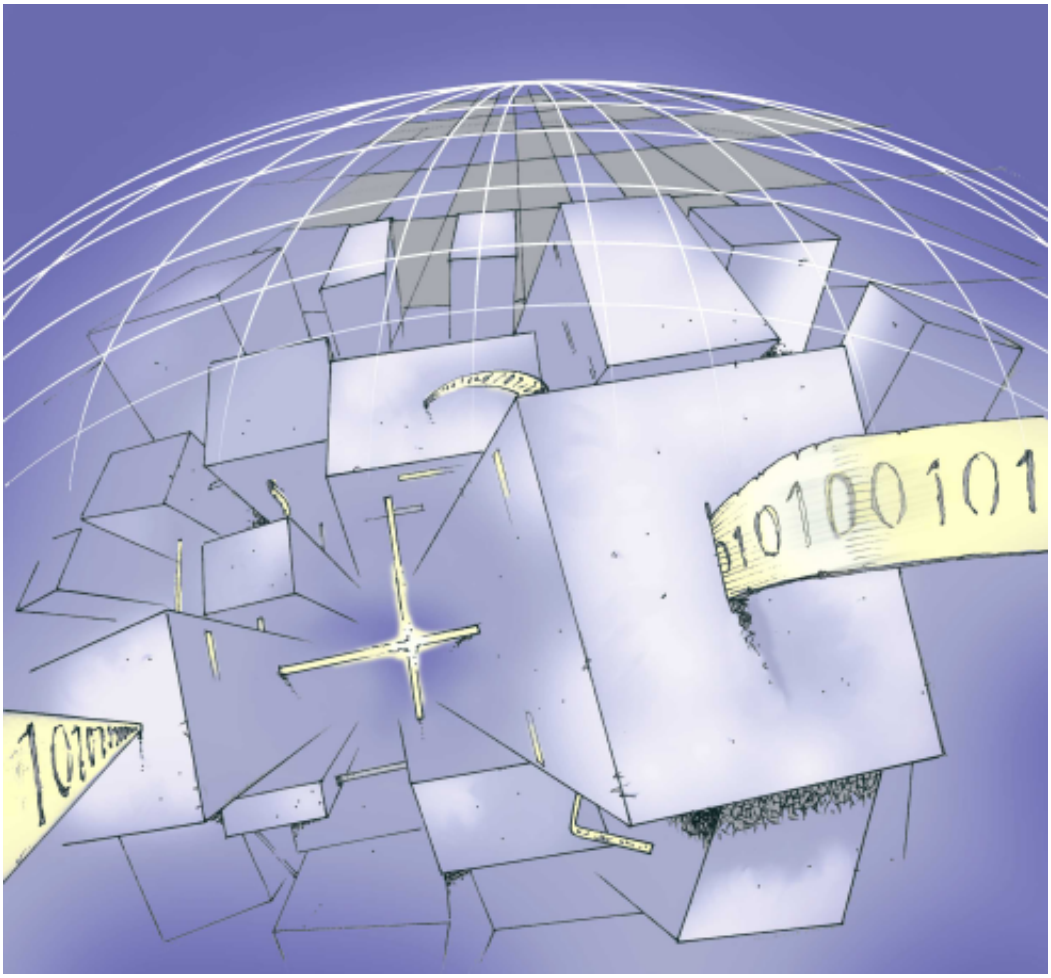
ESPOO 2002 VTT RESEARCH NOTES 2162



Mika Hongisto

Mobile data sharing and high availability

VTT TIEDOTTEITA – RESEARCH NOTES 2162

Mobile data sharing and high availability

Mika Hongisto
VTT Elektroniikka

ISBN 951–38–6081–7 (soft back ed.)
ISSN 1235–0605 (soft back ed.)

ISBN 951–38–6082–5 (URL: http://www.inf.vtt.fi/pdf/)
ISSN 1455–0865 (URL: http://www.inf.vtt.fi/pdf/)

JULKAISIJA – UTGIVARE – PUBLISHER

VTT, Vuorimiehentie 5, PL 2000, 02044 VTT
puh. vaihde (09) 4561, faksi (09) 456 4374

VTT, Bergsmansvägen 5, PB 2000, 02044 VTT
tel. växel (09) 4561, fax (09) 456 4374

VTT Technical Research Centre of Finland, Vuorimiehentie 5, P.O.Box 2000, FIN–02044 VTT, Finland
phone internat. + 358 9 4561, fax + 358 9 456 4374

VTT Elektroniikka, Kaitoväylä 1, PL 1100, 90571 OULU
puh. vaihde (08) 551 2111, faksi (08) 551 2320

VTT Elektronik, Kaitoväylä 1, PB 1100, 90571 ULEÅBORG
tel. växel (08) 551 2111, fax (08) 551 2320

VTT Electronics, Kaitoväylä 1, P.O.Box 1100, FIN–90571 OULU, Finland
phone internat. + 358 8 551 2111, fax + 358 8 551 2320

Otamedia Oy, Espoo 2002

Hongisto, Mika. Mobile data sharing and high availability [Tiedonhajauttaminen langattomassa ympäristössä]. Espoo 2002. VTT Tiedotteita – Research Notes 2162. 102 p.

Keywords weak connections, mobile replication, smart data distribution, mobile distributed systems

Abstract

This work introduces a model for decentralised and platform-independent data sharing for highly dynamic networks. The goal is not to deliver a solution that can be considered similar to database systems. Communication and replication challenges are presented and examined from the perspective of nomadic computing. The aim has been to provide highly available data sharing over features over mobile distributed systems.

The primary functionality of the data-sharing model introduced here has been validated with simulations. A fully functional data-sharing model has been implemented and tested in a simulated environment. A simulated environment was chosen over real network conditions in order to offer data distribution pattern tracking in several different network topologies and environments.

This study provides understanding of the data sharing factors that are important in decentralised distribution models. These factors include location determination for replica storing and control message propagation over weak connections. Smart data distribution is crucial for weakly connected networks to offer access to shared resources. In heterogeneous environments, it is also necessary to consider possible transmission resource savings, and an idea for partitioning data into small elements is introduced. A data distribution algorithm based on this understanding is designed and evaluated.

According to technology surveys and the evaluation of data sharing implementation, it is clear that the optimistic replication schema is the choice for data distribution over weakly connected networks. This requires conflict solving for eventual consistency, or any equivalent method to provide the correct operation for applications. A decentralised data management model is the best alternative to operate in weakly connected networks, as it is able to function regardless of network partitions.

The data sharing approach presented here provides full adaptability to different network topologies and computing platforms, and it is able to offer data sharing services for any device to some degree. A decentralised data sharing approach introduces new challenges; these are discussed, and their viability for implementation is estimated.

Hongisto, Mika. Mobile data sharing and high availability [Tiedonhajauttaminen langattomassa ympäristössä]. Espoo 2002. VTT Tiedotteita – Research Notes 2162. 102 s.

Avainsanat weak connections, mobile replication, smart data distribution, mobile distributed systems

Tiivistelmä

Tässä työssä esitellään malli tiedonhajautukselle, joka on suunnattu dynaamisiin ja heikosti toisiinsa kytkettyihin tietoverkkoihin. Kantavana ajatuksena ei ole luoda perinteiseen tietokantasovellukseen rinnastettavaa toteutusta, vaan alustariippumaton malli, jonka tärkein kohdeympäristö on langattomat ja mukana kulkevat laitteet. Kyseiseen toteutukseen liittyen tarkastellaan kommunikointi- ja replikointiteknologioita erilaisten toteutusmahdollisuuksien arvioimiseksi. Suoritettuun arviointiin perustuen esitellään valitut toteutusteknologiat, joilla pyritään korkeaan tiedon saatavuuteen ja käytettävyyteen kannettavissa päätelaitteissa.

Pääasiallinen toiminnallisuus esiteltävässä tiedonhajautusmallissa on varmistettu simulaatioilla. Täysin toimiva tiedonhajautusmalli on implementoitu ja testattu simuloidussa ympäristössä. Simuloitu ympäristö valittiin todellisen maailman verkko-ympäristön sijasta, jotta pystyttiin paremmin seuraamaan tiedonhajautusjakaumia ja tilanteita erilaisissa verkkotopologioissa ja ympäristöissä.

Tämä diplomityö tarjoaa ymmärtämystä keskittämättömän tiedonhajautusmallin toteutuskonsepteihin ja ominaisuuksiin. Tärkeimpänä osana tätä mallia on älykäs tiedonhajautusteknologia, joka esitellään yksityiskohtaisesti. Älykäs tiedonhajauttaminen on tärkeää keskittämättömässä ja heterogeenisessa ympäristössä, kuten työssä tuodaan esille. Myös kommunikaatioteknologia on tärkeässä roolissa, joten erilaiset tiedonsiirron lähestymistavat ovat vertailussa. Tiedonsiirtoresurssit ovat hyvin vaihtelevia mobiileissa ympäristöissä, minkä johdosta esitellään uniikki ajatus tiedon paloittelemisesta pieniin osiin.

Tämä työ osoittaa, että optimistinen tiedonreplikointimalli on oikea lähestymistapa, kun pyritään korkeaan tiedon käytettävyyteen heikosti kytketyissä verkoissa. Kyseinen replikointimalli vaatii seurakseen teknologian, jolla tiedon semantiikan säilyminen voidaan taata sovelluksille. Keskittämätön kontrollinhajautus on valittu toteutusteknologiaksi siksi, että se kykenee toimimaan verkkojen jakaantumisista huolimatta.

Esitetty tiedonhajautusmalli tarjoaa hyvän mukautumiskyvyn erilaisiin verkko-ympäristöihin riippumatta tiedonkäsittelyalustoista. Tavoitteena on taata mahdollisuus

tiedon käyttämiseen ja muokkaamiseen ilman yhteyttä mihinkään tiettyyn keskitettyyn palveluntarjoajaan. Esitetty hajautusmalli nostaa esiin uusia haasteita ja ongelmia, jotka vaativat teknologialta harppauksia. Lopuksi, esitetyn hajautusmallin käyttökelpoisuus arvioidaan.

Preface

Most of this study has been done in a project under International ITEA Workshop on Virtual Home Environments. ITEA (Information Technology for European Advancement) projects are funded by National Governments with different national schemes for co-operation among countries in Eureka. The Eureka project is dedicated to strengthening European software and software engineering, and the programme has a life span of eight years with a total of over 15 000 developer-years over the next years. The final editing of this work has been done under the PLA-project, which has led to the use of QADA concepts in some parts of this study.

The VHE-project at VTT will finish this summer, and in the same way, this diploma thesis will be finish and reach publication. Work on this thesis has been taking place for several months now, while it has been quiet from time to time when other interests were given higher priority. Sometimes, writing it has felt like grasping for eternity, straining to mould vivid thoughts and visions into the narrow abstraction space of human language. Nevertheless, from the writer's perspective, this thesis is quite successful in describing technologies related to mobile distribution and data sharing, and it provides some unique ideas with varying success.

This work would not have been possible without encouraging support from several individuals. Support can come in many forms, for example, help regarding some problem or a task, an inspiring discussion or supportive mentality. The highest praise goes to Eila Niemelä as an immediate superior, for giving the writer the chance to write this thesis. She has also made a lot of effort in reading through drafts and providing excellent hints for writing. The supervisor of this study has given constructive feedback on the study, which has opened my eyes and made me able to see and be able to correct the major flaws in the thesis, hopefully. Also, special thanks are appropriate for those who have made it possible for me to graduate within a relatively tight schedule. Therefore, a deep bow is given for Juha Röning, as a personal show of gratitude.

Family support during the writing of this thesis has been encouraging, and I want to thank my entire family for providing opportunities for relaxation. And it is not possible to forget two lovely hounds showing disrespect towards my papers, and I would like to thank these beasts for refreshing accompany. Thanks to my friends for giving me a chance to show that I was alive during the dark nights of the winter, and for providing official information stating that graduating is not possible. Alas, I have shown otherwise and welcome them to join our ranks. Last, but not least, special thanks to Mira for staying with me all through the difficult moments during my years of study.

Oulu, 19.5.2002

Mika Hongisto

Contents

Abbreviations

DDP	Data Distribution Pattern
GPRS	General Packed Radio Service
GPS	Global Positioning System
IP	Internet Protocol
ITEA	Information Technology for European Advancement
PDA	Personal Digital Assistant
PLA	Product Line Architectures
QADA	Quality-Driven Architecture Design Analysis
ROWA	Read One-Write All protocol
RAWA	Read All-Write All protocol
SDDA	Data Distribution Algorithm
SDS	Spontaneous Data Storage
UML	Universal Modelling Language
VHE	Virtual Home Environments
VTT	Valtion teknillinen tutkimuskeskus
WLAN	Wireless Local Area Network
XML	Extensible Mark-up Language

1. Introduction

Current trends and the quick developing cycles of high technology devices are bringing new, unforeseen challenges into everyday computing environments. In recent history, primary utilisation of computing resources has been shifting from everyday working use towards leisure interests. This direction is strongly evident in handheld and mobile devices and is one of the motivating issues of this study.

Nowadays, consumers have mobile phones in everyday use and some technology enthusiasts are even playing chess with their brand-new PDAs. This shows a high level of acceptance for mobile computing devices among end users. Many of the current high-end portable devices offer wireless transmission capabilities, and in the short-term future when gadgets for virtual home environments are pushed towards end users, it is necessary to create possibilities to deliver information over different infrastructures. This is especially true for mobile systems.

The work of this study is based on the needs of a previously implemented distributed service platform in VTT Electronics [1] that is focused towards the distribution of services over spontaneous networks. Spontaneous in this context means connections between arbitrary nodes, and not especially wireless mobile networks. It is a connection that is created spontaneously by any human or non-human interaction.

