network, formed by the same set of two domains, belong to the same class.

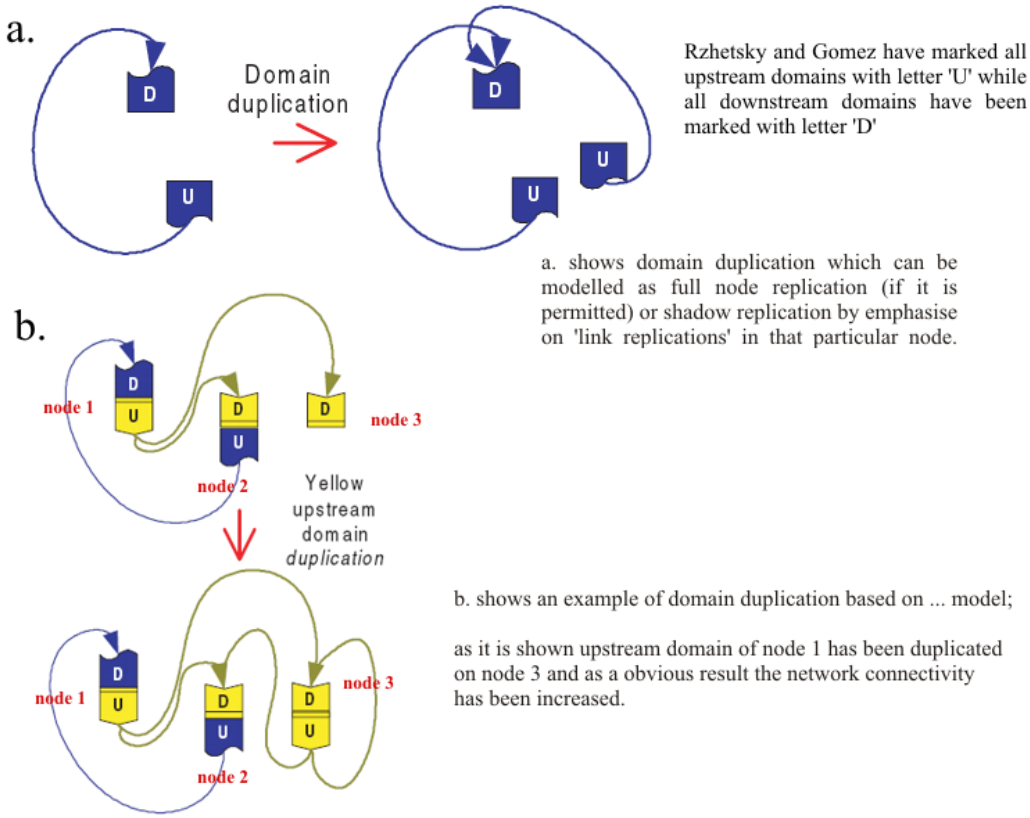


Figure 5-7 Network link replication as duplication of genes

5.3.6 Edge innovation: adding a node to the network

The second type of major event in evolution is the birth of new classes of edges (see Figure 5-8). This is often called *edge innovation*. Similarly to the evolutionary growth of metabolic networks [KWR+02], [RzG01], the birth of a new class of edges can happen during the addition of a new node to the network. Innovation can also be applied during the process of de-fragmentation – that is, when a fragmented network would try to connect to the other fragmented part of the network (cf Section 5.4.4).

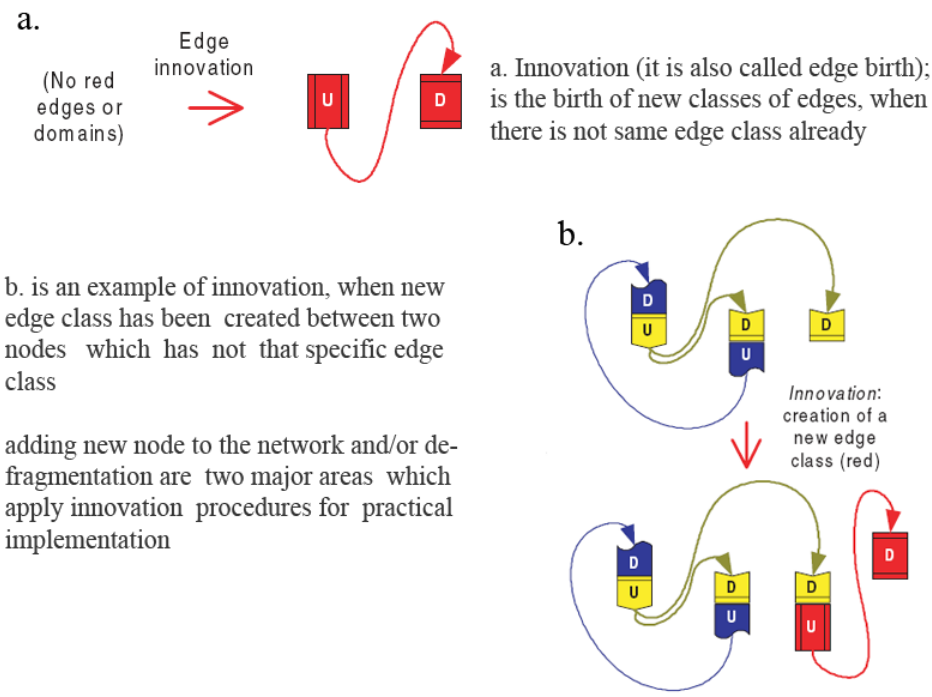


Figure 5-8 Edge innovation for adding a node to the network

The adaptation of basic concepts from the model for the evolutionary growth of metabolic networks, together with the two basic events in evolution – namely, domain duplication and edge innovation – form the foundation for a birth and growth model for the *autopoietic* P2P network in OPAALS. Following this biologically-inspired approach (apart from being within the cross-disciplinary spirit of the OPAALS community), allows for interesting characteristics of this model, such as good connectivity[3] and stability of the resulting scale-free network (with high resistance against hubs attack because of distribution degree and fractal nature of the network), can be inherited in the P2P network. Figure 5-9 provides a few snapshots of a simulated evolutionary growth of metabolic networks.

[3] The total number of network vertices is at least three times as large as the number of genes, based on Rzhetsky and Gomes analysis $D(t) \geq 3 \bullet G \frac{\sum_{i=1}^{V} i^{-\gamma_{in}}}{\sum_{j=1}^{V} j^{-\gamma_{in}+1}}$ where the coefficient γ may vary approximately from 2 to 3 for most metabolic networks, G is number of genes and D is the number of pairs of distinct domains.

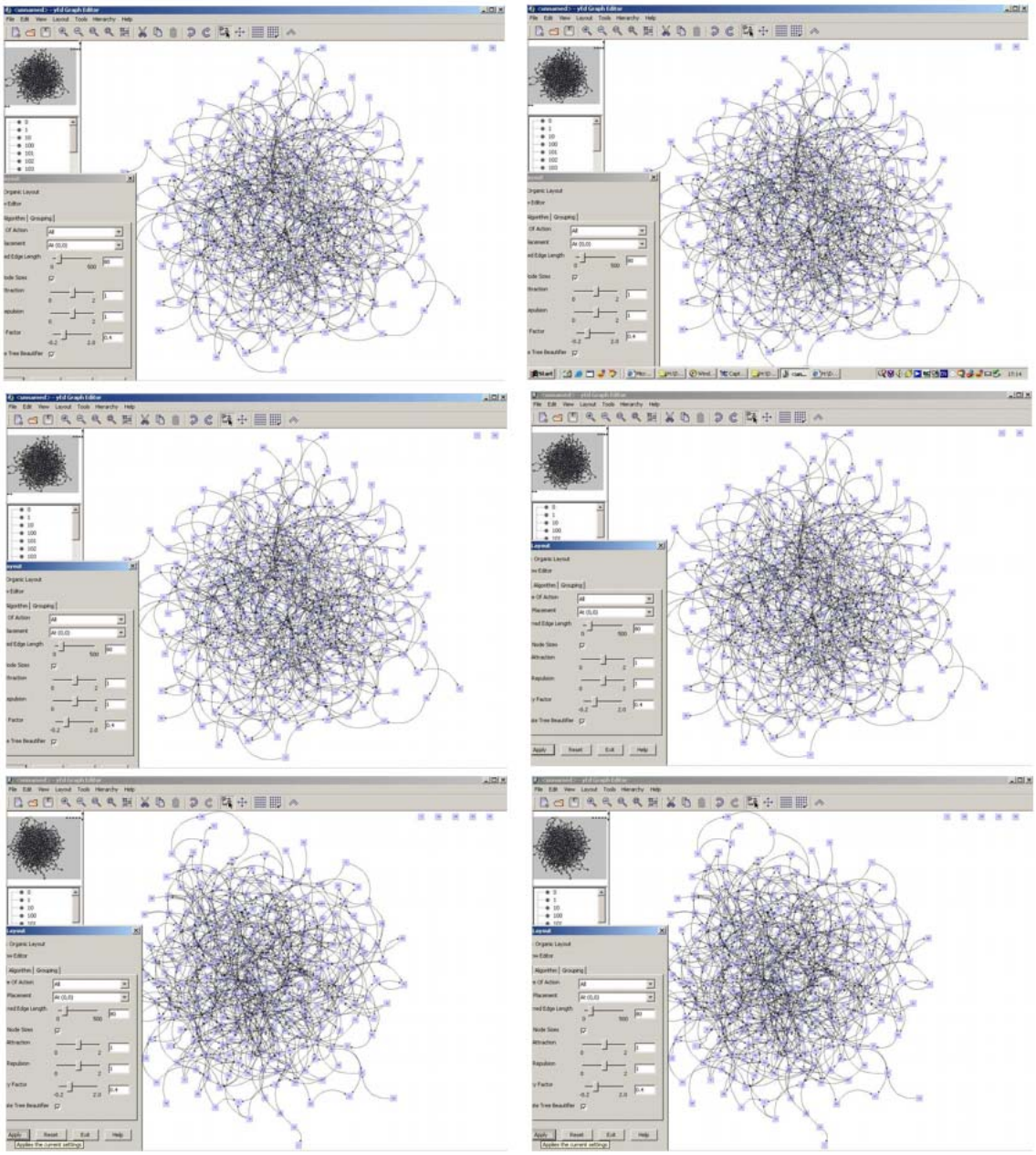
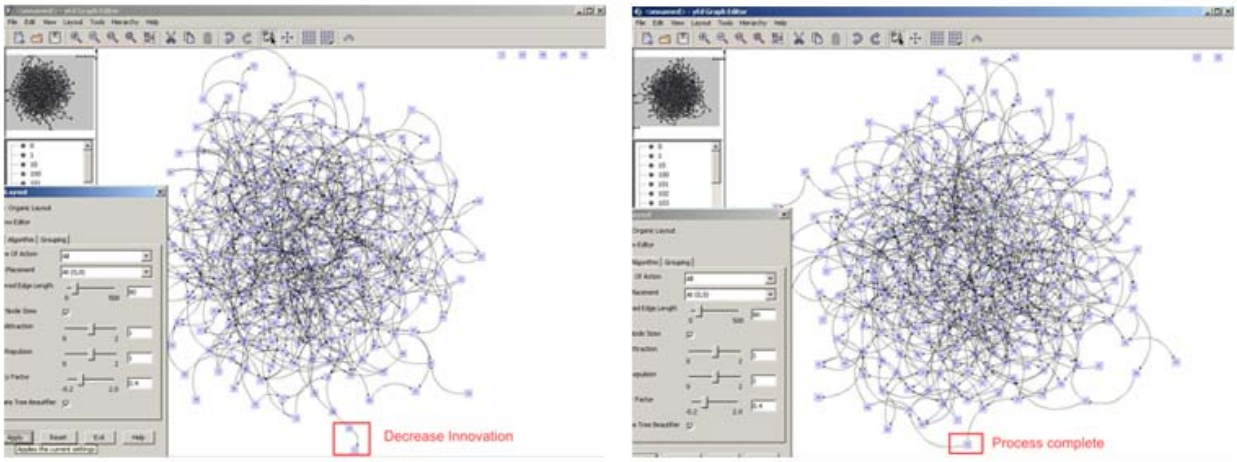


Figure 5-9 Early simulation of evolutionary growth model

5.3.7 Negative innovation: removing a node from the network

In the original evolutionary growth of metabolic networks, domain duplications and edge innovations are applied based on a probability function (which can be calculated statistically), and there is no proper model for loosing class edges. In contrast with that, dynamicity of SMEs' behaviours shows they may disconnect from and join the network regularly – this may even happen as a pre-defined behaviour, such as opening business hours. This is one reason we have started to make some extension to the model governing the evolutionary growth of metabolic networks.



a. loosing a node (decrease innovation) b. process completion

Figure 5-10 Removing a network node based on negative innovation

In the first instance we may reimburse negative innovation by applying domain duplication. In this way, not only the connectivity function does not act in an adverse way but also the scale-free properties may be enhanced (higher distribution degree on the stable nodes which do not disconnect regularly). Figure 5-9 shows an example of negative innovation in which a node after losing an edge will be supported by its neighbours (gets additional edge from them). As a result, the probability function used by the original evolutionary growth of metabolic networks for domain duplications and edge innovations cannot be applied directly. In any case, we are looking forward to seeing the statistical study of other partners to have better estimation of SMEs pattern behaviour in the real-world (specifically TI, SUAS and CN) and as a result re-calculate the probability function for all three major events in the network (domain duplication, edge innovation and negative innovation).

5.3.8 Factors under calculation

Other factors that need to be taken into account in the formal analysis for the P2P network parameters include:

- Latency – distribution degree and high connectivity
- (Providing ability for applying) Search algorithm
- Maximum time-period for reaching stability (after a node enters/leaves the network)
- Cluster coefficient

These aspects are currently being considered in experimenting with various simulations of the network. These should provide useful insights as to setting these parameters and how their values affect the overall behaviour of the P2P network. The results of the formal analysis together with the experiments with the simulations shall be reported in a follow-up report (D3.2).

In what follows we briefly discuss some aspects of each factor.

Latency – distribution degree and high connectivity

The main concern on WP3 is focused on transaction model (consistency, recoverability and then response time). Therefore we will not directly address latency in terms of a knowledge sharing environment but during the analysis of the network characteristics. Feasibility study of potential algorithm (as assisting other partners' tasks) can be possible, if proper output is prepared by other partners.

Providing ability for applying effective search algorithm

Designing search algorithm is not part of WP3 but we study feasibility and potential algorithm for making query and searching in the autopoietic P2P network based on the current structure. Probably the simplest algorithm to apply is the classical query method: flooding. A flooding-based query algorithm normally does not scale as each query generates a large amount of traffic and as a result large systems quickly become overwhelmed by the query-induced load.

In comparison with other distributed and unstructured P2P networks, the autopoietic P2P network provides better characteristics (such as reasonable diameter) which even flooding algorithm with proper TTLs (in comparison with the network diameter) can have reasonably good performance. Various alternatives to the classic query algorithm are working based on data replication strategy, and network topology and etc.

First conservatory which can improve flooding algorithm can be applicable by a little bit modification of the classic algorithm. It can be suggested asking nodes to check with the original requester before forwarding the query to neighbours. However, this approach can lead to message implosion at the requester node. Instead, we can use successive floods with increasing TTLs. A node starts a flood with small TTL, if the search is not successful; the node increases the TTL and starts another flood. The process repeats until the object is found. You expect this method to perform particularly well when hot objects are replicated more widely than cold objects, which is likely the case in Autopoietic P2P network when the network start to be stable (as the effect of negative innovation and domain duplication). This method is called "expanding ring." (We have to mention though expanding ring solves the TTL selection problem, it does not address the message duplication issue inherent in flooding.).

As the second classic solution 'Random walk' is significance to examine too. Random walk is a well-known technique, which forwards a query message to a randomly chosen neighbour at each step until the object is found. This message is called "walker". The standard random walk (which uses only one walker) can cut down the message overhead by an order of magnitude compared to expanding ring across the network topologies. However, there is also an order of magnitude increase in user-perceived delay. To reduce the delay we can increase the number of "walkers". That is, instead of just sending out one query message, a requesting node sends k query messages, and each query message takes its own random walk. The expectation is that k walkers after T steps should reach roughly the same number of nodes as 1 walker after kT steps, actually simulations confirm that. Therefore, by using k walkers, we can expect to cut the delay down by a factor of k.

There are several experimentations with different number of walkers. With more walkers, we can find objects faster, but also generate more loads. And when the number of walkers is big enough, increasing it further yield little reduction in the number of hops, but significantly increases the message traffic. Usually, 16 to 64 walkers give good results.

By using an autopoietic P2P network; it is possible to propose a proper solution, which provides Adaptive termination, minimized Message duplication and small 'Granularity of the coverage' (three important qualities for a query/search algorithm). The distribution degree function of Autopoietic P2P network and at the same time good connectivity of the network plus a short average path (and reasonable

diameter/MAXPATHLEN) are the most important characteristics which shows feasibility for having an ideal search algorithm.

Maximum time-period for reaching stability (after a node enters/leaves the network)

As this mathematical model may need to consider several other important parameters such as minimum number of nodes for creating a stable network, minimum number of hubs in relation to mobile nodes and so on, we consider this parameter as one of the highest priorities but it requires a finalized P2P network and then it can be practically estimated (based on simulations and/or a prototype test).

Cluster coefficient

We are interested in analysing the 'cluster coefficient' according to the neighbourhood of different business domains (which actually comes back to their business model on one hand, and their business processes on the other). Then, it should be possible to optimise a semantic search algorithm on the network, which can be considered as a fundamental step towards explorative service composition (which note is considered as a revolutionary idea in business to consumer interaction). But this task is not part of our workpackage and it needs a final result from few other workpackages.

5.4 Fragmentation

In free-scale networks which are highly dynamic and nodes enter and leave the network continuously, forming and removing links, there is always a possibility that tight clusters or small sub-networks are formed which are isolated. This leads to a situation where we have a number of "islands" rather than a large, free-scale network and is often captured by the term *fragmentation*. Fragmentation is one of the dangerous events which can happen for our network. The situation is described in Figure 5-11. There are several algorithms for de-fragmentation of a distributed network.

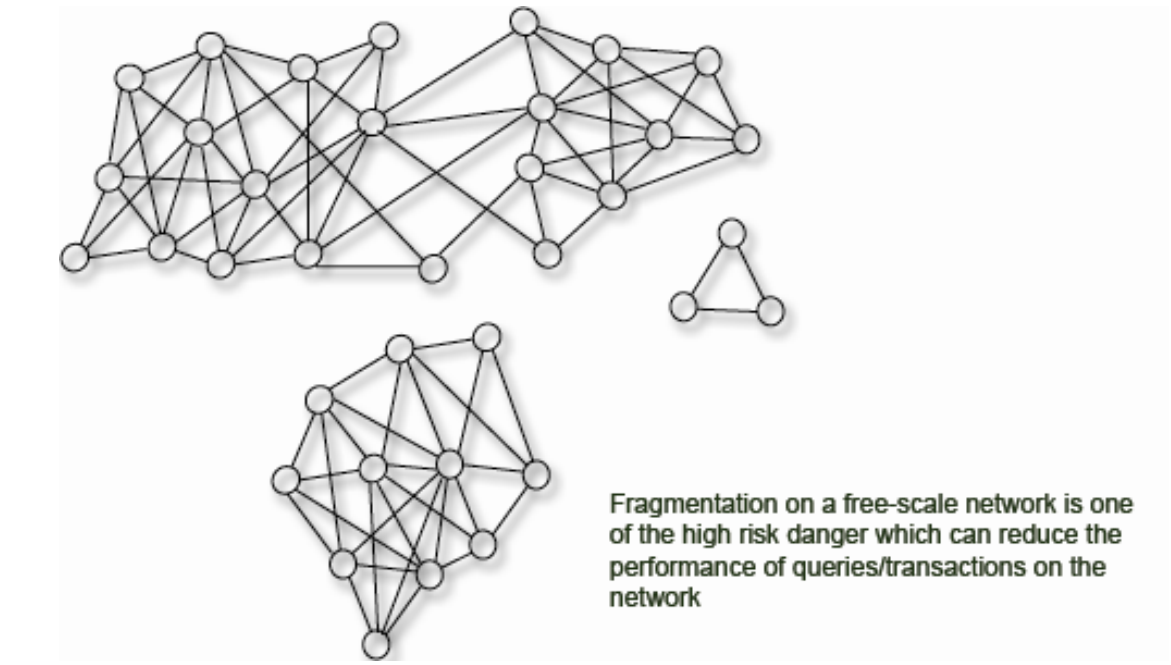


Figure 5-11 Fragmentation

However, as with all things, prevention is better than a cure. If the network can be designed to be stable against possible fragmentation, then significant network traffic to monitor and repair fragmentation can be avoided. Theoretically, the potential for fragmentation can be established during the creation of the network and the main reason for fragmentation is a missing routine in the birth and growth protocols of the network.

Therefore, as a first step, a digital ecosystem as a distributed network needs a birth model that avoids fragmentation. Subsequently a growth algorithm is needed that guarantees connectivity of the distributed network (described in Section 5.3). To achieve some guarantee of connectivity between different nodes, we have to have strong connectivity between nodes. At the same time, for efficiency of transactions and queries we need a short mean path between different nodes. One of the best known mechanisms to achieve these desirable features is replication, described in Figure 5-12.

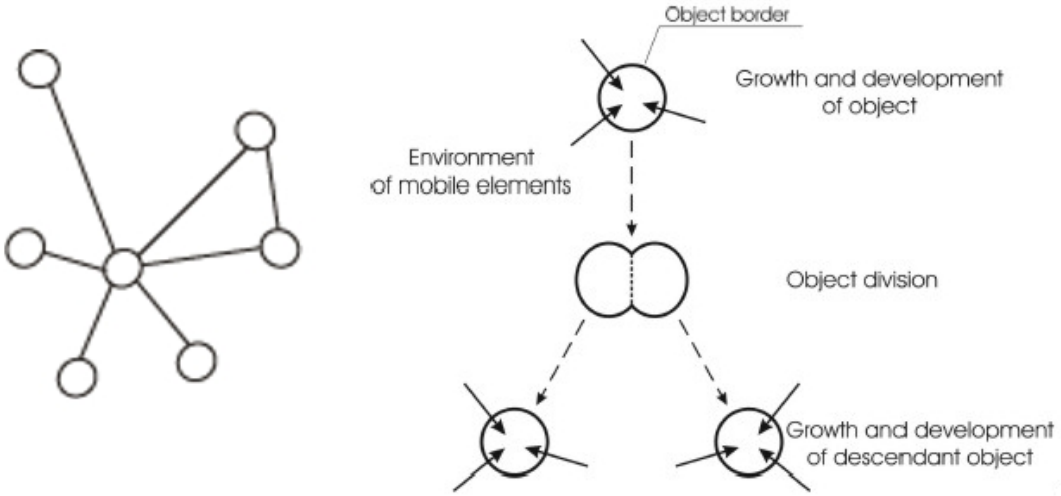


Figure 5-12 Growth through replication provides strong connectivity between nodes

Different levels of replication not only provide strong connectivity and shorter path lengths between nodes but can also be used in the de-fragmentation algorithm (for safe and secure de-fragmentation). We have tried to introduce the first one by using domain duplications (Section 5.3.5) and for de-fragmentation we use edge innovation as a primary solution (Section 5.3.6).