The study provides support for the distributed service platform, as during experiments it became clear that it is necessary to create new data storing features to support the maintenance and provision of services. The fundamental idea of this study is to analyse and solve problems related to data sharing and distribution over weakly and dynamically connected networks. The design goal is to establish data sharing features that fulfil the needs of a distributed platform, while tailoring it also to be suitable for the data sharing needs of some other environments. The primary target environment is mobile distributed systems, as it is one of the most spontaneous in nature.

A *mobile distributed system* is a heterogeneous collection of independent computing based devices linked by a dynamically evolving network and middleware designed to provide transparent access for users over shared computing resources. It allows different combinations of hardware platforms, varying from a couple of PDAs to spontaneous networks of several laptops connected through networks to share computing resources.

Middleware is considered to provide a common way towards full diversity of software, to enable applications to share structure, interconnectivity and common functionality [2]. When considering the position of middleware in the open system interconnect model, it is loosely between network and application layers.

This study concentrates on platform independent data storage and distribution over small-scale and spontaneous networks, with little or no fixed points. The goal is to create co-operation between different weakly connected gadgets while providing data access services of decent quality. Several challenges are met in the designing of the Spontaneous Data Storage to fulfil the needs of small-scale mobile data sharing. In the following chapters, therefore, this study focuses on quality requirements, weak connections, mobile replication and smart data distribution.

2. Challenges for mobile distributed systems

Several challenges are apparent in mobile distributed systems that differentiate the systems from traditional distributed systems. Possible design approaches to mobile distributed systems are various, and more than one approach may be suitable for desired solutions. This chapter presents the challenges faced while designing the Spontaneous Data Storage. The Spontaneous Data Storage is a manifestation of our vision of the mobile data-sharing concept. The primary principle in implementing the vision has been to allow devices with variable resources to take part in the creation of a distributed data sharing system to the extent allowed by their resources and usage targets. The design takes a down-top approach, where the data-sharing concept is designed according to the restrictions and possibilities posed by different technologies. In the following chapters we discuss available resources and limiting technology, which have been guiding the development of Spontaneous Data Storage.

2.1 Scaling and adaptability

Mobile platforms have limited and highly heterogeneous computing and transmission resources, and the typical behaviour of intermittent connections due to battery savings accompanied by high mobility does not ease either of these limitations.

The creation of interconnections between different architectures and network topologies from local and spontaneous networks to global-scale, world-wide networks without sacrificing the high consistency, reliability and scalability of the typical distributed environments is a major challenge for the distributed computing world to face. Where classical distributed environments are braced against fixed and centralised servers, it is not necessarily the only viable route. This study focuses on the design of a data sharing concept with self-adaptive configurations and distributed replication controls over networks.

At the dawn of mobile computing when de-facto industrial standards are still evolving, it is important to introduce a bonding set of fundamental services for a wide range of diverse devices. The creating of a distribution system which offers flexibility to connect any device to another device – if any physical transfer connection is reachable – in such a way that applications are able to utilise shared computing resources is a great step toward universal computing. This is a study to research and present various manifesting problems while designing a model of shared data access for highly dynamic environments.

2.2 Data availability

When dealing with research in the wide area of different technologies, as with distributed systems, it is necessary to maintain the big picture of the whole concept. Application of new technology is in a major role in defining introduced features and fulfilling the quality of service requirements. In this study, quality of service is used to promote the level of availability, performance and fault tolerance of data access guaranteed for applications working with the distributed mobile system. The main concern is to enable high quality data delivery and storing services for mobile devices interacting with wired networks, while satisfying the interconnecting and data sharing needs of distributed mobile computing.

2.3 Mobility and displacement

Classically, network environments have been built over fixed wired and wireless connections. It has not been possible to create spontaneous connections between different devices. Manual configuration and centralised service providers have enabled people to join into networks. However, this is no longer true with the current generation of handheld and portable devices as they are able to generate spontaneous networks in situations where several uniform devices are present. The problem is that the technologies that provide a foundation for spontaneous networks to function have not reached the level of scaling for diverse platforms that is commonly seen in preconfigured systems.

Mobility introduces challenges into networking, as traditional transmission techniques are based on fixed routing paths, and do not provide support for changes in network topology. If a device is displaced into another location in a network, still remaining in the range of other devices, the route that information has to take to achieve its destination may change. There is hardly a way to predict the movement of network nodes, and therefore, connections might have highly spontaneous characteristics. Weak connections are the causative reason for routing and message delivering problems in nomadic computing.

Spontaneous operation in networks implies, at least, the possibility of establishing and losing of connections. It can also be interpreted to include displacement of devices, as moving of a node in a network practically means that some nodes will lose connections while some others will establish them. A temporary absence of a transaction continuum of the existing network caused by the dozing of a battery powered handheld device leads also to disconnection. In general, mobility equals to weak connections. Data

transfer technologies are detailed in Chapter 4, and a mobile perspective is effective in the background when dealing with general considerations.

2.4 Disconnected operation

Disconnected operation provides great value for handheld and portable computing devices. Document editing is an example of the usage target. In a moment of inspiration one could take his laptop and make some modifications on a document, even if the document is not the newest revision. And after moving back to office the updated information is propagated from the laptop and integrated to newer versions. This would provide a modified document of the newest version, and there would be no need for manual intervention. This would also work with several people working on the same documentation, and would provide seamless integration of updates. At least the writer of this thesis would have been grateful of such an outstanding feature.

2.5 Heterogeneity of computing platforms

Technology of handheld devices is evolving at tremendous speed. There are several competing proprietary approaches to provide an expanding variety of features for end users. This competitive situation is advantageous for customers, while it leads to heterogeneous computing and to data transfer technologies and resources between different products.

The recent movement toward wireless transmission standards provides possibilities for cross talk between these diverse platforms. There is no commercial technology for communication between competing products available at the moment, and such technology is necessary for the Spontaneous Data Storage under design. Therefore, this study provides a communication layer on top of the existing transfer protocols in order to provide uniform and platform independent messaging abilities for users.

There is also a wide variety of data storing resources, transmission bandwidth and computing power among different products. A comparison between handhelds and laptops aggravates the situation. As every device does not have similar resources to take responsibility for data sharing services, it is necessary to divide the burden efficiently between hosts. Heterogeneity of platforms is one of the guiding forces behind the design of the Spontaneous Data Storage.

2.6 Data sharing over weak connections

Data sharing over the network is an old and thoroughly researched field. It is not a new idea to share data over weak connections either, and there are some good papers about it for database use [3][4]. These approaches are designed for database use and they take conventional and tested technologies as foundry to provide robust operation. Classical approaches are extended with some new technologies to provide functionality for network failures, or partitions. As in classical databases, they use centralised control models. In this research, a more radical approach is taken and a decentralised control model is used to provide the Spontaneous Data Storage.

Classical data sharing systems are based on an assumption that connections are reliable, and losing a connection is considered a faulty operation. Intermittent connections are characteristic of mobile devices, which leads to the conclusion that networks are not reliable, and the classical approach might not be the best alternative. A possibility to access information in a disconnected situation would certainly improve availability, even though it would bring along problems as well.

This study approaches the data sharing challenge by means of both smart information distribution and efficient and viable communication technologies for data control. Replication is the key for providing high availability, and therefore, replication approaches are analysed in Chapter 5. To make Spontaneous Data Storage a viable alternative, it is necessary to introduce some new technologies. Some approaches to fulfil required functionality are discussed in Chapter 6.

3. Evaluation framework to technologies

To justify different technologies, it is necessary to create an evaluation framework to make concrete comparisons between different approaches. Therefore, necessary terms are defined to provide a framework for evaluating replication and communication technologies. More terms are defined as needed in appropriate contexts.

3.1 Evaluation of communication

Consensus of communication methods lays foundation for shared computing resources. Many problems of heterogeneous environments are brought together, as described in Chapter 2 where the scope of this research is defined. In this section, the necessary communication terms are defined before considering the problem domain of communication.

3.1.1 Definitions for communication

Development in communication technologies has lead to a rich and diverse communication terminology. Table 1 introduces several key terms that are important for this study.

Table 1. Definitions of communication.

Term	Definition
Weak connections	Weak connections [5] imply that communication links are not reliable, and communication failures are part of the natural behaviour of the participating hosts. In general, short range wireless communication and other hosts with intermittent connections belong to this category
Strong connections	Strong connections [5] imply that communication links are failure-proof, and data delivery is reliable. This is the characteristic of fixed networks
Mobility	Mobility implies free movement of device, without restricting speed, direction or moment of time
Unicast	Unicast is communication between a single sender and a single receiver over a network
Multicast	Multicast is the sending of messages from one source to several destination hosts subscribed to the delivery list. It is possible to verify message delivery if backchannel for communication is provided
Broadcast	Broadcast is the sending of messages from one source to every host in the network. There is no possibility to specify the recipients, and neither is there a guarantee of message delivery to every host

The terms from unicast to broadcast are quite common when discussing about Internet and closely related technology. Strong and weak connections deal mostly with hardware interface into environment, and mobility is a loose term that can be emphasised differently in varying contexts. The definitions in the table provide the exact meaning for the usage of detailed terms in this study.

3.1.2 Evaluation model for communication

Communication between hosts is a fundamental part of the distributed systems. A data sharing system over mobile hosts is not an exception, and requires efficient communication techniques over a wide array of different platforms. The situation is not free of problems and leads to several compromises. The goal of this evaluation model is to make efficient use of available transmission resources without disabling any general usability of devices. Following factors are chosen and established for this study.

- It is necessary to make possible the dozing mode of handheld devices. Many handheld and portable devices have power saving features that shut down wireless network connections, or the whole device. If dozing is not allowed, the standby and operation time of the device is greatly decreased.

- Transfer resources between handheld, portable and fixed hosts are far from uniform. Efficient utilisation of available resources is considered to prevent the overloading of connections for low power devices.

- Dozing and intermittent connections due to displacements of hosts requires mechanisms to predict a good time window for data transfer and control signals.

- Different approaches for data dissemination are suitable for several different application targets. Different dimensions of data deliver are examined to find viable alternatives.

These variables provide a strainer that is used to sieve through different existing communication approaches for grains of information to tailor a message passing method suitable for data sharing. The overall topic of communication is surveyed in Chapter 4.

3.2 Evaluation of replication

To gauge the edges and flaws of different data sharing approaches, this study presents a framework for the comparing of the quality of important features from a mobile

perspective. The three most important elements are availability, reliability and adaptability. The goal is to provide decent availability while sustaining the semantics of information; reliability is thereby important. Adaptability is crucial for dynamically evolving environments to provide connectivity and reasonable use of shared resources. The following sections define terms for replication and introduce a model for the previously mentioned comparison.

3.2.1 Definitions for data replication

Database applications and data replication technology is a widely researched field, and its terminology is standardised and consistent. Table 2 provides basic definitions for database terminology that are relevant for this study.

Table 2. Definitions of database terminology.

Term	**Definition**
Consistency	Consistency defines the semantics of accessible information, as it determines the coherence of available data units. In other words, data consistency is a guarantee for all of the accessible data units to be copies of the newest versions
Fault tolerance	Fault tolerance determines the probability for proper functioning of a system. Fault tolerant operation states that information is always available and consistent, even if arbitrary failure events are met in the network, and an application or a user does not have any problems whatsoever in using shared resources
Replication transparency	Replication transparency is one of the general requirements of data sharing. It is not a concern of clients to be aware of the multiple physical copies of the information that exists. They only see one logical item of the information they seek. Operations on that item return only one set of values, even if operation is performed upon several physical copies
Location transparency	Location transparency guarantees that it is not on the responsibility of clients to be aware of any locations of the data items and physical locations of the items are hidden from users
Serialisability	Serialisability states that the concurrent execution of atomic transactions is equal to the serial execution of the same transactions
Atomic operations	Atomic operations guarantee that accessed information is consistent, independent of interference from concurrent operations being performed in other places. Atomic transactions consist of one to several atomic operations, which implies that all or none of the operations are performed when conflicting transactions are detected

The primary environment for database systems consists of fixed and centralised database servers. The ideology behind this study is not focused in providing a real database solution, but a data sharing system that increases data availability in hostile environments, where hostile refers to a difficult environment for computing devices to communicate and co-operate. Therefore, it is necessary to define the key terms for data distribution to suit the context. Table 3 provides the primary terms used in the comparison of different replication technologies.

Table 3. Definitions of quality attributes.

Attribute	**Description**
Availability	Availability determines the probability for the requested data to be ready for access. Indirectly it also includes performance and responsivity of data access. If a data access has high latency or a communication link cannot scoop data packets with decent throughput, the availability of information will degenerate. In this context, high availability states that data remains accessible independently of any communication failures and reasonable access times are preserved despite of high resource utilisation
Reliability	Reliability in this context determines the level on *consistency* and *fault-tolerance* of the data access. It is the rating for general solidity of the approach for replication in highly dynamic networks, with intermittent connections and conflicting updates. High reliability means, with high probability that the semantics of available information does exist
Adaptability	Adaptability determines, for this context, the level of flexibility the system has to self-adjust configurations to sustain decent operation in dynamic environments. High adaptability implies that the system does not need any preconfigured information or manual intervention for the rational control of data flow and shared resources

3.2.2 Comparison model for replication approaches

Tables are used to provide concreteness and easy readability of comparisons. They consist of four rows and two columns with information regarding the evaluation of evaluated technology, and these fields are shaded with grey backgrounds in the example in Table 4. The six fields with the white backgrounds are identical in every chart, as in the example. The implementation field is reserved for a short description of the method that is used to provide the ability stated on the left, and success is a discrete evaluation of viability to fulfil the requirements for the case study. Success rating is a number scale from one to five, and the list of verbal counterparts is the following: poor, low, average, good, and excellent. More accurate descriptions for each comparison factor will be listed later.

Table 4. Example of a comparison chart.