As a solution we can propose three different mechanisms for reducing the probability of fragmentation:

1. Applying effective Distribution degree function
2. Increasing degree of each individual node
3. Network snapshot and link replication

In the following sections, we will review these techniques with particular focus on the main algorithms for replication models. In addition, the relationship between a replication model and a birth-model (de-fragmentation) will be reviewed and analysed.

5.4.1 Applying effective distribution degree function

One of the parameter for increasing the probability for fragmentation is number of nodes with large number of links; on the other hand if the network relies on few hubs (with large number of links), any accident or attack on those hubs can cause fragmentation. Increasing number of these critical hubs will practically change the distribution degree function but as autopoietic P2P network is a self organised network, these can be done just during the birth and growth model of the network (4.3).

5.4.2 Increasing degree of each individual node

Increasing the degree of each individual node (the number of links to the node) can be a positive parameter for decreasing the probability of fragmentation. It means the stable nodes during the time will have more links and help to increase the connectivity of the network. In our first birth and growth model, we have tried to apply this by using the negative innovation mechanism (4.3.7) but later on by analysing this parameter we will try to reach a certain stability function during a time period.

5.4.3 Replication models and the OPAALS target

One of the best known mechanisms to achieve good resistance against fragmentation is replication. Different levels of replication not only can provide strong connectivity and shorter path lengths between nodes but also can be used in the de-fragmentation algorithm (for safe and secure de-fragmentation). Apart of our primary model for that (domain duplication; Section 5.3.5), in this section we try to analyse other prospectus and mechanism in this criteria.

In general, Replication is an important technique in peer-to-peer environments, where it increases data availability and accessibility to users despite site or communication failure. However, determining the extent of replication, the location of the replication manager (who is responsible for the replication of data) and where to replicate the data are the major issues which can affect the whole design of a peer-to-peer network. We reviewed several replication techniques which are applied in distributed or decentralised networks. Evaluation and feasibility analyses of these models show not only the best policies for replication but also can be used to inform choices of the structure of peer-to-peer networks. As we mentioned in previous sections, a decentralised structure is not suitable for OPAALS and these evolutions show even in the replication view that the most feasible model for a scale-free peer-to-peer network is the distributed model. After that we try to introduce the best candidate and brief vision for OPAALS' replication model.

There are a number of replication models for distributed/decentralised networks. HIDRA (as an object invocation mechanism) for the coordinator-cohort and the passive replication model offers support to ensure that all the replicas of the object being invoked are correctly updated before such an invocation is terminated [M-EGB98]. This mechanism also tries to ensure that if a primary or a coordinator replica crashes, the client is able to reconnect to the previously initiated invocations, collecting their results without requiring their re-execution. There are several interesting ideas in this model but the centralised invocations cause a high cost in control messaging, which leads to significant limitations for network growth. That is probably the main reason why there have been no significant implementations of this model since 1998.

The primary-backup replication model is a replication model which is commonly used in approaches to providing fault tolerant data services. Its extension to the real time environment, however, imposes the additional constraint of timing predictability, which requires a bounded overhead for managing redundancy. Zou and Jahanian discuss the trade-off between reducing system overhead and increasing (temporal) consistency between the primary and backup, and explore ways to optimize such a system to minimize either the inconsistency or the system overhead while maintaining the temporal consistency guarantees of the system [ZoJ98]. This motivation has been used as an oracle in several other models which avoid the limitations of the primary-backup replication model. We applied some ideas of Zou and Jahanian in our transaction model for providing a solid and low cost replication mechanism.

Mustafa, Nathrah, Suzuri and Osman in 2004, propose a hybrid replication model for fixed and ad hoc networks in order to achieve high data availability. For the fixed network, data is replicated synchronously in a diagonal manner of logical grid structure, while for the ad hoc network, data will be replicated asynchronously based on commonly visited sites for each user. In comparison to the grid structure technique, the diagonal replication technique (DRG) on a fixed network requires lower communication costs for an operation, while providing higher data availability. This is to be preferred for large systems [MNS+04]. The main problems with these approaches are not only the high cost mechanism for a large system but also particularly they considered indirect limitation for large systems which it can not be defined as scale-free network. Nevertheless, the idea of asynchronous replication based on commonly visited sites for each user is an important idea for OPAALS.

Fraga, Maziero, Lung and Loques Filho evaluate the use of an object-oriented open platform based on the CORBA standard for the implementation of replicated services. To improve the flexibility of the implementation, they use a reflective approach, which allows for separation of aspects related to the replication model from those related exclusively to the service being replicated. This separation makes it possible to modify the replication protocol according to the fault tolerance level desired, without any implications for the application code [FML+97]. Despite the low performance of their model in a scale-free network, their results can be used as valuable input for the implementation and evaluation of practical replication.

Another interesting implementation of a replication model is JReplica which is based on Aspect Oriented Programming (AOP). JReplica allows the separate specification of the replication code from the functional behaviour of objects, providing not only a high degree of transparency, as with previous models, but also the possibility for programmers to introduce new behaviour to specify different fault tolerance requirements [HST01] (we might mention in our later reports when Sun make an implementation plan based on Java language).

One the most important issues that we try to solve/review is the performance of the network in a heterogeneous scale-free network. High performance computing on a large-scale computational grid is complicated by the heterogeneous computational capabilities of each node, node unavailability, and unreliable network connectivity. Replicating computation on multiple nodes can significantly improve performance by reducing task completion time on a grid's dynamic environment. Yaohang and Mascagni in 2003 developed an analytical model to determine the number of task replicas to meet the performance goals in different computational grid configurations. Furthermore, by taking advantage of the statistical nature of grid-based Monte Carlo applications, they extended the computational replication technique to an N-out-of-M scheduling strategy for grid-based Monte Carlo applications, which could potentially form a large category of grid-computing applications. In addition, they established a corresponding model for the N-out-of-M scheduling mechanism. Simulations are used to validate the computational replication models. Their preliminary results showed that the models they used were effective in predicting the required number of replicas to achieve short task completion time with a given high probability [YaM03]. Indeed, we will be using this technique in the design of the OPAALS VPTN.

As can be seen, there are a number of problems with the distributed or replicated model, such as data synchronization, load balancing, and so on. Yu, Zhou, Lan and Yue Wu, in 2003 proposed another novel architecture for an Internet based data processing system based on multicast and anycast protocols. The

proposed architecture broke the functionalities of existing data processing systems, in particular the database functionality, into several agents. These agents communicated with each other using multicast and anycast mechanisms. They showed that the proposed architecture provides better scalability, robustness, automatic load balancing, and performance than the current distributed architecture of Internet-based data processing [YLY03]. In the customised areas, such as OPAALS, this architecture can be used as a part of the transaction model for improving our network performance, which we mentioned in our first draft model.

Fritzke and Ingels in 2001, proposed a replication control protocol that provides transaction properties based both on reliable and atomic multicast primitives and a two-phase locking protocol. Furthermore, they proposed two deadlock prevention rules to obtain two variants of the base protocol [FrI01]. The main problem that can arise from this protocol is the 'long-life transaction' problem, as expanded on in our report on the Distributed Transaction Model. (In our first draft model, with similar reliability rules we used the mixture of two-phase and three-phase locking systems for not only providing reliability and consistency, but we also try to overcome the long-life transaction problem in a way that gives us reasonable performance at the same time).

As a look ahead to the longer term, some of the most powerful ideas in large networks, which give unlimited growth abilities to the network and provide full flexibility for applying any transaction model, come from nanotechnical and nanosystems. By using this technology, not only could we have a fully distributed network, but also the fragmentation problem is solved in the definition of the network. In addition, we can provide an optimised distributed transaction model. The only important condition in nanotechnical and nanosystems is that 'the network has to be created under this model', because the foundation of the model is the network birth (which can be compatible with the OPAALS objectives).

5.4.4 De-fragmentation

In the last section, we tried to reduce the probability for fragmentations but even in the best circumstances, there is some probability of that. In this part, we want to see, if fragmentation is happened, what can be considered as solution. The edge innovation (recall Section 5.3.6) is our potential mechanism for de-fragmentation (as well as joining new node to the network). We propose two more solutions which try to use unique characteristic of the *autopoietic* P2P network by using edge innovation at the same time.

Virtual hubs

As autopoietic P2P network is a business network, each node during business transaction gather knowledge about other platforms who are involved in a transaction. This knowledge can play important role during de-fragmentation process, especially when a node is involved in a large number of transactions. We call such node (which is involved in huge number of transactions), virtual node. A virtual node physically may not be a hub (or have capacity of a hub), but because of collaborations on running several transactions, it has to know the physical addresses of these service providers and therefore it is possible to discover the address of these nodes inside of its transaction logs. By fetching these addresses we are able to do edge innovation (described in Section 5.3.6) and de-fragment the network, as shown in Figure 5-13. Practical design of this mechanism is feasible after finalising the transactional model and birth and growth model.

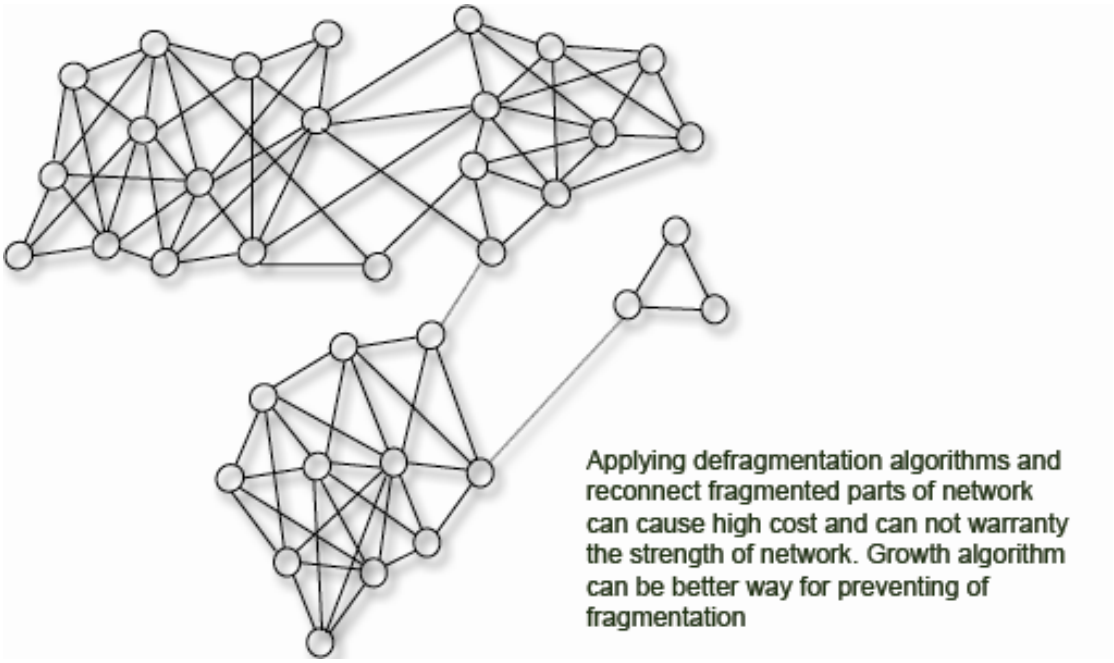


Figure 5-13: De-fragmentation algorithms typically only recover a weak form of connectivity that is not robust against subsequent fragmentation

Coordinator's knowledge through delegation knowledge

Another potential solution for de-fragmentation can be found during the delegation mechanism during business transaction life time (Section 3.3). When we are dealing with a delegation coordinator, two specific addresses should be transparent for the coordinator; source and destination address of delegation. Therefore the address of these two platforms can be found when these two are in different sides of a fragmented network.