Attribute	Implementation method	Success
Availability	Read access is provided for available information. Update access is granted, if the user has access to weighted majority of replica managers	Low
Adaptability	Assignments of weights are static, and it is possible that none of the groups gains the majority status	Low
Reliability	Only one group at time can have the majority, hence conflicting updates are not possible	Excellent
Overall	Can tolerate some network partitions, but cannot operate in fully dynamic situations	Average (2.9)

This grading system is appropriate for comparing data sharing models for the Spontaneous Data Storage. The reasons why availability, reliability and adaptability are important and why they provide necessary weights for the overall grading are illustrated in the following.

3.2.3 Evaluation criteria

The numerical approach that is used in the replication technology comparisons in this study consists of grading, weights and overall values for viability in the studied context. The primary goal is to investigate mobile data sharing technology for highly available data access. To allow numerical evaluation and comparison, it is necessary to create discrete boundaries and categorisations to allow grading. The following subsections provide grading criteria for availability, adaptability and reliability on a scale from one to five, which are the quality factors under investigation.

The goal in the comparison is to achieve an overall grade for the suitability of technology in the studied context for every replication approach. As all of the evaluation criteria are not equally important, it is necessary to choose balanced weight-values for the quality factors. The traditional database solution would embrace reliability over everything else, but this is not the case with our approach. The goal is to provide high availability and decent operation in dynamic situations. High reliability and availability do not co-exist over weak connections simultaneously, and therefore we have carefully tried to balance weight-values to depict our emphasis.

Availability

Highly available shared information for spontaneous environments is the primary goal in this study, and therefore, it is the main factor that determines the value of the system. Availability of the data access is classically divided into two transactions: read requests and update requests. As these requests utilise different amounts of network resources and lead to distinct effects on the managing of the replica system, they are usually enforced in very varying methods. This disparity is the reason why, in general, update requests cannot rival read requests at the level of availability. Therefore, in this evaluation framework, estimation is given only for update transactions, as they determine the lowest possible availability. High availability is the most important factor, and deserves a weight value of 40%.

- Poor grade is given for replica models that offer the weakest possible availability. It is necessary to contact every replica manager before it is possible to update data.
- Low availability is a step forward, and thus, it is possible to access data without the participation of every replica manager. Some fixed group or groups have the authority to grant data update access.
- Average grade requires even more complexity and availability. It provides access to data even if some changes are made into the environment. This category includes models that have a dynamic group membership for the authoring of data.
- Good availability means that it is possible to utilise data in dynamic network partitionings, excluding the most hostile situations.
- Excellent availability states that data is always available for access, if any copy can be found. This is the highest availability ranking.

Adaptability

In mobile environments where displacements of hosts and unreliable connections play a dominant role, the network is constantly in an evolving stage. It is not feasible, and not even possible, to make configurations that suit every situation. Manual intervention could be used to modify configurations to provide temporal operability, but they are slow and expensive to use. And it is worth remembering that common users are not experts to manage their computing devices, and therefore, automatic adaptability to the changes in an environment is crucial. Adaptability is almost equally important for functionality, and deserves a weight value of 30%.

- Poor adaptability is used to describe a system, which is fully preconfigured and does not adapt to any changes in the environment. Every connection and behaviour of a system is determined by manual configuration.

- Low adaptability states that a minor development in the environment is allowed and does not hinder the functionality of a system. The control model is still preconfigured.
- Average adaptability denotes a minor ability to change the behaviour pattern according to changes in environment. It provides some self-configuration features, but still needs some manual configuration.
- Good adaptability guarantees that it is possible to make a system able to adapt to environments without manual intervention. These environments cannot be too heterogeneous and complex.
- Excellent adaptability is the grade of the highest adaptability level. It does not require manual configuration and can adapt to any environment.

Reliability

A certain level of reliability is necessary for maintaining the semantics of data. This level is dependent on the methods utilised by applications. It might be possible to provide the necessary level of reliability for proper operation, with lower consistency guarantees, by limiting the freedom of applications. Therefore, reliability has a lesser impact on the value for an experimental system, as part of the responsibility can be pushed into the application territory. Nevertheless, semantics is important, and deserves a weight value of 30%.

- Poor reliability means that there are no consistency guarantees during data access. It is possible to perform updates on any data item that is reachable, be it local or one in the network.
- Low reliability guarantees some consistency during data access. Most, but not all of the data items are available for a user according to the replica model.
- Average reliability states that most of the data items that are available for a user are in a consistent state.
- Good reliability states that on some occasions, it is possible for a user to access inconsistent information. This is not a big step from average reliability, as in general, inconsistency is not allowed in a data sharing system.
- Excellent reliability is a guarantee that all of the available data items are consistent with the newest version. This is the level of reliability that is used in database systems.

In normal situations, only excellent reliability is adequate. But it might be possible to provide data sharing for users without any strict consistency requirements during data

access. Strict consistency is probably not possible over weak connections, as was implied in Chapter 2.

Overall

With the three described factors, it is possible to calculate a weighted average to get an overall grade. These overall grades are calculated for every replication schema that is put under investigation. The replication model with the highest overall grade is chosen for the implementation of the Spontaneous Data Storage. It is also possible to make a crossbreed between different alternatives, if not a single one of the investigated approaches is adequate.

A comparison between approaches and architecture to support them is detailed in Chapter 5. As the overall grade is a weighted average, it is not a natural number. Therefore, it has a total of nine grades from one to five in half steps to visualise grading. The following is the scale for overall values, and subgrades are in italic: Poor, *weak*, low, *mediocre*, average, *decent*, good, *great*, excellent. The approach that gets the highest ranking is naturally the obvious choice for the implementation.

4. Data transfer in mobile networks

Unlimited mobility of portable devices introduces problems for current networked environments. It is clear that the same methods utilised in static networks with strong connections are not designed with dynamic situations in mind. Therefore, we predict that environments with an ever-evolving network topology and weak connections have a need for technology that is created while considering relevant problem fields of mobile environments. The following bullets present the problems for communication that have manifested themselves during the design of the Spontaneous Data Storage. They are discussed in this chapter and provide the key features for the distribution platform, as the communication is fundamental part of any distribution.

- Routing and packet forwarding. As networks are dynamic, there is a need for routing procedures able to adapt to changes in the environment. A routing method that alternates between efficient technologies when moving from one platform to another and provides general connectivity between diverse devices is a goal worth striving for.
- Data delivery approaches. It is possible to choose an efficient method for data dissemination if a proper selection of data delivery mechanisms is found. Different infrastructures are more suitable for one transfer method than another, and it is necessary to consider both fixed and wireless data transfers in several dimensions.
- Heterogeneous computing platforms and environments. Computing and data transfer resources are very variable between different handheld and wired devices. It is important to notice that not every device is equal, and they do not have a possibility to take equal responsibility of the data transfer and control.

4.1 Data delivery for demand and technology

The traditional approach to data delivery mechanisms has generally been related to pull technology. It is still the most important transfer method and works well with symmetric loads and networks, where data access is arbitrary. Push technologies have gained a steady foothold in the data dissemination field [7][8] after the industry-wide collapse [9]. Articles like "Networks Strained by Push" [10] and "Web push technology is exploding – even though there's no such thing" [11] were very describing of the overhyped technology in the late nineties.

In fact, push technology can be very taxing if the usage target is not deliberately chosen, and should only be used in situations where it really gives some benefit. The easiest way to reduce network performance is to transfer futile information back and forth, if the majority of receiving hosts does not need data. This kind of approach would be

reasonable in asymmetric networks [12], where downstream transfer resources are much higher than less used upstream transfers. A good example would be digital television broadcasts with low bandwidth back channel from clients.

Although the push technology is very promising for wireless networks and is also usable on some occasions in fixed networks, the pull technology is equally as important. It is worth noting that discussion concerning data delivery mechanisms cannot be strictly divided into push and pull technologies, as they form only one dimension of the data transfer. And furthermore, the term 'push' is often used in an incorrect context [9]. Therefore, the following sections define the necessary terms for further discussion and describe data delivery mechanisms.

4.1.1 Data delivery dimensions

This section describes the major characteristics that are worth description. There are also many other factors, some of which are dealt with later. As described with following dimensions, data delivery is an interesting field and it allows unique approaches to the creation of a system. It is necessary to put together the needs of the target system in order to place the correct emphasis on different dimensions.

D1. Pull and push technologies

Pull technology is a traditional concept. Users request information that they need directly from the information provider. This requires that users know a method of contacting the information provider, their location and a specific point in the time. Depending on the needs of user and the characteristics of the application, this can lead to spending an unreasonable amount of time and network resources when polling sites for updated information, and searching for relevant sites [9]. The pull method means that every piece of data has to be specifically requested: otherwise it is unavailable. In addition, pull technology is efficient when not addressing broadcast situations, where several users access the same set of information.

Push technology prefers a more passive approach from the client’s perspective. Users do not request any information; they just sit idly and wait for updates and interesting information. This can introduce latency for information retrieval, but it also relieves users from the above-mentioned burdens of the pull technology. Information control moves from users to data providers, which potentially leads to the receiving of irrelevant information. This is possible due to poorly predictable data access patterns of users and even abusing the system through spamming [9].

Push technology leads to an inefficient use of network resources for the Spontaneous Data Storage if the network layer does not have efficient multicast capabilities. If multicasting is efficiently implemented, as in wireless environments where broadcast ability is an inherent feature, it decreases the amount of redundant messages, and therefore, increases the network efficiency. Push technology introduces edges and flaws, depending on data access patterns, network characteristics and the level of incorrect use of the system.

D2. Periodic and aperiodic transfers

The second dimension after push and pull methods is *periodicity of transfers*. Aperiodic transactions are triggered by data requests or data transmissions, by pull or push mechanisms respectively. This means that the aperiodic fashion is event-driven and time independent, and the delay for a transaction to be performed is related to network performance. Periodic delivery takes the opposite approach where data transfers are not triggered by events, but are engaged in schedules determined at an earlier stage. It is possible for this schedule to be fixed or to include some degree of randomness, depending on the network infrastructure and application requirements. As this study concentrates on creating a data transfer method for mobile environments, it is important for both ends to know when it is possible for transfer to take place, and it is important to use available power-saving possibilities.

D3. Multicast and unicast

The third fundamental characteristic is the factor of reachable hosts with one data transfer. Unicast is communication between a single sender and a single receiver over a network, and is closely related to point-to-point protocols. Anycast is a less commonly used term; it means communication between a sender and the nearest receiver belonging to a specified group. It can be used, for example, to generate and update routing tables for data delivery [13].

Multicast is the sending of messages from one source to several destination hosts subscribed to the delivery list. In theory, a two-way communication medium in addition to knowledge of the recipients establishes the possibility to send messages that eventually reach their destination, though the feasibility of this depends on network infrastructure. It is improper to assume that in highly dynamic and temporary networks every message will eventually reach its destination.

Unicast delivers a similar level of robustness compared to multicast while broadcast introduces an exemption, since it sends transmission over a medium without specifying the recipients. Broadcast introduces several problems when striving for reliability [14]. Therefore, it can be interpreted that it is not possible to create a reliable protocol with broadcast data delivery for mobile environments, though broadcast is an inherent ability of some networks and is a viable foundation for multicast mechanisms.

4.1.2 Data delivery mechanisms

This section presents a classification of data delivery mechanisms based on described characteristics and provides some examples for their usage targets. As a side note, in this study the term 'multicast' is used to refer to any message that is sent to multiple recipients. This includes broadcast messages, as they are a special case of multicasts where, instead of sending messages to some subset of clients, messages are propagated to every client.

Technologies used for different solutions, including some examples, are presented in Table 5. The table focuses mainly on the first and second dimensions of communication as defined in Chapter 4.1.1, and the third dimension “multicast and unicast” is discussed later.

Table 5. Examples of communication solutions.

Technology	**Explanation and examples**
Aperiodic pull	Aperiodic pull is a traditional event-driven mechanism, usually used in collaboration with unicast protocols. It is used for sending a request and waiting for a response-style data delivery as with the downloading of web pages. If a multicast connection is used instead of unicast, it is possible for clients to snoop on the requests of other clients, and receive information not directly requested [15]
Periodic pull	Periodic pull is used to send requests to other sites to check updates and the status of information. As periodic data delivery allows the predictable behaviour of data access patterns, it can be seen that it encourages the use of multicast to send updates for a subset of clients instead of a single client, if the subsystem supports efficient multicasting. Internet-based 'push' systems, such as webcasting, is one example of periodic pull implementations where clients subscribe to a channel and listen for periodic unicast packets [9]

Aperiodic push	Aperiodic push is one of the methods commonly used to disseminate information in a network [16]. It is used on models where the user subscribes to a dissemination service by submitting profiles that describe his/her interests and holds back to receive filtered information [17]. Push protocols are based on multicast-style delivery mechanisms, even if the inherent ability for multicast messages is lacking in the network infrastructure. It is possible to emulate multicast messages by sending several messages over a unicast interface. The aperiodic fashion of push delivery means that it is hard to predict data delivery patterns, and therefore it does not give any hints about network partitions, as is the case with periodic transfers
Periodic push	Periodic push is also popular with data dissemination systems. It has predefined schedules for data sending and is therefore a more predictable way to deliver information. In this study it has become apparent that periodic push is more suitable to fulfilling the needs of reliable data delivery than the aperiodic version. If updated information is not received according to schedules, then the current version of local data might be outdated, a conclusion which cannot be achieved with aperiodic delivery. Broadcast discs and e-mail list digests are good examples of periodic push delivery. Data dissemination over intermittently connected mobile hosts is also easier with periodic multicast messages, because it allows battery savings [18]

Only one method would probably not satisfy the requirements for our data sharing system. For control signals it seems proper to use periodic push, as it reaches as many recipients as possible with high network efficiency. And it also gives some hint about network circumstances. On the other hand, it might not be the best alternative for data transfer and packet forwarding. Technologies that are used in the Spontaneous Data Storage are described in more detail in Chapter 7, and further considerations about data delivery are explained in Table 6. Comparison table for data delivery technologies lists some considerations for the described methods.