Specifically this mechanism is interesting because in delegation process, we are dealing with unstable nodes and by applying this mechanism we are able to use even the knowledge of unstable nodes (as well as virtual hubs). We hope after implementation of transaction model by TechIdeas, we use this mechanism in practise.

5.5 Other important issues

In this section, additional dimensions of the *autopoietic* P2P network for OPAALS are discussed. These are not strictly related to supporting distributed long-term transactions in open communities of SMEs, but are certainly important for enabling open and trusted collaborations within a digital ecosystem for business. More specifically, we touch upon the topics of *distributed* identity and trust since these aspects are necessary for fostering a collaborative environment that facilitates the sustainability of the digital ecosystem. These topics are investigated in other parts of the OPAALS Network of Excellence and will be addressed from a socio-economic as well as a computing point of view. Here, the discussion is motivated from a computing angle, and in particular from the relation to the P2P architecture.

5.5.1 Distributed trust

In P2P systems, peers often must interact with unknown or unfamiliar peers, who may even reside in different security domains, and a major challenge is how to establish *trust* between different peers. By and large, the purpose of trust in this context is to protect against those who offer services rather than from those who want to access them. (The latter can be understood as an issue mostly related to security.)

The major challenge in a P2P environment, and especially one that is fully distributed, as considered in OPAALS) is how to establish trust between different peers *without* the benefit of a trusted third-party or authority to mediate the interactions. In this section we outline some preliminary thoughts and surrounding issues with regard to the development of a distributed trust model in relation to the autopoietic P2P network in OPAALS. We note that the OPAALS Network of Excellence (particularly WPs 4 and 7) will be looking at the development of a distributed trust model that goes beyond traditional trust models in that it should incorporate concepts from ecology and economics.

We have seen that a prerequisite for the autopoietic P2P network in OPAALS is that it is fully distributed (no centralised control). This implies that the challenge of establishing trust between peers needs to be addressed in a purely distributed manner. For example, eBay and Amazon foster trust between strangers in e-commerce but this assumes the existence of a centralised third-party and thus is not compatible with the design philosophy and intent of the OPAALS P2P architecture. Sometimes even when some trusted authorities are available (e.g. an authentication server or a certification authority), it is not advisable to assume that such authorities can monitor transactions and then declare the level of trust of different peers. A conventional certificate chain for instance, even if perfect, would at best validate the identity of a given party, but would not be able to guarantee that the given party is trustworthy for a particular purpose, whether this is a transaction involving a small payment or a multi-million deal. Consequently, peers have to rely on the previous history of a given peer, including the experience other peers have in interacting with it.

Trust models (e.g. [WaS06, WaS07, YSS04]) for large-scale distributed systems such as P2P systems (that go beyond file sharing) take a view of trust as a reputation mechanism that incorporates the knowledge of others in the system in deciding whether to trust another party (also in the system). In [YCC04] the authors take a view of trust that incorporates "reliability" (ability to provide high-quality services) and "credibility" (ability to provide trustworthy ratings to other peers).

In [WaS07] the authors augment the notion of trust in P2P architectures with "belief" or expectation – that the other peer's behaviour will support the peer's plans. Here trust incorporates the notions of "belief" (expectation of a positive outcome, good behaviour of the peer in question), "disbelief" (expectation of bad outcome) and "uncertainty" (no expectation). The approach taken is in fact based on available *evidence* – number of positive and negative experiences in trust reports gathered from multiple sources. This allows to talk about the effect of evidence (the more the better, i.e. the less uncertainty) and the effect of conflict or conflicting evidence (the more, the less certain).

It transpires that key aspects in dealing with distributed trust in P2P networks/architectures have to do with the generation, discovery and aggregation of the required rating information. Such aspects are briefly outlined below.

Ratings generation

Existing approaches tend to consider binary ratings – in binary ratings a peer rates the services from another peer as one of two values, either 0 or 1, either negative/unsatisfactory or positive/satisfactory. In fact, the Gnutella protocol, discussed in [YCC04], is a reputation-based approach to establishing trust in

P2P systems which uses a distributed polling algorithm based on binary ratings (0 or 1). This may be good for file-sharing (the file is either the correct one or not) but is not suitable for a business environment involving SMEs - factors such as previous years of collaboration, delivery time, QoS, credit payment etc cannot be considered.

It can be seen that there is an issue of assigning a suitable measure of trust. One possible solution might be to consider probabilistic ratings in the interval [0, 1] instead.

Ratings discovery

This aspect deals with how a peer gathers the information it needs in order to decide whether another peer is trustworthy. The algorithms for discovering the ratings are usually based on Gnutella protocols in which the requesting peer broadcasts the message to all other peers within a given TTL (Time-To-Live). It is not hard to see that in such polling algorithms each peer essentially queries all of its neighbours and hence come at a cost of network usage (bandwidth) and processing power. (Note that the reply, if any, is also sent back following the reverse path.) This is of concern when a number of long-running transactions are expected to take place over the distributed P2P network.

To alleviate the problems incurred by polling algorithms [YSS04] propose the use of *referrals* through which peers help each other find witnesses [YVS03]. That is, neighbours who indeed have previous experience with peer in question.

Ratings aggregation

The rating scheme should take into account that a peer has its own perception of the 'trustworthiness' of another peer, but also requires the knowledge of other peers via their own experience of interactions with that peer. Therefore, there is a notion of a local rating of a peer which should be combined with the testimonies received by other peers to produce the *aggregate* rating for the peer in question. Further, there is an issue of whether this aggregate rating is should be passed on to other peers (when a testimony is requested) or the testimony of a peer about another peer should only include the local rating.

A related aspect has to do with identifying deceptive or unreliable peers who as witnesses do not provide valid ratings (of another peer). The returned testimonies may include exaggerated positive or exaggerated negative ratings or even the complement of the true rating. This may be due to limited knowledge of the peer in question or an attempt of a witness to bias the rating generation for its own benefit.

Security considerations

Security is understood here as the process by which the trust model defends against attempts to bias the rating generation scheme that defines the level of trust of the peers. It is sometimes the case that peers return ratings that are not based on their own interaction with the peer in question. This has the effect that the ratings are based on *rumours* rather than the actual experience of the peers involved. As ratings are circulated in the P2P system, malicious peers may take advantage and spread rumours to attack normal peers. Another threat is posed by *deception* where a peer returns multiple ratings, and as a result has more influence in the decision on the trustworthiness of the peer in question. This would be easy to detect when the multiple ratings are sent using the same key (as the requester can discard duplicate messages with the same key). However, malicious peers may have stolen keys of other peers and act as a requesting (trusted) peer to query others multiple times. They may then send multiple ratings encrypted by different keys. The *autopoietic* P2P network in OPAALS is highly dynamic and nodes can enter or leave the network at any time (in Section 5.4 we described a preliminary birth and growth model for this purpose). This raises the issue of *pseudonyms* – that is, peers leave and re-enter using a different identity.

Other dimensions of distributed trust

Existing approaches to distributed trust in large-scale P2P systems, including the ones mentioned above, consider a trust model for a P2P network in which peers may leave or join the network but have a fixed number of neighbours. The autopoietic P2P network in OPAALS is a scale-free network whose topology is expected to continuously change, even if the characteristics that contribute to the survival of the participating communities are preserved, and this assumption is no longer reasonable.

Further considerations have to do with how trust evolves over time, especially in the face of a dynamically changing topology where peers acquire new neighbours and may lose old ones. It is not uncommon that parties (peers) alter their behaviour, either intentionally or as a by-product of the additional information gathered over time. Additionally, there is the question of whether trust propagates and if not, how can we still use the information provided by other peers. Finally, there is whole set of issues regarding social and organisational aspects which touch upon community building and community currencies which are addressed in other parts of OPAALS but are interrelated.

5.5.2 Distributed identity

Distributed identity (as opposed to identity) as a means for avoiding the bottleneck of dependency to a central authority is one of the most important issues in today's large scale networks, especially when the uniqueness of the identity is considered as the key point of the network. Further more according to the nature of autopoietic P2P network, not only each node needs a unique address but also each transaction has to have a unique identity too. During the process of delegation these two unique identity play crucial role for consistency of the model.

Unique identifiers, or at least sources of symmetry breaking, are necessary for point-to-point communication over a shared channel. If two units have duplicate IDS, both will respond to the same set of messages. When both try to respond simultaneously, there is no way for them to break the deadlock (since the two units begin in the same state and proceed deterministically, in particular pausing and retransmitting at the same time). The 'random exponential backoff' used to recover from collisions in CSMA/CD networks (e.g., Ethernet) is impossible without either a source of thermodynamic randomness, or an identifier such as the so-called MAC address, a unique number associated with each Ethernet adapter, that can seed a pseudo-random number generator. But this can be considered as a contradiction with machine indecency of resources (services, knowledge and transactions) and further more high probability of collision during platform migrations. This may cause a critical problem in autopoietic P2P network as mentioned in WP4 (particularly Task 4.1 and Task 4.4 shall be concerned with addressing this problem).

Present communication protocols (such as Ethernet) specify that the manufacturers must coordinate with one another in order to avoid assigning the same ID twice [Tan89]. As you can see in the classic solution for the problem, breaking symmetry to each logical unit identity is involved to two distinct subparts: The first part is to make all the units different in some way (to break symmetry and increase in entropy) and the second part is to assign the unique IDs to the units, assuming that the symmetry had already broken a decrease in entropy and results in some coordinated behaviour.

Smith [Smi99] in an interesting article analyse the cost and the problematic procedures for doing that and explores methods by which the devices could coordinate with one another to manage ID) assignment dynamically and automatically.

Another interesting solution to the problem has been introduced at biological knowledgebase criteria [CML04], they mention lack of the inability to programmatically identify locally named objects that may be widely distributed over the network. They conclude this shortcoming limits the ability to integrate multiple knowledge-bases, each of which gives partial information of a shared domain, as is commonly seen in bioinformatics (similarity of the problem at shared resources and services in Distributed Collaborative knowledge sharing (Task 4.2, in OPAALS) and Distributed Identity (Task 4.4, in OPAALS) seems quite interesting) . They used Life Science Identifier (LSID) and LSID Resolution System (LSRS) to provide simple and well-designed solutions to this problem (which it is based on the extension of existing internet technologies). Probably this structure and/or its adaptability to autopoietic P2P can be discussed or considered as one of the primary solutions for further extension!

Other practical solutions come from Sensor Networks [EDB+05] and [HBT04]. At [EDB+05], the authors' proposed algorithm starts by assigning long unique IDs and organizing nodes in a tree structure. The addressing size of this tree is based on (related to) the size of the network; then, unique IDs are assigned using the minimum number of bytes. Globally unique IDs are useful in providing many network functions, e.g. configuration, monitoring of individual nodes, and various security mechanisms. But as you can see the scalability of the network can argued (even with considering the interesting justifications of the authors!).

Alternatively at [HBT04], the authors develop a distributed identity management algorithm for multiple targets in sensor networks. The proposed algorithm is designed based on neighbouring nodes' knowledge (as each sensor has only a knowledge about its neighbourhood, not the global picture of the whole system) which can be solved in polynomial time [BHT04]. Each sensor is assumed to have the capability of managing identities of multiple targets within its surveillance region and of communicating with its neighbouring sensors (They Use Bayesian analysis to derive a data fusion algorithm that needs a prior probability of the given data). Apart of some differences in applied environment, algorithm has quite interesting characteristic which can be considered in a large scale network (as the authors avoid any presumptions of the network size or any pre-knowledge of the scale of network by each node, this can be quite applicable in autopoietic P2P when the size and dynamic nature of network do not let nodes to have a global image of the whole system and they can use just their neighbours!).

6 Concluding Remarks and Future Directions

In this report we have described work related to the key target (objective 6) of the OPAALS project which is concerned with the development of a P2P architecture to support distributed long-running transactions. The context in which this objective is considered is given by open communities of SMEs and the need for enabling open and trusted collaborations between them.

The term "Digital Ecosystem" is used at a variety of levels. It can refer to the run-time environment that supports deployment and execution of e-services. It can include the "factory" for developing and evolving services. But most importantly it can be expanded to include the enterprises and community that uses the ecosystem for publishing and consuming services. It is this last that is the most important driver for the underlying technology, since it is the ability to support and healthy and diverse socio-economic ecosystem that is the primary "business goal" of the OPAALS project. From that comes a specific focus on supporting and enabling e-commerce with Small and Medium-sized Enterprises – contributors of over 50% of the EU GDP.

The development of the autopoietic P2P network in OPAALS aims to facilitate a service-oriented business environment that supports business transactions between SMEs in a loosely-coupled manner and without relying on a centralised provider. In this way, SMEs can provide services and initiate long-running business transactions directly with each other.

Current web services transaction models and workflow managers provide inadequate support for long-running distributed transactions. This was highlighted in Chapter 3 of this report – a thorough review of existing popular transaction models can be found in the DBE report [KMR06] while some of the concerns raised have been acknowledged elsewhere, e.g. see [VZG+05], [FuG05]. Current designs are geared towards a centralised, or at best limited decentralised, model and this does not allow to realise the full promise of Service-Oriented Computing [PTD+06]. This is against the fundamental principle of loosely-coupled services and does not allow for the full promise of Service-Oriented Computing [Pap03], [PTD+06] to be realised. This may not be of concern for a relatively small number of highly visible service providers but raises barriers to the adoption of SOC by SMEs.

We have described in this report a distributed model of multi-service long-running transactions which has been designed for open collaborations within a community of SMEs in digital ecosystem. The model considers various forms of service composition to cover a wide spectrum of business scenarios. It allows the sharing of uncommitted results within a transaction as well as the exchange of results across transactions before their final commitment (partial results). At the heart of the model are the distributed log structures, provided by the IDG and EDG graphs, which in combination with the fine-grained lock mechanism allow for maximal concurrency in the transactional environment of a digital ecosystem. In addition to providing consistency in the transaction model, the locks mechanism provides the capability for efficient recovery management, in terms of preserving as much progress-to-date as possible (omitted results), and provision for alternative scenarios or paths of execution (forward recovery).

The coverage of most B2B scenarios increases the expectation for adoption by SMEs. This is enhanced by the fact that there is no dependency on centralised authorities (and the underlying P2P network plays a critical role in this), which inadvertently are a few large organisations. Further, this view of long-running transactions comes with the potential of a *virtual organisation*, which can act as the service aggregator envisioned in service-oriented computing roadmap [PTD+06] in that it offers compositions (possibly through *explorative* composition) of composite services. This opens an array of opportunities for SMEs with regard to evolving their business and expanding the range of provided services.

The full potential of the distributed transaction model in OPAALS cannot be realised without the significant development of the underlying P2P architecture. In this report, we have identified a number of areas that need to be addressed in the design of a P2P network to support distributed long-running transactions. We started by identifying the attributes that are needed in digital ecosystems and then analysed where current models fail in meeting such requirements.

One of the pillars of the P2P design in OPAALS is that the network is *fully distributed*. This means there are no dependencies on single organisations, as there is no central authority, and no critical points of failure. In fact, the topology of the network draws upon concepts from social network analysis and has *small world* and *scale-free* characteristics. Considering that the main driver behind the *autopoietic* P2P network is that it supports business transactions between SMEs, a node may interchange between being 'live' and 'offline' continuously, which means we are working with a highly dynamic transactional environment. Since this behaviour is to be expected, the topology of the supporting P2P network is ever changing and this has been factored in its design so it can adopt accordingly. The challenge is how to ensure that the main characteristics of the network are preserved in the presence of an ever changing topology.

This underlines the need for evolution based on a birth and growth model that perpetuates network parameters such as connectivity, average path, distribution degree and copes with fragmentation. This is where we turn to biology and examine whether the design of the *autopoietic* P2P network can be informed by biochemical processes. The proposed model draws upon concepts appearing in the evolutionary growth of metabolic networks [RzG01], [GLR01]. This has been adopted to model the growth of the P2P network, including an extension with capabilities for modelling the removal of a node (e.g. consider an SME going offline, based on different time zones or regional policies).

In the first instance, we have compensated for negative innovation in metabolic networks with domain duplication. Early experiments on the simulated evolutionary growth of the autopoietic P2P network indicate that not only the connectivity function does not act in an adverse way but also the scale-free characteristics of the network structure are reinforced. Further experimentation with simulations is under way and in the next phase this should take into account the pattern of behaviour of SMEs in the real world, based on the statistical study of other partners within the OPAALS Network of Excellence. Early results have been encouraging and some have been included in this report (e.g. in Section 5.3). The results of the formal analysis, supported by a simulated evolutionary growth model, will be reported in the oncoming deliverable D3.2 of work package 3.

Other factors that need to be taken into account in the formal analysis of the P2P network parameters include latency (consistency, recoverability, and then response time), maximum period of time needed for reaching stability after a node enters/leaves the network, estimations for determining the most suitable cluster coefficient, and providing the ability for the application of an efficient search algorithm and lookup mechanism. Such aspects are currently under consideration and various simulations should provide useful insights as to setting these parameters and deriving a formal description of their effect on the overall behaviour of the P2P network. As mentioned above, the formal analysis together with results from the simulations and experiments will be reported in deliverable D3.2.

In addition, a further dimension of the formal analysis in work package 3 has to do with the formal description of the transaction model. The discussion on modelling multi-service long-running transactions in Chapter 3 of this report illuminated the importance of the dependencies that arise in a highly dynamic transactional environment both *within* and *between* transactions. Setting these out properly is a prerequisite for a successful outcome of the long-lived business activity supported by the corresponding transactions. In addition, such dependencies need to be captured appropriately if the transaction model is to provide recovery management that goes beyond simply aborting the whole transaction in the event of some failure.

Recently there has been some interest in the formal methods research community in work providing a foundation for a theory of *reversible* actions. Among them is the attempt to enhance the Communicating Sequential Processes (CSP) [Hoa85] algebra with *compensating* actions, found in [BHF05]. Another strand,

which is one of the primary interests of the *Sensoria* project [SEN06], is looking at the adoption of *Sagas* [BMM05] algebras to a nested transaction model.

However, and although this is only based on preliminary analysis at this stage, such approaches tend to overlook the intrinsic aspects of *distributed* long-running transactions in a digital ecosystem for business. Apart from the all important issues of full distribution and local autonomy, the focus seems to be on a sequence of actions and a principled way of undoing all actions (in the reverse order) from the point of failure onwards. Hence, true-concurrency is not addressed (with potentially significant effects on latency, for instance) and other issues such as partial results, forward recovery and omitted results are not taken into consideration.

Our starting point is the transactions themselves (and their characteristics in a European digital ecosystem) and we consider a true-concurrent framework for the formal description of a transaction, in terms of the order of execution of the underlying services, their dependencies and alternative scenarios (if any). This is based on Shields' more general theory of non-interleaving representation of communicating systems' behaviour [Shi97] and has been recently applied to modelling interactions [Mos05].

The framework is expressive enough to capture various modes of interaction and distinguish between concurrency (and this is on top of parallel composition) and nondeterminism. Recently, it has been applied to reasoning about scenario-based specifications [MSK07], in the presence of true-concurrency, as it has been shown to support the gradual elaboration of scenarios [Mos05]. The aim is to use the formalisation to reason about the set of behaviours that are possible within the design of a transaction and identify what behavioural scenarios are possible in recovery management. Preliminary ideas on this formal framework for the behaviour of transactions have appeared in [RMK07a]. A thorough understanding of the behaviour exhibited by a transaction can ease deployment, implementation and testing and increase the expectation of a successful outcome. A proper account of the formal analysis within the distributed transaction model will be given in deliverable D3.2.

This report has identified the key aspects and characteristics in the design of the autopoietic P2P architecture in OPAALS. The formal analysis of further experiments with the parameters of this design used to drive the prototype implementation, reported in D3.2, should lead to the refinement of the present design into the final architecture of the autopoietic P2P network in OPAALS (reported in D3.3, at the end of the first phase).

7 References

[ABC+01] Aitken, D., Bligh, J., Callanan, O., Corcoran, D., & Tobin, J., (2001), Peer-to-Peer Technologies and Protocols, Available at: http://ntrg.cs.tcd.ie/undergrad/4ba2.02/p2p/ (last accessed: 02/11/06).

[AbH02] Aberer, K. and Hauswirth, M. An Overview on Peer-to-Peer Information Systems, Workshop on *Distributed Data and Structures (WDAS-2002)*, (available at: minoas.di.uoa.gr), 2002.

[Ada99] Adamic, L. A. The Small World Web. Proceedings of *Third European Conference on Research and Advanced Technology for Digital Libraries (ECDL 99)*, Paris, France. Springer Verlag, Lecture Notes in Computer Science vol 1696, pp. 443-452, 1999.

[AdH00a] Adamic, L. A. and Huberman, B. A. Power-law Distribution of the World Wide Web. *Science*, 287 (5461): p. 2115, 2000.

[AdH00b] Adamic, L. A. and Huberman, B. A. The Nature of Markets in the World Wide Web. *Quarterly Journal of Electronic Commerce*, 1(1): 5-12, 2000.

[ALH02] Adamic, L. A., Lukose, R. M. and Huberman, B. A. Local Search in Unstructured Networks, arXiv:cond-mat/0204181 v2, 4 Jun 2002.

[ALH01] Adamic, L. A., Lukose, R. M., Puniyani, A. R. and Huberman, B. A. Search in Power-law Networks, Physical Review E, Vol 64, p. 046135 , 2001.

[AJB00] Albert, R., Jeong, H. and Barabási, A. Error and Attack Tolerance in Complex Networks, *Nature*, 406: 378-382, 2000.

[AzM03] Azzedin, F. and Maheswaran, M. Trust Modeling for Peer-to-Peer Based Computing Systems, Proceedings of 17th *International Parallel and Distributed Processing Symposium* (IPDPS'03), pp: 99.1, 2003.