Table 6. Comparison table for data delivery technologies.

Technology	**Communication considerations**	**Mobility considerations**
Pull	Efficient for wired networks	Requires routing path
Push	Efficient for wireless networks	Inherent broadcast ability
Aperiodic	Responsive, unpredictable	Does not allow dozing
Periodic	Predetermined time schedule, predictable	Allows dozing

4.1.3 Data delivery considerations

From the perspective of this study it is not reasonable to categorise current protocols strictly as push and pull. Both technologies are integrated to different extent into diverse variety of protocols, and there are other dimensions in data delivery mechanisms to take into account. In heterogeneous networks there are also many different protocols and network layers on top of each other, and it is not reasonable to define end-to-end connections as pull or push. Different network infrastructures support diverging features and potentially, data transfer may include co-operation of different protocols, which takes a totally different approach to data delivery.

The confusion created by push delivery spawns from conflict between actual data delivery mechanisms and observation performed by users. The advantages of push over pull are obvious for some data dissemination applications, and hard to recommend for others. It depends on the environment and application needs and the computing environments, as suggested below.

- If the broadcast ability is the inherent ability of hosts, as it is in the case of wireless connections, it is an efficient way to distribute information regardless of irrelevant information received by the majority of the hosts. On the other hand, a broadcast can be an unreasonable model for networks that lack efficient multicast abilities.
- The predictability of periodic mechanisms can be utilised in many situations. It allows dozing of mobile hosts to reduce power consumption, and gives suggestions about network partitions or other unpredictable events. On the other hand, it is less responsive to events, as information is sent only at certain intervals.

4.2 Communications asymmetry

Most of the devices in a network do not have similar network capabilities and equally strained resources, which introduces asymmetry to connections. Communications asymmetry implies that the flow of information is potentially more fluent in one direction than other. This can be caused by several factors, and is not limited to the network infrastructure [12].

Network asymmetry states that the bandwidth of the communication channel is not the same for upstream and downstream directions. Downstream means data flow from a data source to a user and upstream the opposite. Usually the backbone channels of networks offer the highest overall bandwidth, and only some high-speed local area networks or clusters offer generally higher transfer rates. The connection between two local area networks connected by a backbone introduces network asymmetry, as the

potential data transfer bandwidth is not the same between different parts of the communication link. From the end-to-end perspective this seems symmetric, but the infrastructure under it is not, and information does not travel through a fixed route. This example was provided to bring forward balancing issues of infrastructure: for a clearer picture of asymmetric operation it is possible to consider client-to-server connection from the perspective of a home user. The majority of Internet connections provided for home users are asymmetric in nature. A user has higher potential downstream bandwidth than upstream, as it is in the case with a cable modem, digital subscriber lines and even modern modem protocols. An extreme case of this is a television broadcast where the upstream channel is non existent. Figure 1 presents an example of asymmetric communication. A client requests data through a modem connection, and receives it through a satellite broadcast.

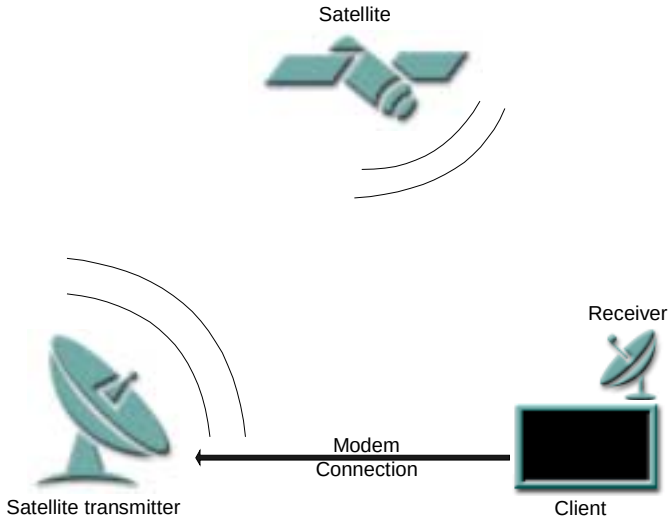


Figure 1. Satellite broadcast with backchannel.

Data volume and direction may stress network resources in a one-sided manner. Applications designed for data retrieval in particular use short request messages to engage the downloading of large data sets [9]. Similarly unbalanced behaviour is common for the majority of applications. This introduces asymmetry to the available bandwidth in networks with symmetric network resources. Also, the frequency of transfers in different directions can cause asymmetric operation, as it is the case with the client-server approach. If a large number of clients are accessing information on a single server, it is possible that the server bandwidth will be saturated, which leads to a degrading of the data transfer performance.

As has been described above, data delivery over heterogeneous networks is a complex issue. Interconnecting diverse devices ranging from handheld wireless hosts to fixed network servers without overflooding connections is a difficult task. One of the factors

that have to be taken into account in this task is the asymmetric nature of connections, especially when dealing with very diverse technologies. As can be seen from the given examples, it is important to take into account the major factors in communications asymmetry to provide efficient data delivery mechanisms for spontaneous environments. The only factor is not the network asymmetry, but the application perspective also has to be considered.

4.3 Routing and packet forwarding

Routing in wired networks is not a new research issue. There exist many decent techniques, and general guidelines on how to make a good implementation. Mobile routing, on the other hand, is a more loosely converged field, which reflects a wide spectrum of different implementations. This research is not exclusively for mobile networks, and a need to study different routing approaches is apparent. The Spontaneous Data Storage requires a decent routing and packed forwarding implementation for every host, fixed or mobile. And therefore, it is necessary to see potential problems and possibilities that are generally provided by ad-hoc routing technologies.

In network environments with a dynamic topology, it is not reasonable to assume that every host in the network would have valid mappings of routing paths, and neither is it practical to strive for it. This leads to the conclusion that multicast routing is the proper selection for mobile distributed systems. The following sections discuss issues in multicast routing. Routing is closely related to addressing mechanisms, and that viewpoint is also included.

4.3.1 Why multicast?

Many transmission methods bring an important feature to assist the maintenance of the connections and topology of a network in a form of multicast and broadcast messages, especially with mobile devices. In wireless environments, transmission signals are released into the same physical medium in a form of electromagnetic waves, in a way that every device within the broadcast receives the same transmissions. The multicasting capabilities of transmission technologies are not limited only to the wireless methods, and the wired technologies such as Ethernet provide similar features for local networks. IPv6 offers multicast abilities for wider variety of devices but is not generally available yet.

This inherent broadcast ability enables the possibility of multicast routing and packet forwarding, which is an elementary part of the designed data sharing approach. Mobile devices having high mobility and intermittent connections attend to the majority of spontaneous operations. These lead to the displacement of hosts in the network topology and spontaneously established connections respectively. Multicasting is an outstanding method to establish connections between unknown hosts and to collect information concerning the network structure and topology.

4.3.2 Global addressing

Addressing mechanisms are a fundamental part of data transfers. It is clear that without accurate information concerning source and destination it is not possible to provide reliable and efficient data transfers. If the addressing scheme is used on a global scale, it has to be universal. The absolute identification of every user has to be guaranteed. This promotes the ability to be able to transfer data between any hosts in the network, including handheld devices and toasters. This kind of addressing scheme with wide enough addressing space is not in general use. To encourage spontaneous operation in dynamically evolving networks, it is necessary for every host to be able to generate a unique location identifier for themselves. Furthermore, the addressing scheme could imply something about routing possibilities to the destination, but it should not take an active role in it. After all, the address scheme is the foundation of routing systems and should support them.

Currently there is no industrial address scheme that fulfils the requirements defined above, and therefore, it is necessary to introduce one. In networks without a fixed infrastructure, it is hard to implement routing schemes based on address information. It would be possible with help of geographical information from GPS, for example, as is the case with some high-end mobile phones. (This is not to say that it is reasonable to assume that low-end devices have any capability of finding out their physical locations.) Because network topology is highly dynamic, it is not necessary to include the routing information in the address. To provide the ability for every host to generate a unique location identifier the easiest route is taken in this study to fulfil the specifications. A long random string will serve as an address. It is not the most efficient, but it provides the necessary functionality. IPv6 might be the real alternative in the future, after it has been globally accepted and is integrated into every low-end device.

4.3.3 Multicast routing

Research has been carried out into the delivery of reliable messages in a mobile environment [19], but it has mostly focused on technologies based on unicast. On wired backbone networks, multicasting is build on the top of the unicast routing infrastructure, and it appears to be a common method of providing multicasting abilities for ad-hoc and mobile hosts in similar concepts. This kind of approach conceals the mobility and the resource limitations of mobile devices, and demands equal treatment for every host. As static and mobile hosts are different in nature, it is not optimal to establish similar connections over heterogeneous data transfer mediums. In this study, the flexibility to be able to shift protocol execution according to the connection types of the host from unicast to multicast algorithms is promoted, when dealing with mobile and fixed nodes.

Multicast communication is an efficient mean to support group-oriented applications, regardless of the network environment, and this is especially true for mobile hosts with an innate broadcasting ability [20]. The following factors of multicast routing and packet forwarding in hybrid mobile environments are emphasised:

- Efficient routing. Conventional multicast routing schemes try to maintain an up-to-date picture of network topology and use every mobile host as a router. A compromise between routing robustness and performance is necessary to provide reasonable hardware requirements.
- Active adaptability. Accurate information of the network topology is hard to maintain in highly dynamic environments with limited transmission resources, and therefore, a solution for mapping the network topology in run time has to be specified.
- Integrated multicast. There is a need for diverging multicast solutions for efficient wired and wireless environments. Multicast over unicast is inefficient between mobile hosts and these technologies should be implemented separately from each other.
- Unlimited mobility. Some of the existing multicast solutions restrict direction, speed and number of simultaneously moving hosts while others prefer discrete mobility where periods of movement are followed by periods of rest. The preferable choice is unlimited mobility with fully spontaneous and self-adaptive configuration and routing, independent of network infrastructure.
- General addressing policy. Mobile hosts are free to migrate between different platforms and infrastructures from spontaneous ad hoc networks to direct connections to fixed networks, which leads to the need for general addressing and routing policies.

As the above statements are somewhat general, further considerations are made to support the implementation of simulation. Reliability of information in the network has to be partly sacrificed to attain reasonable network performance and the utilisation of the limited and variable resources of mobile hosts. Every host needs a globally unique identifier, as described earlier in Section 4.3.2, to have absolute recognition among other hosts and the ability to transparently connect different infrastructures. In this study, it is assumed that it is possible to create these connections by defining general multicast guidelines for searching the environment and listening for contact situations.

This study shows that an important factor in the tracking of route paths in a dynamically evolving network is to minimise state information of hosts. State information that is outdated is useless, and efforts to keep absolutely correct mappings of network topologies are clearly out of the question. On the other hand, it is unreasonable to dismiss all the state information when dealing with fixed points and wired networks that are quite static, although multicasting is expensive on many wired infrastructures.

4.3.4 Summary of multicast

The subject of this research is data sharing, and communication technologies have only been studied to provide adequate insight into data transfer and connection possibilities of mobile and fixed networks to prevent false assumptions of available capabilities. The approach used in simulation is supposed to satisfy routing and adaptability requirements for the data sharing system, and is a simplification created according to considerations that have been brought forward earlier in this chapter.

The first key point of this approach is to provide an absolute identification of different hosts independent of platforms and networks. The second major goal is to make efficient use of multicast without wasting too much resources on wired networks without inherent multicast ability. This is done by using unicast over reliable connections and multicast in dynamic environments. As wired nodes are usually reliable, they are able to use unicast transfers, while mobile devices use multicast packet forwarding. In general, unicast is used if routing paths are stable, even in mobile networks, if nodes are not moving or dozing. This approach minimises performance impact on transfer hardware based on unicast, while it provides decent performance in mobile environments.

5. Replication

Replication is the distribution and maintenance of on-line copies of information. It has a major role in efficient distributed systems, providing high availability, enhanced performance, fault tolerance and the effective use of distributed computing resources [21, pp. 316–318]. Replication is used widely in wired networks with strong connections, while it still seems to be an unworn path for wireless networks. Some replication approaches [22][23] are designed for weak connections, but they are based on centralised models. The goal of this study is to provide a decent level of data sharing services for mobile hosts with weak connections, and during the study it has become apparent that a decentralised model is the preferable choice.

Communication technology is one of the key factors that contribute to the field of nomadic computing, and it has been discussed in Chapter 4. Data transfer in mobile networks. This study combines the limitations of mobile technology with views of traditional replication principles, as is illustrated in Figure 2.

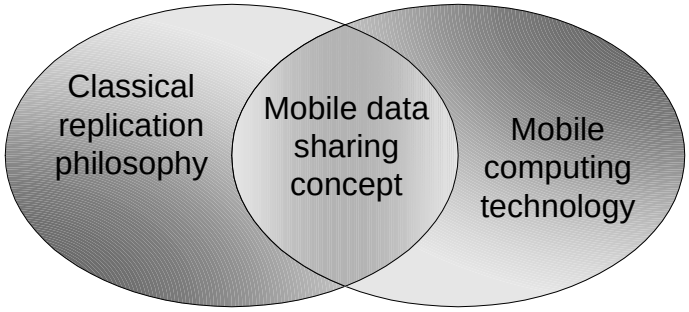


Figure 2. Uniting technologies.

This union introduces new challenges into the replication field. The classical principles created for traditional databases are inadequate for functioning in mobile environments, and therefore, the following considerations are raised by this study. These considerations are discussed further later in this chapter.

- Performance is responsiveness and throughput of data access. It is especially important in devices with heterogeneous and limited resources. The caching of data on the client side is a common way of improving performance, while it introduces consistency problems. Replication for several servers improves response times, especially in environments where connections are highly heterogeneous. It is preferable to distribute replicas for several servers to provide an optimal route for clients to achieve information.
- High availability is difficult to establish in intermittently connected networks without sacrificing reliability of information. Disconnected operation needs local

caching of data to make information available in an off-line state. An increasing amount of replicas distributed on different servers improves the possibility to access information during network partitions, while decreasing dependency on failures of independent servers. The described approach improves greatly the availability in highly intermittent environments.

- There is no absolute guarantee of availability with intermittently connected hosts, and therefore it can be hard to meet real-time requirements specified by users. Nevertheless, efforts to enable fault tolerant operation should not be dismissed. It can be improved, but possibly not easily perfected in heterogeneous mobile environments, where every host operates according to its own needs.
- It is trivial to maintain consistency among replicated files if only read transactions are allowed. When update requests on information are performed, every replica has to be updated or consistency of information is lost. Inconsistency of physical data units is not generally accepted, as it can lead to improper operation of applications dealing with the same logical data unit. On the other hand, strong consistency limits the possibilities for mobile data sharing.

The following sections are designed to establish an overview of replication in mobile systems. Replica management and different consistency schemas are discussed in general, and considerations of fundamental ideas used in the designed replica model are made. Furthermore, the general architecture requirements for replication that are relevant to this research are presented.

5.1 Replica management

Without correct control of data flows, the information in the replica system would degenerate into chaos. It is critical to create a decent model of replica management, which satisfies application requirements and is viable in a target environment. Several data management schemas exist for different usage targets that fulfil their quality requirements, even though they use totally different approaches. Choosing an approach for implementation is a challenging trade-off between different possibilities. The following sections forge the replication model from different perspectives to engender a viable foundation for mobile data sharing systems.

Two extreme models for replica management exist: synchronous and asynchronous [21, pp. 312–316]. In a simple asynchronous model the local replica manager processes all client requests, including updates. Updates are propagated to other replica managers and are eventually completed. This model allows the replica manager to respond to a client transaction as soon as it has been performed locally and informs peers of updates whenever convenient. Updates are processed when received. As can be seen, this model

provides high availability while introducing conflicting updates and totally annihilating consistency of data replicas. This model utilises optimistic consistency schema.

The synchronous model has a totally contrary approach. Not before every replica manager, concerned by a transaction, has updated the requested information; the replica manager processing the client update request is allowed to pass control back to the client. In this model, requests are processed at all replicas in the same order and strong consistency is guaranteed, with the expense on availability. This model utilises conservative consistency schema. The early impression is that neither of these approaches is very viable for dynamic environments on its own, and therefore, a trade-off between availability and consistency has to be agreed on and a model for tackling conflicting updates needs to be specified. Alternatively, some new technology has to be used in co-operation with some existing consistency model.

5.2 Consistency schemas

Although numerous data sharing systems have been established in recent years, the most are suited only for a relatively low amount of servers, which are tightly interconnected. These systems implement a conservative consistency scheme. This scheme guarantees that if one copy of information is updated, then every copy is updated to preserve consistency. It has been claimed that the enforcement of strong consistency imposes unbearable overheads on mobile computing [24][25]. In the following sections, different fundamental approaches for establishing consistency are analysed.

5.2.1 Instantaneous data consistency

Instantaneous data consistency schemas implies that after completing an update request, the available information in the network is instantly in a consistent state. Conservative consistency schemes – such as weak and strong consistency – follow the idea of instantaneous consistency.

Strong consistency has been used as a general reference for consistency of data replication [26]. It seems that this method is only temporarily possible in intermittently connected networks, as it strives towards one-copy serialisability, which ensures that programs access the latest data, and all data copies are synchronically updated. *Weak consistency* gives more breathing room, as it allows read access to any copy of the data, but restricts write access to the latest copy of the data. As is the case with strong consistency, the latest copy of the data is not always available, and therefore they both suffer from similar limitations in dynamic network partitions.

These replication approaches are viable for traditional databases. The same is not the case with the Spontaneous Data Storage, as those replication schemes force a denial of service for update requests to certain copies of data in network partitions, and therefore lowers the level of availability and fault tolerance. Intermittent and weak connections between server hosts are natural attributes of mobile environments, and conservative replication schema considers them as a faulty operation [27]. This contradictory situation implies that instantaneous consistency schemes are not adequate for nomadic computing, and less conservative concepts to utilise data replication are needed. The final decision of the most suitable replication approach is based on numerical evaluation according to Chapter 3.2. Evaluation of replication.

5.2.2 Eventual data consistency

Consistency itself is not a goal to struggle for; it is only a tool to maintain the semantics of data. Several types of data do not need the serialised nature of transactions for data access, and therefore, they are suitable for asynchronous replication schemes [28]. Eventual data consistency is one of these asynchronous and optimistic replication schemes. Furthermore, in some situations it is considered acceptable to use external user intervention to restore the semantics of data for other data types, or even minimal inconsistency may be allowed [4]. Disconnected operation is one of these situations. This study strives to achieve high data availability for mobile devices, and it is not reasonable to deny data access even if arbitrary communication failures are faced. This leads to the conclusion that conservative replication schemes are not suitable for the effective use of systems with weak connections.

The *Eventual data consistency* scheme implements the *optimistic* method for data distributing, as it grants access to any data replica at any time. This introduces a problem of achieving a decent level of consistency [29]. Updates are performed on a locally stored data item and integrated with other replicas as data propagates to other hosts. In this scheme, every update transaction temporarily violates consistency. It takes a while for the update to reach every replica manager holding a copy of accessed data item, and even longer to reach coherency of the local cache. For the time being, when data is inconsistent it is possible for clients to access an old version of the data. If updates are made on the old version of the data, conflicting updates will occur. These conflicts are resolved when detected by replica managers automatically if possible, or with user intervention as needed. After data updates have reached every replica manager and are properly integrated, this approach ensures that data eventually reaches an identical state in every replica, if new updates have not been requested in the system [30].

The eventual consistency scheme solves the problem of availability for weakly connected networks, as it provides the best possible availability. On the other hand, this approach creates a new problem not existing in instantaneous consistency models: eventual consistency does not guarantee consistency during data access. While this might not be such a serious issue with updates on heavily shared files [31], as multiple users do not frequently update them, it is an issue with read requests. Data eventually reaches a consistent state, but meanwhile provides access to arbitrary old versions of data items, which means that it is possible for the client to access an older version of data that it has previously done. Read requests do not generate update conflicts, and therefore eventual consistency does not recognise them. While the ignoring of read-dependencies is a more common problem, eventual consistency also ignores write-dependencies [31]. It is possible for a single user to perform updates on several different versions of the data, and perform some updates even without inviting read requests to modified shared data. Furthermore, it is possible for users not to notice these problems at all.

5.2.3 View consistency

Eventual consistency does not provide any tool to offer consistency upon data access, and therefore, it is not generally suitable for a wide area of implementations. View consistency [32] makes a trade-off from availability to consistency. It is a minor one, but it could be enough to make an optimistic consistency schema a viable alternative. It provides consistency from the user perspective, as it prevents data access to data items that are older than the ones the user has accessed beforehand. This guarantees that the user does not modify arbitrary versions of a data item. If the user has not accessed the requested data before, it can use any version of the data. Otherwise, it has to access the previous one, or any newer version of the data. This limits available information, but decreases the amount of conflicting updates. View consistency was introduced separately from other optimistic schemas because it is used in one of the replication schemas to replace an original consistency schema.

5.2.4 Summary of consistency approaches

As weak consistency is a step toward higher availability from strong consistency, so is view consistency a step toward higher consistency from eventual consistency. Both these approaches have been developed to offer a better trade-off between availability and consistency. A similar problem is faced in the design of the Spontaneous Data Storage, and unfortunately trade-offs have to be made to satisfy requirements. It is clear that instantaneous consistency is not possible over weak connections, and it is preferable to use an optimistic approach with increased consistency.

Increasing availability of information is not only increasing the amount of stored copies. It might be a tempting method as it is a simple one, but it is not efficient. Equally important, or perhaps even more, is where the information is stored. Too aggressive model to distribute data items can be a real resource hog. It can require a lot of storing resources and increase the amount of control signals that are needed to maintain consistency over a practical level. For consistency, it would be good to minimise the amount of replicas that are floating around, and to store them on nodes with high stability and a wide variety of communication possibilities. Good balancing of distribution is one of the most important factors in mobile data sharing, and it seems that it is necessary to develop some new technologies to create a viable implementation into the target environment.

5.3 Architectural model of data replication

This section provides a basic architectural model generally used for the management of replicated data, and describes necessary components by their roles [21, pp. 316–318]. Furthermore, different existing architectures are briefly detailed and discussed to provide insight into their shortcomings and benefits in mobile environments. And finally, the replication model used in the Spontaneous Data Storage is presented. Table 7 provides general descriptions for components in the basic architectural model.

Table 7. Components of the basic architecture model.

Component	**Description**
Replica manager	Replica manager is a more general term for replica server, in not defining any implementation model. It is a process containing the replicas and performing operations upon them directly. In typical centralised systems each replica manager stores and maintains a physical copy of every logical data item
Client	Clients are users of the distributed system. They invoke a series of requests to perform read and update transactions on logical data items stored in the system. Requests that do not include any updates are called read-only requests, and those that modify at least one data item are called update requests. Update requests can also include read-only requests
Front-end	Front-end processes each request generated by a client and work as a bridge between a client and replication managers. It communicates with one or several replica managers on behalf of the client to fulfil requested transactions. This is done by sending update and read requests to replica managers, collecting replies and providing message-passing abilities. It is the component to provide replication transparency. Furthermore, it can also have the ability to collate results from several replica managers and promote different attainable results for the client

Figure 3 depicts the basic architecture model. The double-pointed arrows represent the data access transactions between clients and front-ends, and between front-ends and replication managers. These transactions can be read or update requests. The shaded box containing replica managers is the environment where replica managers co-operate. The basic architecture does not provide information that considers relationships between replica managers, and provides only a conceptual view. Clients communicate with front-ends, front-ends communicate with clients and replica managers, and replica managers communicate with other replica managers and front-ends. The architecture picture shows the relationships between these three entities, which have been described in Table 7. Components of the basic architecture model.

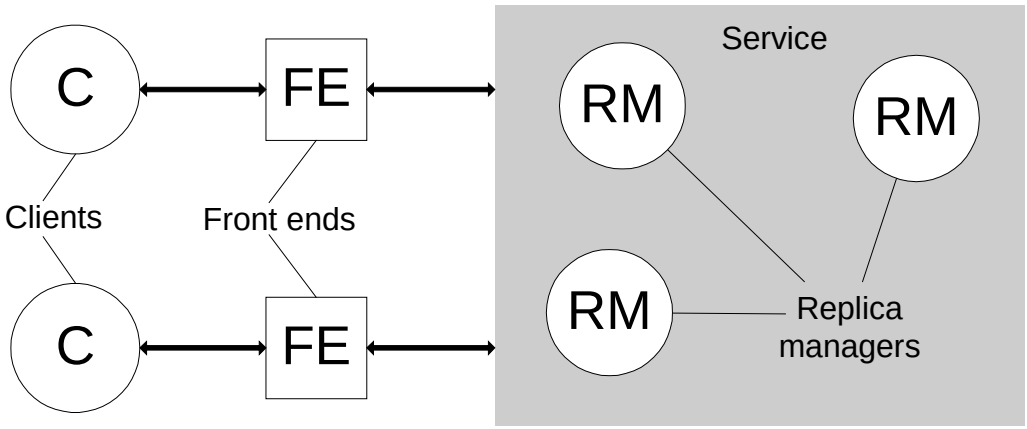


Figure 3. Basic architectural model.

This study shows that the strict pursuing of classical ideology behind this model is not optimal for hosts with heterogeneous physical resources and network environments. Data should be distributed where it is needed, if it is possible, according to resources. It is not possible for a mobile host to contain a physical copy of every item in the database, and neither is it possible to offer connections into the database in every situation. An envisioned architecture model is designed for classical databases with strong connections and uniform servers. Anyway, this architecture presents a high level representation that is viable for our data-sharing concept.

5.4 Replica control protocols

Data sharing is an important feature for bringing availability and flexibility to a wide area of different applications. Various models for data replication have been created to fulfil requirements of diverse computing needs. A peek into the most common models is taken with a discussion concerning their advantages and deficiencies from the perspective of mobile computing.

The most basic protocols, such as RAWA and ROWA are simple and inefficient. These are hardly used anywhere, and are not relevant for consideration. Instead, some common conservative and optimistic protocols such as primary copy, available copy and voting mechanisms are introduced. Finally, a summary of availability versus consistency is presented from the perspective of mobile replication.

5.4.1 Primary copy model

The primary copy model [33] has been created to increase availability of shared data. The model consists of several replica managers where one is a master and the others are slaves. Every front-end maintains a list of nodes with which it can communicate, and needs to access a primary server to perform updates on data items. In many situations it is more efficient to read the item from a slave as those copies are consistent with the primary copy. Slaves bring better availability and performance for heavily loaded systems. When a data item is updated, the primary server propagates update transactions to slaves. The primary copy method denies write access if the primary copy is not available. It is not available when the connection is lost, and weak connections are common in mobile environments.

The primary copy model provides simple locking methods to guarantee strong consistency requirements. As detailed in Table 8, this approach does not provide adequate adaptability for dynamic environments. On the other hand, reliability is decent even though the primary copy model does not offer high availability in network partitions. Figure 4 demonstrates the operation of the primary copy method according to the concept of the basic architecture model.

Table 8. Evaluation of primary copy model.