[Bag05] Baghdadi, Y. A Web Services-based Business Interactions Manager to Support Electronic Commerce Applications", Proceedings of the *7th International Conference on Electronic Commerce*, Vol. 113, pp: 435-445 , 2005.

[BHT04] Balakrishnan, H. Hwang, I. and Tomlin. C. (2004), 'Polynomial approximation algorithms for belief matrix maintenance in identity management.' In Proceedings of the 43rd IEEE Conference on Decision and Control, Atlantis, Paradise Island, Bahamas, December 2004.

[BaA99] Barabasi A. L and Albert R. Emergence of Scaling in Random Networks. *Science* 286(5439): 509–512, 1999.

[BAJ99] Barabasi, A. L., Albert, R. and Jeong, H. The Diameter of the World Wide Web, arXiv:cond-mat/9907038 v1, 2 Jul 1999.

[Bar03] Barabasi L. A. *Linked: How Everything is Connected to Everything Else and What it Means for Business, Science and Everyday Life*. PLUME, USA, 2003.

[Bar01] Barkai, D. Peer-to-Peer Computing, Technologies for Sharing and Collaborating on the Net. Intel Corporation, 2001.

[BE07] Bininda-Edmonds, O.R.P. *et al*, *Nature* 446, 507-512, 2007.

[BeN97] Bernstein P. A. and Newcomer E. *Principles of Transaction Processing*. Morgan Kaufmann Publishers, 1997.

[Ber03] Bergner, M. Improving Performance of Modern Peer-to-Peer Services, , Department of Computer Science, UMEA University, 2003.

[BHMcC+03] Booth, D., Haas, H., McCabe, F., Newcomer, E., Champion, M., Ferris, C. and Orchard, D. Web Services Architecture, 8 August 2003. Available at: http://www.w3.org/TR/2003/WD-ws-arch-20030808/

[Bro02] Broberg, J. C. Glossary for the OASIS WebService Interactive Applications (WSIA/WSRP), OASIS Web Services Interactive Applications TC Committees, 2002. Available at: http://www.oasis-open.org/committees/wsia/glossary/wsia-draft-glossary-03.htm

[BMM05] Bruni R., Melgratti H. and Montanari U. Theoretical foundations for Compensations in Flow Composition Languages. Proceedings of *32nd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pp. 209-220, ACM Press, 2005.

[BPM07] Brown, J.W., Payne, R.B. and Mindell, D.P. Biol. Lett. doi:10.1098/rsbl.2006.0611, 2007.

[Buc03] Buchanan M. *Small World: uncovering nature's hidden networks*. Weidenfeld and Nicolson, London, 2002.

[BEK93] Bukhres, O. A., Elmagarmid, A. K. and Kühn, E. Implementation of the Flex Transaction Model. *IEEE Data Engineering Bulletin*, 16(2): 28-32, 1993.

[BHF05] Butler M. J, Hoare A. C. R. and Fereira C. A Trace Semantics for Long-Running Transactions. *Proceedings 25 Years of CSP*, Lecture Notes in Computer Science, Vol. 3525, pp. 133-150, Springer, 2005.

[CCJ+04] Cabrera, F. L., Copeland, G., Johnson, J. and Langworthy, D. Coordinating Web Services Activities with WS-Coordination, WS-AtomicTransaction, and WS-BusinessActivity. Available at: http://msdn.microsoft.com/webservices/default.aspx, January 2004.

[CCC+03] Cabrera, L. F., Copeland, G., Cox, W., Feingold, M., Freund, T., Johnson, J., Kaler, C., Klein, J., Langworthy, D., Nadalin, A., Orchard, D., Robinson, I., Shewchuk, J., Storey, T. and Satish Thatte. Web Services Coordination Framework (WS-Coordination), September 2003.

[CCC+04a] Cabrera L. F. Copeland G., Cox W., Feingold M., Freund T., Johnson J., Kaler C., Klein J. Langworthy D., Nadalin A., Orchard, D., Robinson I., Shewchuk J., Storey T. and S. Thatte. (2004-a) Web Services Atomic Transaction Framework(WSAtomicTransaction), January 2004.

[CCC+04b] Cabrera L. F., Copeland G., Cox W., Feingold M., Freund T., Johnson J., Kaler C., Klein J., Langworthy D., Nadalin A., Orchard D.; Robinson, I., Shewchuk J., Storey T. and S. Thatte. (2004-b) "Web Services Business Activity Framework (WSBusinessActivity)", January 2004

[CdeS95] Campos, R. V., de Souza e Silva, E. Availability and Performance Evaluation of Database Systems Under Periodic Checkpoints, Proceedings 25th *International Symposium on Fault-Tolerant Computing (IEEE CNF)*, 27-30, pp: 269 – 277, 1995.

[CDF+03] Ceponkus, A.; Dalal, S.; Fletcher, T.; Furniss, P.; Green, A. and Pope B. Business Transaction Protocol Version 1.0, An OASIS Committee Specification, 3 June 2002. Available at: http://www.oasis-open.org/committees/download.php/1184/2002-06-03.BTP_cttee_spec_1.0.pdf

[ChG94] Chen, Y.-W.; Gruenwald, L. Research Issues for a Real-time Nested Transaction Model. Proceedings of the *IEEE Workshop on Real-Time Applications*, pp:130 – 135, 21-22 July 1994.

[CTT97] Chin, C. Chung-Kie Tung and Shang-Rong Tsai. Phaeton: a log-based architecture for high performance file server design. Proceedings of *Pacific Rim International Symposium on Fault-Tolerant Systems*, pp. 28-33, 1997.

[CML04] Clark, T. Martin, S. and Liefeld, T. Globally distributed object identification for biological knowledgebases, *Briefings in Informatics*. 5(1):59–70. March 2004.

[CMH+02] Clarke, I., Miller, S.G., Hong, T.W., Sandberg, O. and Wiley, B. Protecting Free Expression Online with FreeNet", *IEEE Internet Computing*, 6(1) 40-49, Jan-Feb 2002.

[CSW+00] Clarke, I., Sandberg, O., Wiley B. and Hong, T. W. FreeNet: A Distributed Anonymous Information Storage and Retrieval System, In Proceedings of the *ICSI Workshop on Design Issues in Anonymity and Unobservability,* Berkeley, California, June 2000.

[CrP02] Crowcroft, J. and Pratt I. Peer-to-Peer: Peering into the Future. In Proceedings of the *IFIP-TC6 Networks 2002 Conference*, Pisa, Italy, May 2002.

[CKM+03] Curbera F.; Khalaf R.; Mukhi, N.; Tai, S. and Weerawarana, S. The Next Step in Web Services, *Communications of the ACM*, 46(10): 29-34, October 2003.

[DTL+03] Dalal, S.; Temel, S.; Little, M.; Potts, M. and Webber, J. Coordinating Business Transactions on the Web, *IEEE Internet Computing* 7(1): 30-39, January - February 2003.

[DaP90] Davey, B. A. and Priestley, H. A. *Introduction to Lattices and Order*, Cambridge University Press, 1990.

[Dat96] Date C. J. *An Introduction to Database Systems*. 5th Edition, Addison Wesley, USA, 1996.

[DSW94] Deacon, A.; Schek, H.-J.; Weikum, G. Semantics-based multilevel transaction management in federated systems. Proceedings of *10th International Conference on Data Engineering (IEEE CNF)*,, pp. 452 – 461, 1994.

[DKA+04] Derbas, G., Kayssi, A., Artail, H. and Chehab, A. TRUMMAR - a trust model for mobile agent systems based on reputation. Proceedings *IEEE/ACS International Conference on Pervasive Services* (ICPS 2004), pp. 113 – 120, 2004.

[DBE06] Digital Business Ecosystems (DBE) EU-FP6 IST Integrated Project No 507953. Available at http://www.digital-ecosystem.org [last accessed 19 March 2007]

[DNB03] Ding, C. H.; Nutanong, S.; Buyya, R. Peer-to-peer Networks for Content Sharing, Technical Report, GRIDSTR-2003-7, Grid Computing and Distributed Systems Laboratory, University of Melbourne, Australia, December 2003.

[DLD97] Domel, P.; Lingnau, A.; Drobnik O. Mobile Agents Interaction in Heterogeneous Environment, *First International Workshop on Mobile Agents*, April 1997, No 1219, pp: 136-148
Eder, J. and Liebhart, W. The Workflow Activity Model WAMO. Proceedings of *3rd Int. Conference on Cooperative Information Systems (CoopIS)*, 1995.

[DKR02] Druschel, P.; Kaashoek, M. F. and Rowstron, A. I. T. Peer-to-Peer Systems. Proceedings of *First International Workshop, IPTPS*, 2002.

[Edw02] Edwards, J. *Peer-to-Peer Programming on Groove*, Addison-Wesley, 2002.

[Elm94] Elmagarmid A. *Database Transaction Model for Advanced Applications*, Morgan – Kaufmann, 1994.

[EDB+05] ElMoustapha Ould-Ahmed-Vall, Douglas M. Blough, Bonnie S. Heck, George F. Riley (2005) 'Distributed Global Identification for Sensor Networks', School of Electrical and Computer Engineering, Georgia Institute of Technology Atlanta, GA 30332-0250.

[EmT00] Emmerich, W.; Tai, S. Advanced Transactions in Enterprise Java Beans, Proceedings of *Engineering Distributed Objects: Second International Workshop*, pp: 215, Springer Berlin / Heidelberg, 2000.

[FML+97] Fraga, J. Maziero, C. Lung, L.C. Loques Filho, O.G. Implementing Replicated Services in Open Systems Using a Reflective Approach, Proceedings of *Third International Symposium on Autonomous Decentralized Systems* (*ISADS'97*), 9-11 April 1997, pp: 273 – 280, 1997.

[FrI01] Fritzke, U., Jr. Ingels, P. Transactions on partially replicated data based on reliable and atomic multicasts. Proceedings of *21st International Conference on Distributed Computing Systems*, April 2001, pp. 284 – 291, 2001.

[FuG05] Furnis, P. and Green, A. Choreology Ltd. Contribution to the OASIS WS-TX Technical Committee relating to WS-Coordination, WS-AtomicTransaction and WS-BusinessActivity. OASIS WS-TX Technical Committee, 16 November 2005. (avalaible at www.oasis-open.org/committees/download.php/15808/Choreology.WS-TX.TC.Contribution.2005-11-16.doc , last availability; 06.June.2006).

[FDF+04] Furnis, P.; Dalal, S.;Fletcher, T.; Green, A.; Ceponkus, A. and Pope, B. Business Transaction Protocol version 1.1.0, Committee Draft, November 2004. Available at: http://www.oasis-open.org/committees/download.php/9836/business_transaction-btp-1.1-spec-wd-04.pdf

[GaW01] Gallagher D. and Wilkerson. R., Network performance statistics for University of South Carolina. 2001. Available at: http://eddie.csd.sc.edu, [Last accessed 13 March 2007].

[G-MS87] Garcia-Molina, H. and Salem, K., Sagas. Proceedings of *Association for Computing Machinery Special Interest Group on Management of Data 1987 Annual Conference*, San Francisco, California, May 27-29, 1987, pp. 249-259, 1987.

[G-MG+91] Garcia-Molina, H.; Gawlick, D.; Klein, J.; Kleissner, K.; Salem, K. Coordinating activities through extended sagas: a summary. Proceedings of *Compcon Spring '91*. Digest of Papers (IEEE CNF), 25 Feb.-1 March 1991, pp. 568 – 573, 1991.