Attribute	**Implementation method**	**Success**
Availability	Read access is available if a connection to any of the slaves exists. Update requests need accommodation of the master replica manager	Poor
Adaptability	If the primary server fails, then one of the slaves has a possibility to claim status of the master	Low
Reliability	The primary replica manager controls updates and data integrity on slaves. A possibility for conflicting updates does not exist	Excellent
Overall	Sensitive to network partitions, and poorly suitable for dynamic situations	Mediocre (2.5)

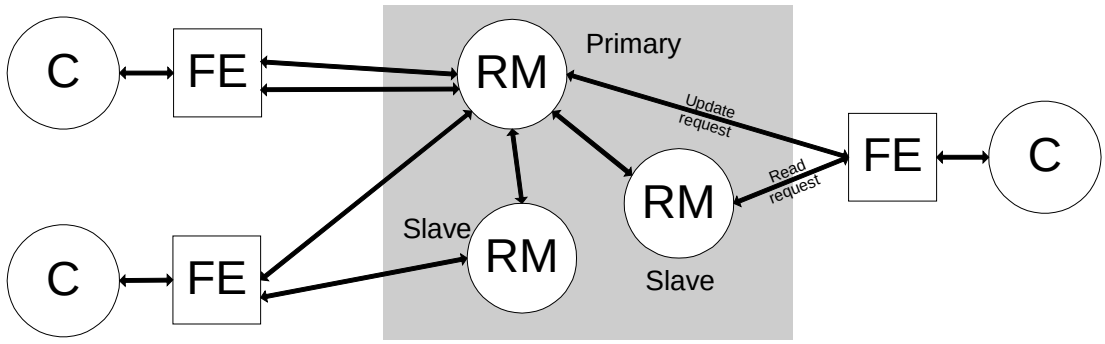


Figure 4. Primary copy model.

5.4.2 Majority voting

There are different voting models, for example, quorum consensus [34] and weighted majority voting [35]. They are quite similar in the nature and provide data access for a client if a defined quorum group or any similar entity grants access. Therefore, it is necessary to survey only one of them, and as majority voting is the simplest approach it is used as an example case.

Voting is a common technique used to prevent groups of nodes from performing a restricted operation in arbitrary network failures. A restricted operation implies a transaction that may lead to the improper function of applications, such as conflicting updates with replicated data. Majority voting can ensure that only a single group of nodes in a distributed system has control over information in periods of network partitions. If the information control were granted for isolated groups in situations of network failures, the data would eventually diverge and lose consistency. With majority voting this inconsistency can be eliminated.

The group bearing the majority of votes is active and has access to the updated database, although the rest of the groups have read access to available information. If the network has been partitioned to several isolated groups, it is possible that none of the groups has enough votes to grant majority access. This means that even this voting model restricts the active role for only one group at time and it cannot guarantee that there is always an active group. The most natural approach to grant votes for hosts is to allow one for everyone, as this would give equal weight. The weight of assigned vote values can have a critical impact on the reliability of distributed systems. When defining set of groups to provide a majority it is necessary to consider the network structure to prevent system from drifting into a halted mode. Halted mode states that no group is active.

Table 9 gives an evaluation for voting algorithms, and shows distinction between primary copy and voting algorithms. The primary advantage is the possibility to access information if the majority of nodes are available instead of one specific node.

Table 9. Evaluation of voting algorithms.

Attribute	**Implementation method**	**Success**
Availability	Read access is provided for available information. Update access is granted, if the user has access to weighted majority of replica managers	Low
Adaptability	Assignments of weights are static, and it is possible that none of the groups gains the majority status	Low
Reliability	Only one group at time can have the majority, hence conflicting updates are not possible	Excellent
Overall	Can tolerate some network partitions, but cannot operate in fully dynamic situations	Average (2.9)

5.4.3 Dynamic voting

All of the previously mentioned quorum-based replica control schemas are static in nature. Quorum sets and vote assignations are fixed and groups are not modified by changes in the network, which can render the data inaccessible due to network failures. Several of the current networks have highly dynamic characteristics, which promotes a need for more dynamic replica control mechanisms. Good examples of this are wireless ad-hoc networks with high mobility and weak connections. Dynamic voting [36] mechanisms use the current state information of the network to determine the best possible quorum set for providing availability and fault tolerance. The consistency of replicas is already guaranteed by all of the described protocols, and the same is the case with dynamic voting.

The primary problem of the static replica mechanisms is the inability to adapt into an evolving environment. Dynamic protocols such as dynamic voting are tailored for failure-prone networks with high probability for network partitions. The level of availability is comparable to static protocols with frequent manual assignation of quorum sets. With dynamic protocols, manual intervention is not needed for optimal functionality and the number of hosts necessary for an update to be possible scales with a number of replicas in existence. For example, in majority dynamic voting the active group is a group of replicas that are able to communicate among themselves with the majority replicas. This implies a need for up-to-date network state information to be able to distinguish the majority group.

As this is the first viable alternative, according to overall grade, it might be worth to consider implementation. Introducing a mechanism able to perceive the current state of the network topology would make possible the resolution of a majority group. The synchronisation cost required by the mechanism could be quite considerable in highly dynamic networks, and the complexity of controlling replication quite high. Table 10 clearly shows that dynamic voting algorithms are superior to primary copy and static voting algorithms, as availability and adaptability are higher, with no negative impact on reliability. The trade-offs to achieve this are increased complexity in algorithm and synchronisation costs.

Table 10. Evaluation of dynamic voting.

Attribute	**Implementation method**	**Success**
Availability	Similar to other quorum-based protocols, but probability for halted mode is lower	Mediocre
Adaptability	Observers change in the network environment, and adjust proper weights and group selections	Mediocre
Reliability	Offers similar level of reliability as other conservative protocols	Excellent
Overall	Trades synchronisation cost for better availability in failure prone and dynamic networks	Decent (3.6)

5.4.4 Available copies method

Previous sections have described some conservative replica control schemas for instantaneous consistency. This section introduces one of the basic optimistic replica schemas, the available copies replica control protocol [37]. Basically, in the available copies scheme, a client can read from any node that currently has a copy of the desired file, while update requests must go to all functioning nodes. When a node falls temporarily from the network, all of its replicas become unavailable. However, as long as a single replica of a file is accessible to a client it remains fully available. There is no guarantee that data requests are performed upon the latest version of the data, and it is clear that plain available copies replica protocols does not guarantee data consistency.

On the other hand, there is a correct way to establish available copies method for *conventional systems*. The read operations must be directed to any replica manager holding the latest version of data, and write operations will be completed if any replica manager accepts the update. To complete an update transaction two steps of validation are needed. Replica managers communicate with each other to verify that every replica that has not received the update is unavailable, and to make sure that all replicas being accessed during transaction are still available.

The available copies method promotes high availability, but according to classical principles, it is not suitable for fault-prone networks, as network partitions can potentially lead to multiple inconsistent versions of the replica. However, it is possible to use available copies with validation to maintain availability for read operations in network failures, and integrating or aborting update modifications when the network partition is repaired. Integration of conflicting updates is not always possible and some of them are aborted. How viable this is depends on the applications.

In this study the classical principles are scrapped to offer availability even in the most hostile environments. With the available copies method, this kind of modification provides the maximum amount of availability: every data item is always available, if it is physically reachable. Table 11 shows comparison details without consistency guarantees.

Table 11. Evaluation of available copies.

Attribute	**Implementation method**	**Success**
Availability	Offers the best possible availability, by restricting neither read nor update requests	Excellent
Adaptability	There are no restrictions for data access, and updates propagate freely. Update conflicts are solved when encountered, and there is nothing to adapt for	Excellent
Reliability	No consistency is enforced during data access, and update integration for solving conflicting updates is necessary	Poor
Overall	Certain types of data do not need serialised order for update transactions, and work acceptably in circumstances where consistency is not guaranteed	Good (3.8)

5.4.5 Available copies with view consistency

With this approach, a step backwards in availability is taken to provide some consistency guarantees. View consistency [32] was detailed in Section 5.2.3. This approach increases complexity and does not provide a consistency level that meets the requirements of conventional replica managing systems. It is not possible to meet classical principles without preventing inconsistent situations.

This method is envisioned in this study as a traditional available copies method with only one modification. The eventual consistency schema is replaced with view consistency. As the weighted overall grade (shown in Table 12) fell below the available

copies method, this is not the best alternative to use. Especially as the complexity level is much higher, almost as high as with dynamic voting. Regardless of this, the comparison table is presented below.

Table 12. Evaluation of view consistency.

Attribute	**Implementation method**	**Success**
Availability	Offers good availability, by not restricting access to newest versions of data	Good
Adaptability	Requires mapping of recently accessed information by the user, but does not need any centralised control for replication, and therefore, it can work in any environment	Excellent
Reliability	Provides a limited amount of consistency, as it does not allow a user to access older data than it has previously accessed	Low
Overall	If users and applications can survive with the limited amount of consistency, this is a good solution. However, It does not give the maximum availability	Decent (3.7)

5.4.6 Gossip architecture

Although gossip is not a widely used architecture, it is a popular method to demonstrate features of data replication [21, pp. 327–334]. This is probably because it has a very flexible design and it uses an extensive collection of replication characteristics. It is tailored for high availability and allows the use of different strengths of update ordering. High availability suits the needs of the Spontaneous Data Storage, but different update orderings and complex architecture does not. Nevertheless, some of the ideas behind gossip architecture are quite interesting, and have given inspiration to design out the Spontaneous Data Storage.

- Updates between replica managers are propagated through gossip messages, which contain the most recent update information they have received.
- The gossip messages are exchanged in a lazy fashion; they may be sent only occasionally, after several updates have been received.
- Architecture supports three different levels of update ordering, which are casual, forced and immediate ordering mechanisms.

As updating is performed through gossip messages that are delivered in lazy fashion, it does not provide strong consistency guarantees. A similar method could be used in the Spontaneous Data Storage with some modification introduced for mobile environments. For example, messages could be sent periodically to allow battery-powered devices to doze most of the time when they are not in use. Different levels of update ordering constitute an interesting approach, but it might not be possible to offer immediate mode ordering over mobile networks. Also, a forced mode might introduce some problems, and including more than causal mode would bring unnecessary complexity. Inspiration has also been drawn from innovative file and music sharing programs that are scattered around the Internet, for example Gnutella [39].

5.4.7 Conclusions of replica control

The evaluation framework introduced in Section 3.2 makes it easy to perform comparisons and draw conclusions. From Table 13 it is easy to see the three top performers, and that each of them are adequate for the Spontaneous Data Storage. Available copies and view consistency represent the forces of the optimistic approach, and dynamic voting is the sole survivor of conservative methods in this tight challenge.

Table 13. Summary of evaluations.

Quality factor	Weights	Primary copy	Majority voting	Dynamic voting	Available copies	View consistency
Availability	40%	Poor (1)	Low (2)	Average (3)	Excellent (5)	Good (4)
Adaptability	30%	Low (2)	Low (2)	Average (3)	Excellent (5)	Excellent (5)
Reliability	30%	Excellent (5)	Excellent (5)	Excellent (5)	Poor (1)	Low (2)
Overall	100%	Mediocre (2.5)	Average (2.9)	Decent (3.6)	Good (3.8)	Decent (3.7)

The problem with the rest of the competitors is that they are unable to deal with network partitions. The conservative schemes permit data update access only in one partition, if at all. This is also true for dynamic voting, but reliability is also an important factor to consider, which is the reason why dynamic voting is able to challenge optimistic methods. Adaptability is also a concern and dynamic voting is the only conservative method for achieving an average grade. Overall, dynamic voting would not perform well in the most hostile environments, but it could be viable in some mobile environments. As was hinted at earlier, there is no good reason to pick view consistency over the available copies method. The considerations for the best alternative, available copies method, are following:

- It provides best possible availability and adaptability.
- It does not provide any consistency guarantees, and therefore it needs some technologies to support semantics of information.
- It is easy to implement, as there are no restrictions for data access. It is easy for communication, as it does not require reliable connections.
- It requires an algorithm and advanced technology, to perform the integration of updates and to resolve conflicting updates, respectively.

Technologies to support the available copies method for the Spontaneous Data Storage are described in the following chapter, and the implementation principles are explained in Chapter 7.

6. Be smart and efficient

This chapter deals with additional technologies and methods that can be utilised to improve the efficient use of resources for data sharing and access. In the previous chapters, a lot of emphasis has been placed on the limitations of mobile technology. The primary problem has been unreliable connections. The biggest concern is not the bandwidth of different transmission mechanisms, but the existence of a connection at all. As the increase in hardware resources does not overcome the principal problem, this study presents an approach that tries to minimise the problem of temporary disconnections.

This study takes the smart data distribution to a new level, as it is not only discussed in detail, it is also implemented; and the functionality of the implementation is analysed. Furthermore, a unique idea of information granulation into small elements is presented later in this chapter.

6.1 Intelligent replica distribution

Spontaneous networks and nomadic computing promote a new approach for fulfilling the requirements for the replication scheme. Many studies have been focusing mainly on availability, performance or fault tolerance [40]. Wireless network performance is nowhere close to physical network transfer rates and reliability is much lower. Therefore, it can be assumed that a replication scheme that would focus on smart distribution and control of replicas would greatly benefit mobile environments, while having lesser impact on fixed high performance networks. Providing quick access to data and availability in network partitions is a trade-off between performance and availability. The first section provides some insight for intelligent replica distribution, and the second describes general ideology of the proposed data sharing model.

6.1.1 Smart distribution in perspective

An important issue in the designing of smart data distribution mechanisms is to determine how many replicas to have and where to place them [40]. Usually it is preferable to access data that is quickly available. Data is commonly quickly available if it is near the user. Therefore, in order to provide good performance it is necessary to store replicas in a neighbourhood where it is needed. This leads to the grouping of replicas around certain user points. Read requests are quickly performed on the closest possible neighbour, and write accesses are easily performed for a group of replicas.

In this study wireless connections imply some sort of geographically located information. This leads us to the conclusion that the grouping of replicas provides good performance for frequent users of replicated items, but low performance for random users further away from the neighbourhood, or even nil availability in network partitions. It is impossible to predict the invoking of data requests from an arbitrary location, and therefore, it is not possible to distribute replicas to every relevant neighbourhood. Different approaches for the grouping of information do exist. Even if semantics or any other grouping of information is used, it is necessary to consider a geographical location, as mentioned earlier. In network partitions, which are common in mobile environments, connections to some parts of the network is lost. In other words, a certain geographical area is out of reach of the network, and therefore, the information stored in the blind spot is not available. When trying to provide good availability for mobile hosts, it is necessary to distribute information into places where it is possibly used, but has not been used before.

A good replication schema has generally been considered to balance performance and availability according to the read-write pattern of the object [40]. Strong connections in fixed networks provide a good skeleton for fault-tolerant operations. This is not the case with mobile computing. As has been proven, it is not reasonable to aim at absolute consistency of information in mobile networks [41]. Allowing some minor inconsistency provides the possibility to distribute replicas more freely. This study tries to make good use of that freedom.

6.1.2 Case-specific considerations

It can be assumed that heterogeneity of different hosts play a major role in the distribution of responsibilities. Some of the hosts have higher resources, stronger connections and a bigger neighbourhood than others, and therefore, they are better suited to claim a greater share of responsibilities over the distributed system. When distributing replicas it is crucial to choose a reliable and focal host for data storing. Hosts that work as a bridge between potential partitions of the network are good locations to consider. This would reduce traffic between clusters of the network, and increase availability and performance in those clusters. If hosts are present only temporarily in the network they are not suitable for increasing availability for other hosts, and therefore, persistency of the host in the network is the most important factor. Absent devices should not control data distribution, they should only use the data. It is possible to cache information for quick local access, while consistency guarantees cannot be provided. Grouping the replicas tightly where the access is needed enhances performance, and distributing the replicas to a few potential locations can enhance the availability.

In the Spontaneous Data Storage, the responsibility of replica management is distributed. Every host bases its actions on saving a data item in or deleting it from the locally available information. Locally available information consists of information about neighbourhood, update messages from other replica managers, locally stored data and configuration. The host itself has the best knowledge about its future actions, and is not concerned about acts of others. Replica management is solely independent on every host. Both push and pull methods are utilised for advertising and requesting information.

6.2 Data update integration

It is common for information that is stored to be updated occasionally. It does not bring about any problems in conventional database systems, as "the fundamental assumption in the classical relational model is that data is consistent and hence no support is provided for dealing with inconsistent data" [42]. The concept of the Spontaneous Data Storage is not classical, and there is no support for consistency at data access. Therefore, it is necessary to have robust support for dealing with inconsistent data. The goal is to have eventual consistency between each and every replica in the data storage. Furthermore, the used technology does not provide reliable connections between nodes, and therefore, it might take a while for every replica to reach consistency. This study presents a unique idea of dividing files into small elements, which can be updated in parts.

6.2.1 Granulation of information into small elements

A general method to update a file is to send a new version of it to replace the old one. This is a good method if transmission resources are not a concern. For devices with low transmission resources this can be a formidable resource hog, and it is preferable to find an alternative method for updates. It is possible to update only the part of the file that has been modified, but how is it possible for the replica manager to update the correct part of the file? This is a considerable problem. If there were several patches coming in from *different clients and in an arbitrary order* it would be difficult, if not absurd, to integrate them. If a file is updated without any trace of a patch, it is quite difficult to determine between different files which of them to use, as they would not be consistent. Without a version history it is not possible to integrate files without the possibility of conflict. This same problem would be present even if entire files were to be send when performing updates. A new approach to solve this problem is presented, at least partly.

By partitioning the file into small elements it would be possible to update some specific area of the file. An update message that can be used to send information from a user to replica managers contains the new information, and the location in the file it will replace. Instead of replacing the part of the file, it is attached to its end. This provides the original file with a revision history, including patches. With this information it would be possible to integrate patches to the file when it is requested for a download to provide the most recent information. When several patches are modifying the same location of the file, a need for conflict resolution arises. The newest revision of updates from every author is included, and for the rest it could be rational to use manual intervention. If a file were bloating as a result of frequent updates, it would be possible to integrate some of the oldest patches into the file to decrease the length of the history. Conflicts in this concept are less frequent than in models that are uploading whole files. It would also be possible to utilise any of the existing technologies to resolve the conflict. This kind of approach would decrease the amount of conflicting updates, and could be optimal for mobile data sharing. It could also be used in systems with high consistency to decrease network traffic, or some applications where several people are working with the same files, such as programming.

6.2.2 Changes in the application perspective

The described approach requires new features in applications and replica managers. It is necessary for applications to be able to partition the file before uploading it into data storage, and it is necessary to compose patch information and location in the file. This introduces complexity into interfaces, but at the same time it decreases network traffic substantially. Current trends of advancement in technology show that computing power is not a problem, but data transmission capacity certainly is. This direction is clearly visible in mobile phones where integration of image compression chips enables the streaming of live movies over next generation mobile phone networks. This would not be possible without efficient image compression. According to this vision, it is possible to minimise network traffic without reducing the availability of data sharing.

The replication manager also needs a support to integrate patches, and to manage revision history. Multicasting of updates is also important in order to achieve consistency in the data storage, as there is no certainty that packets have been received in every location. The introduced method for data partitioning and the managing of different file versions introduce an enormous amount of complexity into the software domain. It could be worthy to make a trade-off between software development and network performance.

7. Spontaneous data storage

The previous chapters describe the problem fields of mobile data sharing, and provide technologies to overcome various obstacles. This chapter describes architecture and simulation models used in implementing and validating the Spontaneous Data Storage. The Spontaneous Data Storage is a concrete result of our data-sharing concept.

The most advanced part in the implementation is the data distribution algorithm, the heart of autonomous data management. As hinted in the earlier chapters, the designed distribution model does not utilise any kind of centralised control. Therefore, every participant in the distribution has to be able to operate autonomously. The following sections provide background information and requirements for data distribution before moving forward into architecture and simulation descriptions.

7.1 Background of spontaneous data storage

As explained in the introductory chapter, the idea for this study originated from experiences in implementing the Distributed Service Platform, which is presented as an example case for QADA architecture in VTT Publications 456: Quality-Driven Architecture Design and Quality Analysis Method [1]. The technical report on the service platform is still in progress, and thus, it is not possible to give a reference to it. The publication provided does not describe the latest version of the Distributed Service Platform and therefore, it is necessary to cover some basic ideologies that were used in the platform.

The distributed service platform consists of two primary parts: user services that provide the ability to access and fetch services, and service providers that offer and advertise services. Before it is possible to access a service, it must be distributed and located. This is where Spontaneous Data Storage steps in, as it tries to fulfil the requirements of a distribution channel for services. Some of the main ideas of the Distributed Service Platform are presented in the following bullets.

- Fully spontaneous operation in a way that every node in the network has the ability to claim responsibilities for itself without any preconfiguration.
- It is designed to work in hostile environments with the use of an inherent ability to reconfigure itself to its surroundings.
- It has the ability to establish its existence whenever circumstances in an environment are suitable.

The data distribution method designed for this study must emphasise these same principles to be able to operate in similar hostile environments. This reflects a requirement of decentralised functionality. According to the technology surveys in earlier chapters, it is necessary to limit the implementation into small-scale spontaneous networks, as current technology does not allow decentralised nature on a global scale. For example, broadcasting is a technology which does not require centralised service providers, but it is an absurd idea to use it on a global scale.

7.2 Requirements for spontaneous data storage

This section introduces the primary requirements and features that are important for Spontaneous Data Storage. Links to theory, or technology to fulfil described requirements are also provided. Currently, the most valid target platform incorporates mostly new handheld devices with storing and computing capabilities, with co-operation from some fixed desktops and servers. Therefore, an emphasis on the perspective of nomadic computing has played a dominant role, as explained in Chapter 2. Challenges for mobile distributed systems. This study provides the necessary considerations of technology limitations and possibilities. According to these considerations, Table 14 lists solutions for various requirements. In the following sections, the knowledge that has been gathered during technology surveys is converged into an inspired implementation.

Table 14. List of implementation requirements.

Requirement	**Technology or theory**
Ability to operate without external control	Decentralised approach (7.1). Periodic control messages with push technology to provide predictability (4.1.2) and information exchange
Possibility for heterogeneous nodes to participate in data usage and provision	Dynamic roles for services providers and users (Sections 6.1.2 and 7.3)
Highly available data access regardless of environment	Available copies replication model (Sections 5.4.4 and 5.4.7) with smart data distribution (6.1)
Reliable information storing	Smart data distribution (Chapter 6) and resource management (7.4.2)
Unlimited mobility	Multicast routing for small-scale spontaneous networks (4.3)
Special requirements of portable handheld computing devices	Periodic control messages to allow dozing (4.1.1). An option to choose established services to support devices with low hardware resources (7.3)

7.3 Principal ideas and partitioning

This is a concept for data distribution over spontaneous and fixed networks without centralised control. The Spontaneous Data Storage subsystem has been divided into different service packages, which have different functionalities and purposes. These packages are listed in Table 15. Service components of Spontaneous Data Storage. They provide a link to architecture used in conventional databases in Chapter 5.3 to clarify the positioning of these services parallel to classical architecture.

Table 15. Service components of Spontaneous Data Storage.

Service	**Description**
Client service	Client service provides users and applications an interface to use, store and update information in data storage. It has a very similar functionality to the *front-end* (see Section 5.3)
Server service	Server service is the information distribution and management unit, and is in some ways similar to a *replica manager* (Section 5.3). It integrates updates and serves data on requests as a conventional replica manager. It has an asymmetric approach, which does not have conventional control signals, and does not require reliability
Communication service	Communication service is the connecting link between every node that participates with the data storage. It provides a messaging interface for server and client services

Figure 5 visualises interaction between the services in Spontaneous Data Storage. The interaction model is based on the ideology behind Figure 3. Basic architectural model. The shaded area depicts a communication environment that is based on a network of communication services. The box with disjointed borders represents an entity of one node in the replication model. The conceptual architecture of a node is explained in Figure 6. The rest of the components are explained in the picture.

Server services float freely in the shaded area, as they do not interact with any components outside the communication environment. The application domain does not belong to Spontaneous Data Storage, and therefore, it does not have any contact with the shaded area. The client services are located in the border of the communication environment. The reason behind this is that they perform their internal messages regarding the SDS through the interface of communication services. On the other hand, they do interact directly with applications and provide an interface to the outside of the SDS.

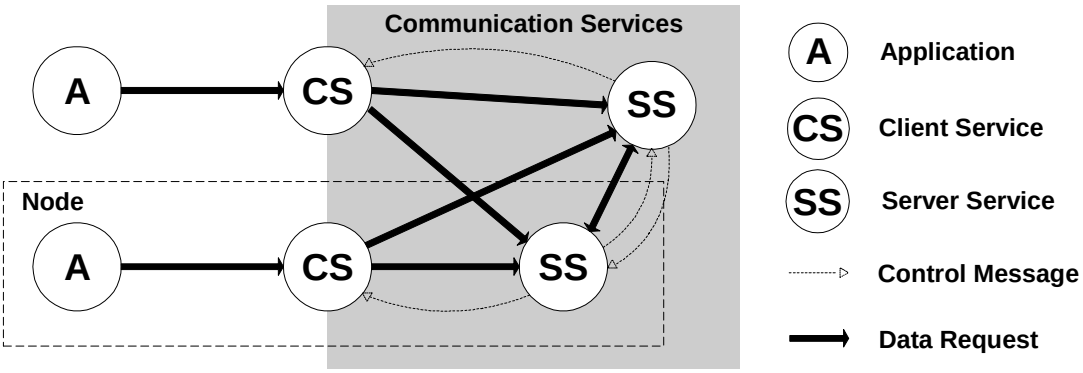


Figure 5. SDS interaction model.

The defined services provide middleware between applications and network solutions, as presented in Figure 6. The conceptual architecture picture is drawn according to the QADA method [1], which has some similarities to UML-notation. In the picture, the node is an entity that contains all the applications and hardware resources of one computing device, as described in Table 16. As shown in the picture, the Spontaneous Data Storage is not designed to replace any of the existing functionality of computing devices, but it provides an alternative middleware layer to utilise a new data-sharing concept.

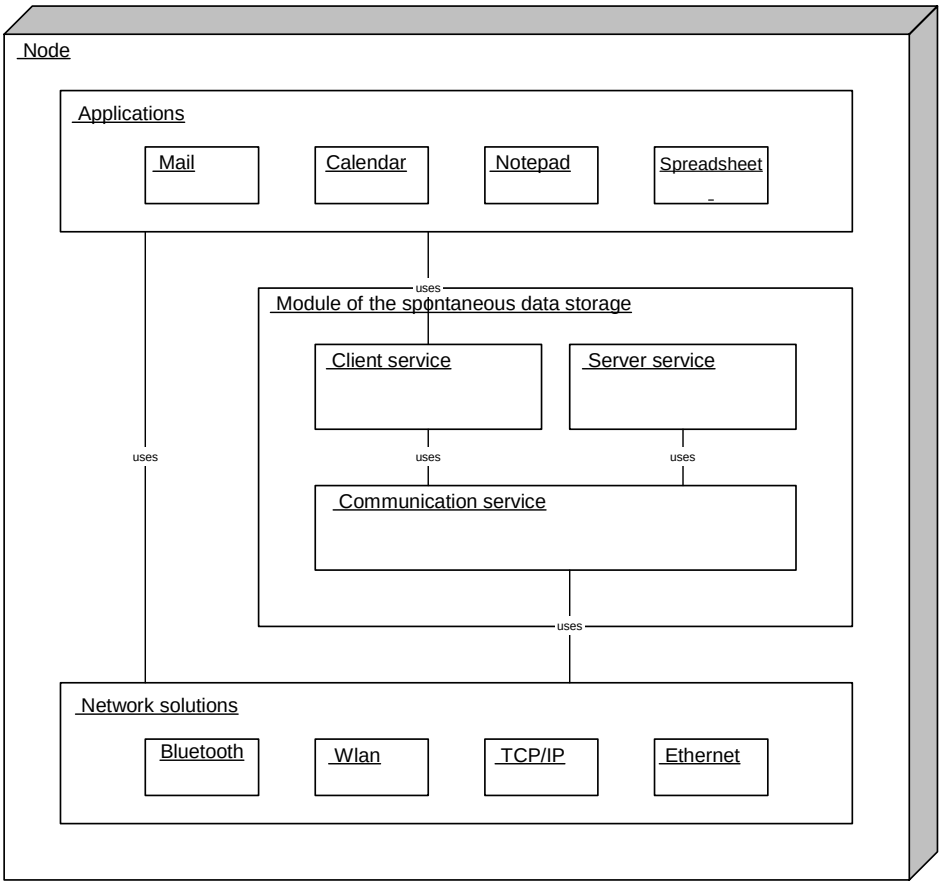


Figure 6. Conceptual view of a node.