[GAP04] Gilbert, A. Abraham, A. Paprzycki, M. A system for ensuring data integrity in grid environments. *Proceedings of International Conference on Information Technology (ITCC'04): Coding and Computing*, Vol. 1, pp. 435 – 439, 2004.

[GiC00] Gillies J., Cailliau R. *How the Web Was Born: The Story of the World Wide Web*, Oxford University Press, England, 2000.

[GLR01] Gomez,S.M., Lo,S.H. and Rzhetsky,A. Probabilistic prediction of unknown metabolic and signal-transduction networks. *Genetics*, 159:1291-1298, November 2001.

[GNR03] Gomez S., Noble W. S. and Rzhetsky A., Learning to predict protein–protein interactions from protein sequences, *Bioinformatics*, 19(15):1875-1881, 2003.

[Gon01] Gong, L. JXTA: a network programming environment. *Internet Computing*, IEEE, 5(3):88-95, May/Jun 2001.

[Gud04] Gudgin, M. Secure, reliable, transacted: innovation in Web Services architecture. Proceedings of the *2004 ACM SIGMOD international conference on Management of data*. ACM Press, pp. 879-880, 2004.

[LLW+03] Guo Qiong Liao, Yun Sheng Liu, Li Na Wang. Chu Ji Peng. Concurrency control of real-time transactions with disconnections in mobile computing environment. Proceedings of *International Conference on Computer Networks and Mobile Computing (IEEE CNF)*, 20-23 Oct. 2003, pp: 205 – 212, 2003.

[Hag95] Haghjoo M. S. Transaction Actors in Cooperative Information Systems, PhD Thesis, The Australian National University, Australia, 1995.

[Hag96] Haghjoo M. S., Scheduling and Scripting Mega Transaction. Proceedings of *2nd International Computer Conference*, Iran Computer Society, Amir Kabir University, Iran, 1996.

[Hag97] Haghjoo M. S. Compensating Mega Transaction. Proceedings of *3rd International Computer Conference*, Iran Computer Society, Iran University of Science and Technology, Iran, 1997.

[HaP92] Haghjoo, M.S.; Papazoglou, M.P., TrActorS: a transactional actor system for distributed query processing. Proceedings of the *12th International Conference on Distributed Computing Systems* (IEEE CNF), 9-12 June 1992, pp: 682 – 689, 1992.

[HPS93] Haghjoo, M.S.; Papazoglou, M.P.; Schmidt, H.W. A semantic-based nested transaction model for intelligent and cooperative information systems. Proceedings of *International Conference on Intelligent and Cooperative Information Systems* (IEEE CNF), 12-14 May 1993, pp. 321 – 331, 1993.

[HST01] Herrero, J.L.; Sanchez, F.; Toro, M. Fault tolerance as an aspect using JReplica. Proceedings of *Eighth IEEE Workshop on Future Trends of Distributed Computing Systems* (FTDCS'01), 31 Oct.-2 Nov. 2001, pp: 201-207, 2001.

[HYP01] Heuvel, W-J Van; Yang, J. and Papazoglou, M.P. Service Representation, Discovery, and Composition for E-Marketplaces. Proceedings of *International Conference on Cooperative Information Systems* (cooPIS'01), September 2001.

[Hoa85] Hoare C. A. R. *Communicating Sequential Processes*. Prentice Hall, 1985.

[H-RC02] Hong-Ren Chen; Chin, Y.H. An efficient real-time scheduler for nested transaction models. Proceedings of *Ninth International Conference on Parallel and Distributed Systems* (IEEE CNF), 17-20 Dec. 2002, pp: 335 – 340, 2002.

[HBT04] Hwang, I. Roy, K. Balakrishnan, H. and Tomlin C. A Distributed Multiple-Target Identity Management Algorithm in Sensor Networks. Proceedings *43rd IEEE Conference on Decision and Control*, Atlantis, Paradise Island, Bahamas, December 2004.

[JSM01] Janaki Ram, D.; Chandra Sekhar, N.S.K.; Uma Mahesh, M. A data-centric concurrency control mechanism for three tier systems. Proceedings of *IEEE International Conference on Systems, Man. and Cybernetics*, Volume: 4 , 7-10 Oct. 2001, pp. 2402 – 2407, 2001.

[JTK89] Jenq, B.-C.; Twichell, B.C.; Keller, T.W. Locking performance in a shared nothing parallel database machine. *IEEE Transactions on Knowledge and Data Engineering*, 1(4): 530-543, Dec 1989.

[Kan01] Kan, G., *Gnutella, Peer-to-Peer: Harnessing the Power of Disruptive Technologies*, A. Oram (ed.), O'Reilly Press, USA.

[KaX92] Kakeshita, T.; Haiyan Xu Transaction sequencing problems for maximal parallelism. Proceedings of *Second International Workshop on Transaction and Query Processing* (IEEE), 2-3 Feb. 1992, pp: 215 – 216, 1992.

[KWR+02] Karev G., Wolf Y., Rzhetsky A., Berezovskaya, F. and Koonin E., Birth and death of protein domains: A simple model of evolution explains power law behavior, *BMC Evolutionary Biology*, 2:18, 14 October 2002.

[KuR03] Kurose, J. F. & Ross, K. W. *Computer Networking: A Top-Down Approach Featuring the Internet*, Addison Wesley, Boston, USA, 2003.

[LaP01] Lala, C.; Panda, B. Evaluating damage from cyber attacks: a model and analysis. *IEEE Transactions on Systems, Man and Cybernetics*, 31(4): 300-310, July 2001.

[LiT96] Liang, D.; Tripathi, S.K. Performance analysis of long-lived transaction processing systems with rollbacks and aborts. *IEEE Transactions on Knowledge and Data Engineering*, 8(5): 802-815, Oct. 1996.

[Lie06] Lieberman, B. SOA transaction management -- Part I: The transaction coordination service. Developer Works on SOA and Web services at IBM Ltd,15 May 2006. Available at: http://www-128.ibm.com/developerworks/rational/library/may06/lieberman/

[LiZ04] Limthanmaphon, B. and Zhang, Y. Web service composition transaction management. Proceedings of the *fifteenth conference on Australasian database*, vol. 27, pp: 171-179, 2004.

[LiF03] Little, M. and Freund, T. A comparison of Web services transaction protocols. Developer Works on SOA and Web services at IBM Ltd., 07 October 2003. Available at: http://www-128.ibm.com/developerworks/webservices/library/ws-comproto/

[LoC04] Losee, R.M.; Church, L. Information retrieval with distributed databases: analytic models of performance. *IEEE Transactions on Parallel and Distributed Systems*, Volume: 15 , Issue: 1, Jan. 2004 pp:18 – 27, 2004.

[LCC+02] Lv, Q.; Cao, P.; Cohen, E.; Li, K. and Shenker S. Search and replication in unstructured peer-to-peer networks. Proceedings of the *16th international conference on Supercomputing*, 2002, pp: Pages: 84 – 95, 2002.

[Mad97] Madria, S.K. Concurrency control algorithm for an open and safe nested transaction model. Proceedings of 1997 *International Conference on Communications and Signal Processing* (IEEE CNF), 9-12 Sept. 1997, pp: 907 - 912 vol.2, 1997.

[Maz88] Mazurkiewicz, A. Basic Notions of Trace Theory. In de Baker, de Roever and Rozenberg, eds, *Proceedings of Linear time, Branching Time and Partial Orders in Logics and Models for Concurrency*, Lecture Notes in Computer Science, Vol. 354, pp. 285-363, Springer-Verlag, 1988.

[Mil01] Miller, J. (2001), Jabber: Conversational Technologies. *Peer-to-Peer: Harnessing the Power of Disruptive Technologies*. Andy Oram, editor. O'Reilly and Associates, Sebastopol, California. 2001.

[Mos85] Moss J. E. B. *Nested Transactions: an Approach to Reliable Distributed Computing*. MIT Press, USA, 1985.

[MoS04] Moschoyiannis, S. Shields, M. W. (2004) A Set-Theoretic Framework for Component Composition. *Fundamenta Informaticae* 59(4): 373-396, 2004.

[MSK05] Moschoyiannis, S., Shields, M. W. and Krause, P. J. Modelling Component Behaviour using Concurrent Automata. In *Proceedings ETAPS 2005 – Formal Foundations of Embedded Software and Component-Based Architectures (FESCA'05)*, Electronic Notes on Theoretical Computer Science, Vol. 141, pp. 199-220. Elsevier, 2005.

[Mos05] Moschoyiannis, S. (2005). Specification and Analysis of Component-Based Software in a True-Concurrent Setting. PhD Thesis, University of Surrey, 2005.

[MKS07] Moschoyiannis, S., Krause, P. J. and Shields, M. W. A True-Concurrent Interpretation of Behavioural Scenarios. In *Proceedings ETAPS 2007 – Formal Foundations of Embedded Software and Component-Based Architectures (FESCA'07)*, Electronic Notes on Theoretical Computer Science, Elsevier, 2007. *To appear*

[M-EGB98] Munoz-Escoi, F.D. Galdamez, P. Bernabeu-Auban, J.M. ROI: an invocation mechanism for replicated objects. Proceedings of *Seventeenth IEEE Symposium on Reliable Distributed Systems*, 20-23 Oct. 1998, pp: 29 – 35, 1998.

[MNS+04] Mustafa, M.D. Nathrah, B. Suzuri, M.H. Abu Osman, M.T. Improving data availability using hybrid replication technique in peer-to-peer environments. Proceedings of AINA 2004 *18th International Conference on Advanced Information Networking and Applications*, 2004, pp: 593 - 598 Vol.1, 2004.

[MRW+93] Muth, P.; Rakow, T.C.; Weikum, G.; Brossler, P.; Hasse, C. Semantic concurrency control in object-oriented database systems. Proceedings of *Ninth International Conference on Data Engineering* (IEEE CNF), 19-23 April 1993, pp: 233 – 242, 1993.

[NeC02] Nektarios, G. Christodoulakis, S. UTML: Unified Transaction Modeling Language. Proceedings of the *Third International Conference on Web Information Systems Engineering* (IEEE CNF), 12-14 Dec. 2002, pp: 115 – 126, 2002.

[Nod93] Nodine, M.H. Supporting long-running tasks on an evolving multidatabase using interactions and events. Proceedings of the *Second International Conference on Parallel and Distributed Information Systems* (IEEE CNF), 20-22 Jan. 1993, pp: 125 – 132, 1993.

[NoZ94] Nodine, M.H.; Zdonik, S.B. Automating compensation in a multidatabase. Proceedings of the *Twenty-Seventh Hawaii International Conference on Software Technology* (IEEE CNF), Volume: 2, 4-7 Jan. 1994, pp: 293 – 302, 1994.

[Pap03] Papazoglou M. P. Service-Oriented Computing: Concepts, Characteristics and Directions. Proceedings of *4th International Conference on Web Information Systems Engineering (WISE'03)*, Rome, Italy, 2003.

[PaG03] Papazoglou M.P. and Georgakopoulos, D. Service-Oriented Computing. *Communications of the ACM*, 46(10):24-28, October 2003.

[PDB+97] Papazoglou, M.; Dells, A.; Bouguettaya, A.; Haghjoo, M. Class library support for workflow environments and applications. *IEEE Transactions on Computers*, 46(6):673-686, June 1997.

[PDH+96] Papazoglou, M.P.; Delis, A.; Haghjoo, M.; Bouguettaya, A. Language support for long-lived concurrent activities. Proceedings of the *16th International Conference on Distributed Computing Systems* (IEEE CNF), 27-30 May 1996, pp: 698 – 705, 1996.

[PaH05] Papazoglou, M.P.; van den Heuvel, W.-J. Web services management: a survey. *IEEE Internet Computing*, 9(6):58-64, Nov.-Dec. 2005.

[PTD+06] Papazoglou M. P., Traverso P., Dustdar S., Leymann F. and Kramer B. J. Service-Oriented Computing Research Roadmap. In *Dagstuhl Seminar Proceedings 05462, Service-Oriented Computing (SOC)*, pp. 1-29, 2006. Available at: http://drops.dagstuhl.de/opus/volltexte/2006/524 [Last accessed: 19 Apr 2006]

[PaH99] Pavlova,E.; van Hung, D. A formal specification of the concurrency control in real-time databases. Proceedings of *Sixth Asia-Pacific Software Engineering Conference*, (APSEC '99), 7-10 Dec. 1999, pp: 94 – 101, 1999.

[Plo00] Plonka. D. Uw-madison napster traffic measurement. Available at: http://net.doit.wisc.edu/data/Napster/, March 9 2000.

[PoH01] Posselt, D.; Hillebrand, G. Database support for evolving data in product design. Proceedings of *Sixth International Conference on Computer Supported Cooperative Work in Design* (IEEE CNF), 12-14 July 2001, pp: 377 – 383, 2001.

[QXL02] Qiujing Li; Hong Xu; TingLong Liu Coordinating transaction model in CDBMS. Proceedings of the *7th International Conference on Computer Supported Cooperative Work in Design*, 25-27 Sept. 2002, pp: 421 – 425, 2002.

[RaN99] Ramampiaro, H.; Nygard, M. Cooperative database system: a constructive review of cooperative transaction models. Proceedings of *International Symposium on Database Applications in Non-Traditional Environments (DANTE '99)*, IEEE CNF, pp: 315 – 324, 1999.

[Raz99] Razavi A. R. Design and Implementation of Recovery Management for Mega Transaction and Model Implementation, MSC Thesis, Iran University of Science and Technology, Iran, 1999.

[RMK07a] A. R. Razavi, S. K. Moschoyiannis and P. J Krause. A Coordination Model for Distributed Transactions in Digital Business Ecosystems. Proceedings of *IEEE Int'l Conference on Digital Ecosystems and Technologies (IEEE-DEST'07)*. IEEE Computer Society, 2007.

[RMK07b] A. Razavi, P. Malone, S.Moschoyiannis, B.Jennings, P.Krause. A Distributed Transaction and Accounting Model for Digital Ecosystem Composed Services. Proceedings of *IEEE Int'l Conference on Digital Ecosystems and Technologies (IEEE-DEST'07)*. IEEE Computer Society, 2007.

[RKM06] A. R. Razavi, P. J .Krause and S. K. Moschoyiannis. DBE Report D24.5: DBE Distributed Transaction Model, University of Surrey, 2006.

[RFH+01] Ratnasamy, S.; Francis, P.; Handley, M.; Karp, R. and Shenker, S. A scalable content-addressable network. Proceedings of the *2001 conference on Applications, technologies, architectures, and protocols for computer communications*, 2001, pp: 161 – 172, 2001.

[Rip01] Ripeanu M. Peer-to-Peer Architecture Case Study: Gnutella Network. Proceedings of *First International Conference on Peer-to-Peer Computing (P2P'01)*, 2001.

[RLF02] Ripeanu, M.; Iamnitchi A. and Foster I. Mapping the Gnutella Network. *IEEE Internet Computing*, 6(1): 50-57, Jan/Feb 2002.

[RoD01] Rowstron A. and Druschel P. Storage management and caching in PAST, a large-scale, persistent peer-to-peer storage utility. Proceedings of the *eighteenth ACM symposium on Operating systems principles*, 2001, pp: 188 – 201, 2001.

[RzG01] Rzhetsky A. and Gomez1 S. Birth of scale-free molecular networks and the number of distinct DNA and protein domains per genome. *Bioinformatics*, 17 (10): 988-996, 2001.

[ScP03] Schmidt, C.; Parashar, M. Flexible information discovery in decentralized distributed systems. Proceedings of *12th IEEE International Symposium on High Performance Distributed Computing*, 22-24 June 2003, pp:226- 235, 2003.

[SHD+01] Segun, K.; Hurson, AR; Desai, V.; Spink, A. and Miller, LL. Transaction Management in a Mobile Data Access System. *Annual Review of Scalable Computing*, 2001.

[SEN06] Software Engineering for Service-Oriented Overlay Computers (SENSORIA). EU-FP6 IST Integrated Project. Available at: http://sensoria.fast.de/ [Last accessed: 19 march 2007].

[Shi85] Shields, M. W. (1985), Concurrent Machines. *Computer Journal*, 28:449-465, 1985

[Shi97] Shields, M. W. *Semantics of Parallelism.* Springer-Verlag, London, 1997

[SCW+04] Siau, K.L.; Chan, H.C.; Wei, K.K. Effects of Query Complexity and Learning on Novice User Query Performance With Conceptual and Logical Database Interfaces. *IEEE Transactions on Systems, Man and Cybernetics,* 34(2): 276-281, March 2004.

[Smi99] Smith, J.R. Distributing Identity. *Robotics & Automation Magazine*, *IEEE Publication*, pp: 49-56 - ISSN: 1070-9932, 1999.

[Sta03] Stallings W. *Cryptography and Network Security Principle and Practice*. Third Edition, Prentice Hall, USA, 2003.

[TUH+97] Tada, H.; Uchida, K.; Higuchi, M.; Fujii, M.28 A model of nested transaction with fine granularity of concurrency control. Proceedings of *IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*, Volume: 2 , 20-22 Aug. 1997, pp: 977 - 980 vol.2, 1997.

[Tan89] Tanenbaum, A.S. *Computer Networks*. Engleweood Cliffs, Prentice-Hall, 1989.

[TeI03] Terai, K. and IzumI, N. Coordinating Web Services based on business models. Proceedings of the *5th international conference on Electronic commerce,* vol. 50, pp:473-478, ACM Press, 2003.

[USD07] US Department of Health and Human Services. Fact Sheet: Isolation and Quarantine. *Department of Health and Human Services, Centres for Disease Control and Prevention*. Last accessed: Feb 2007. (http://www.cdc.gov/ncidod/dq/isolationquarantine.htm last access: 08/03/2007)

[VaP02] Valencia A. and Pazos F. Computational methods for the prediction of protein interactions. *Current Opinion in Structural Biology* 2002, 12:368–373, 2002.

[vdMDD+03] van der Meer, D. Datta, A. Dutta, K. Ramamritham, K. Navathe, S.B. Mobile user recovery in the context of Internet transactions. *IEEE Transactions on Mobile Computing*, 2(2):132-146, April-June 2003.

[VeP98] Verharen, E. M. and Papazoglou, M.P. Introducing Contracting in Distributed Transactional Workflows. Proceedings of *Thirty-First Annual Hawaii International Conference on System Sciences,* Volume 7, pp:324-334, 1998.

[VZG+05] Vogt, F.H.; Zambrovski, S.; Gruschko, B.; Furniss, P. and Green, A. Implementing Web service protocols in SOA: WS-Coordination and WS-BusinessActivity. Proceedings of S*eventh IEEE International Conference on E-Commerce Technology Workshops*, 19 July 2005, pp: 21- 26, 2005.

[WaS07] Y. Wang and M. Singh. Formal trust Model for Multiagent Systems. Proceedings of IJCAI'07, pp. 1551-1556, 2007.

[WaS06] Y. Wang and M. Singh. Trust Representation and Aggregation in a Distributed Agent System. Proceedings of *21st Conference on Artificial Intelligence (AAAI'06)*, pp. 1425-1430, 2006.

[Wan96] Wanlei Zhou Performance evaluation of nested transactions on locally distributed database systems. Proceedings of *Second International Symposium on Parallel Architectures, Algorithms, and Networks* (IEEE CNF), 12-14 June 1996, pp: 353 – 356, 1996.

[Wat99] Watanabe, T. Agent-oriented model for managing long-lived transaction, based on work-flow and task-graph. Proceedings of *Third International Conference on Computational Intelligence and Multimedia Applications ICCIMA '99* (IEEE CNF), 23-26 Sept. 1999, pp: 10 – 13, 1999.

[WaS98] Watts D. J., Strogatz S. H., Collective dynamics of small-world networks. *Nature* 363: 202-204, 1998.

[WDN02]Watts, D. J., Dodds, P. S. and Newman, M. E. J. Identity and Search in Social Networks. *Science*, 296:1302-1305, 2002.

[WWZ02] Wgrzyn, S.W. Winiarczyk, R. Znamirowski, L. Nanotechnologies and nanosystems of informatics as a basis for self-replication. Proceedings of the *2nd IEEE Conference on Nanotechnology, IEEE NANO'02,* pp: 99 – 102, August 2002.

[Wik06] Wikipedia. 'Web service'. Wikipedia online encyclopaedia, 2006. Available at: http://en.wikipedia.org/wiki/Web_service (accessed date; 14/06/06)

[WZB+01] Xiao Weijun; Lu Zhengding; Li Bing; Sarem, M. Transaction management in mobile multidatabase systems. Proceedings of *2001 International Conference on Computer Networks and Mobile Computing*(IEEE CNF), 16-19 Oct. 2001, Los Alamitos, CA USA, pp: 513 – 518, 2001.

[YaG02] Yang, B. and Garcia-Moline, H., Designing a Super-Peer Network. Technical Report, Standford University, February 2002.

[YPH02] Yang, Jian; Papazoglou, M P. and Heuvel, W-J van den. Tackling the Challenges of Service Composition in E-Marketplaces. Proceedings of *Twelfth International Workshop on Research Issues in Data Engineering: Engineering E-Commerce/E-Business Systems* (RIDE-2EC 2002), IEEE Computer Society, pp:125-133, 2002.

[YaM03] Yaohang Li; Mascagni, M. Improving performance via computational replication on a large-scale computational grid. Proceedings of *(CCGrid 2003) 3rd IEEE/ACM International Symposium on Cluster Computing and the Grid*, 12-15 May 2003, pp: 442 – 448, 2003.

[YK-MA+02] Younas, M. Kuo-Ming Chao Anane, R. James, A. Multi-agent transactions for Web-based design activities. Proceedings of *7th International Conference on Computer Supported Cooperative Work in Design*, 25-27 Sept. 2002, pp: 258 – 263, 2002.

[YSS04] Yu B, Signh M. and Sycara K. Developing Trust in Large-Scale Peer-to-Peer Systems. Proceedings of *First IEEE Symposium on Multi-Agent Security and Survivability*. IEEE, pp. 1-10, 2004.

[YLY03] Yu, S. Zhou, W. Lan, M. Yue Wu. An architecture of Internet based data processing based on multicast and anycast protocols. Proceedings of (*PDCAT'2003) Fourth International Conference on Parallel and Distributed Computing*, 27-29 Aug. 2003, pp: 104 – 110, 2003.

[ZLB04] Zirpins, C.; Lamersdorf, W. and Baier, T. Flexible coordination of service interaction patterns. Proceedings of the *2nd International Conference on Service-Oriented Computing,* pp. 49-56, ACM Press, 2004.

[ZoJ98] Zou H. Jahanian, F. Optimization of a real-time primary-backup replication service. Proceedings of *Seventeenth IEEE Symposium on Reliable Distributed Systems*, 20-23 Oct. 1998, pp: 177 – 185, 1998.