The relations between subsystems in the picture are as follows. Applications are able to use data sharing features through a client service. Client and server services communicate with services located on other nodes through a communication service. The communication service utilises different transmission protocols and technologies to deliver messages over different infrastructures.

The server service provides information storing resources and manages distribution in co-operation with other server services. The client service provides an interface to store, search and fetch information. When a client service is storing data items it negotiates with the server services of available storing resources, and decides on a storing location. It is able to generate search queues and fetch located information.

Spontaneous Data Storage presents a revolutionary model for data sharing, and Table 16 on page 60 provides definitions of unique terms used in various descriptions later in this chapter.

Table 16. Definitions for distribution model.

Term	**Definition**
Node	A node is a physical unit that provides a platform for communication, client and server services to function with. It is a hardware and software entity that possesses computing and transmission capabilities
Neighbour node	Neighbour node defines a group of nodes that are directly connected to the node under study. In other words, only one hop away. From the visual point of view, these are all the nodes that are seen before the horizon
Data distribution algorithm	Data distribution algorithm (DDA) is the mechanism to determine storing locations for data items. It utilises several different parameters of the network environment and knowledge of replica locations. It forms the heart of server services and controls the flow, storing and removal of data items
Data distribution pattern	Data distribution pattern (DDP) refers to different location combinations of stored data items. Distribution patterns evolve according to decisions by data distribution algorithms. A good pattern is a combination of replica locations where data access availability is high

The resources of portable devices do not allow the storing of all data items in every node, which is the reason why it is necessary to implement smart data distribution. The idea behind smart data replication is to provide optimal storing locations for data items. A data distribution pattern is used to describe different location combinations of stored

data items, as shown in Figure 7. The circles represent nodes and their locations in the network topology, and the lines represent connections between nodes. The shaded circles point to the nodes currently storing the data item under observation. The network topology is the same in both pictures, and both of the networks store three replicas of the observed data item. The difference between networks is the locations where the data items are stored, which is the difference between the data distribution patterns shown in the pictures.

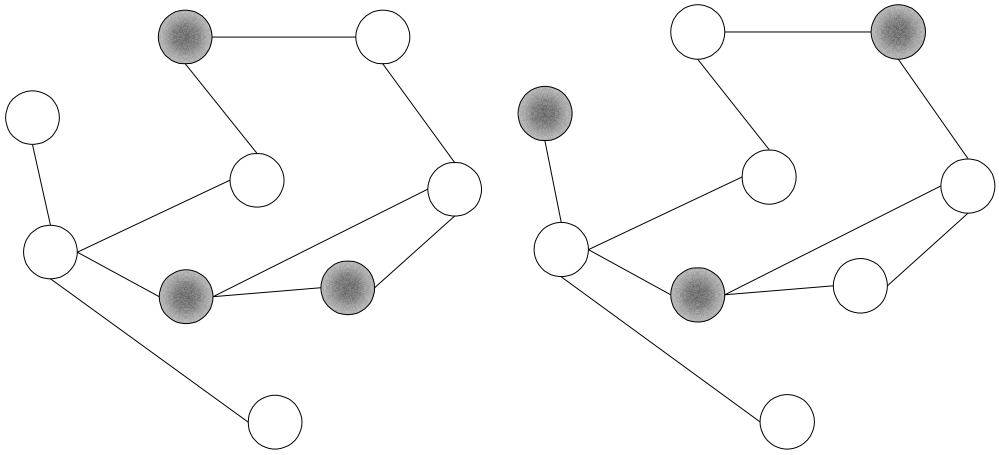


Figure 7. Two different data distribution patterns.

SDS functionality has been divided into three different packages to support heterogeneous computing platforms. As an additional benefit, it provides a clearer presentation of the functionality of data sharing subsystem. The original idea was to allow the possibility of limited data sharing responsibilities, in order to provide a chance for low end computing devices to participate in the data sharing system.

Every participant node must include communication services, while client and server services are fully optional. This allows more complex and computing-heavy functionality to be dropped from lesser nodes, while also limiting data access possibilities. An established Spontaneous Data Storage needs at least one server service in a network to enable data storing abilities, and a client service to provide information access. If more servers are engaged in the network, the burden of data storing is automatically divided between servers according to every server's own resources and interests.

Communication and file transfers between service hosts are based on multicast messages and point-to-point transfers. The address that is used to reach a service consists of two parts. The first part is a globally unique identifier that is unique in every communication service. The second part is a service identifier, which points to a correct service on the node. A globally unique identifier is used to reach a correct node, and a service identifier to reach the correct service on that node.

All of the transactions between different services are based on this addressing system, which provides a platform and protocol-independent environment in which to send and receive messages. The addressing scheme provides the necessary uniformity for a wide range of diverse nodes to communicate with each other, which is not currently available in existing technologies. This approach had to be taken to allow every device to address any other device unambiguously, as it is not reasonable to assume that every device in the target environment – small-scale networks and future virtual home environments – has compatible addressing mechanisms.

7.4 Components of the data storage

This section gives a high level view of the different service packages and their collaboration. Additionally, most of the responsibilities and some implementation features are described. The model for the Spontaneous Data Storage has two layers, as explained below.

Clients and servers are transparently connected through a dynamically evolving cloud of middleware to every other node in the network. This middleware is a form of centralised routing server, which has been distributed on different sites without knowledge of its surroundings more distant than they can see directly. The functionality of this cloud is loosely packed into communication services, and consists mainly of an ability to provide transparent transactions for clients and servers to operate with each other. Servers and clients run on the top of communication services as shown in Figure 6. Servers are run on volunteering hosts to provide physical data storing, distributing and control abilities. A host node provides resources to server services. Client services bring about the possibility to store, fetch and update information on these servers. All of the operation is done transparently, and none of the server and client services need to know anything about the topology or structure of the network. The following sections provide a more in-depth explanation of different service packages.

7.4.1 Client service package

Data storing, fetching and updating is provided by client services. Client service is a service for those, who want to utilise shared information without providing resource sharing on a local host. It is the only way for users to access information in the data storage, as it has been designed to be impossible to acquire data items directly from server services. Client service has several responsibilities in the co-ordination of data storage. The following are the responsibilities of client services.

R1. Processing of upload requests by negotiating with servers

Data storing is performed through client services. For successful storing it is necessary for a user and any of the server services to agree on storing terms. This is transparent to users, and everything is done between the client and server services based on the requirements given by a user. Feedback for negotiation of data storing is provided by some of the servers in the network. It is not necessary to communicate of data storing with every participant in the data storage, as this would cause fruitless overheads for transmission. After the storing location is determined, it is possible to complete the upload request. As with any existing storing system, it is not always possible to complete a storing sequence, especially if enough storage capacity is not remaining. The storing capacity of the Spontaneous Data Storage does not increase linearly with the increased physical storage.

R2. Processing of update and read requests from users

Client services process user update and read requests. In conventional databases, an update request presents updating and storing of a file with a request. In this concept, it is split into two, an upload request and an update message. It is not correct to use the term update request in this context, as there is no request for an update. A client multicasts an update message into the network, and servers deal with integration of updates. There is no locking or asking for permission to modify files prior to the sending of a message, and thus, nothing is requested. This approach was taken because the most recent information is hard to obtain due to frequent network partitionings.

R3. Providing of location and replication transparency

Transparent access is one of the basic requirements of file sharing systems. A user should only see one data item, and in one location. Reasoning behind this is quite clear, as it is only way to provide a common and simple interface for data access. All of the required functionality to enable such operation is to encapsulate functionality to some component, and it is commonly placed in the front-end. The client service is effectively in a similar role in the Spontaneous Data Storage, and therefore, it has to provide these features.

R4. Providing of subscription for information

It is necessary to track information that has been stored into the data storage to get a picture of possibly available information. When a user requests specific information it

can be interpreted that, the user shows some interest in a specific category of information. This automatically adds the user to the subscription list of an information group, and it will continue to receive multicast updates for that group. It is also possible to request a full list of available information without subscripting to every information group. The idea behind this strategy is to lower the amount of futile information traffic. Furthermore, it is possible to achieve some information of data items that currently have some presence in the data storage by listening to traffic that is forwarded and is passing by. This method does not give a full list of stored information, but it does not consume any transmission resources and it is free. There is little point in keeping an up-to-date list of logical data units in a dynamically evolving network, and neither it is reasonable to keep information concerning all the data units on every host.

R5. Mapping of logical data items to physical replicas

If a user specifies a logical data item it wishes to access, the list of available items is not important. The client services store only the absolutely necessary information, which does not include locations of items. A buffer for a recently accessed data item might be used, but is not necessary. The idea is to reduce the use of local resources to a minimum in order to enable portability into low-end devices. When a logical data item is accessed, a client tries to map it to a physical data item. This is performed by sending a multicast query to servers and by waiting for replies to find an optimal server to perform the download. After which the client service deals mainly with message passing, and might change to an alternative download source if the connection to the current one is interrupted.

7.4.2 Server services

The server service package is for those who are willing to contribute part of their bandwidth and storage space for common use. The control, storage and distribution of information are based on the storing capabilities of server hosts and bandwidth resources of nodes in the network. Without several servers it is not possible to guarantee high availability of information during network partitions or in other hostile circumstances, and it would not be possible to meet the requirements of the Spontaneous Data Storage. Therefore, it is recommended or even obligatory, for every participant with a reasonable amount of physical resources, to make contributions to the data storage for common benefit and usability. The fundamental principle of this data storage is to control information without centralised and fixed control, which provides flexibility for distribution and storing of data with weakly connected devices. The responsibilities of the server services are the following:

R1. Controlling of distribution and maintenance of stored data

Smart distribution and maintenance of information is the heart of the Spontaneous Data Storage. Robust availability of information is closely related to redundant copies of information distributed all over the network. As there is no centralised control, it is necessary to develop decent algorithms for the server service to choose optimal locations to store replicas, and to make efficient use of the transmission bandwidth and computing power. Servers are autonomous, and therefore, the algorithm has to decide which of the replicas the local server is interested to store in and serve. More information on smart distribution is discussed in Chapter 6, and algorithm implementation in Section 7.6.3.

R2. Providing of resources for data storing

A user provides resources for the server service by allocating a slice of storage space and computing power from the resources of a local node. These resources are used freely by the server service in the most efficient way. A major part of the physical storage space is filled with data replicas, while some space is required for data management. All of the available space is used to achieve high utilisation of resources. This approach increases the amount of replicas, which increases availability. When a client sends an upload request, the server may clear enough space for the data item, if possible, and compose a reply to a client. Not all the servers respond to the upload reply, as it is enough if one of the servers downloads the data item; it is unnecessary to waste network resources. Servers do not know anything about each other in this situation, and it is up to the client service to determine the optimal location for storing information. From the chosen location the information will eventually drift to other places. A decision to send an upload reply to a client is based upon information of the data item that is integrated into the upload request, and the current state of the server service. It is upon to the server hosts to agree on the storing of information, and they act according to their own interests.

R3. Providing of access to physical data items

Access to data items consist of update messages and download requests. It does not include upload requests, as it does not link to information that is not currently stored in the SDS. The client locates physical copies of information by sending a multicast message, and waiting for replies. Server services send replies if they are holding a replica of the desired information at the moment. After collating replies, the client chooses an optimal server to download and sends a download request. The update

message is multicasted into the network from a client, and servers integrate updates when the message is propagated to them.

R4. Providing of update integration to achieve eventual consistency

Update integration and conflict resolving is one of the larger problem fields of this study. The optimistic replication model used in this study allows read and update accesses to any data unit at arbitrary moment in time. High availability is achieved at the expense on consistency, but methods to minimise the negative effect are introduced in Chapter 6. The subsection on data update integration gives a description of how consistency is improved. This development is one of the major ideas of this study, and an XML-implementation method was visioned, but the time frame did not allow the implementation. Anyway, the goal of update integration is to achieve eventual consistency.

R5. Providing of reliably stored information

One of the more interesting problems that Spontaneous Data Storage introduces is the possibility to lose stored data items. This responsibility is introduced separately from maintenance of data, as it is a new problem that is not present in classical databases. As every server acts autonomously, it is possible to concur decisions in several servers over a short time period to produce devastating results. As information is distributed according to the interest values of data items, it is possible to come into similar reasonings in different locations of the network. If several servers, for example, see data item A as the least interesting one, and all the others who have a replica of the A come to the same conclusion, it is possible that A will be lost. When the server decides to replace the least interesting item with a more interesting one, and when several other nodes drop it, there might be a period where every server believes that information is still stored in several locations, and A is lost. This problem is studied in greater depth when running simulations.

In network partitions it is also possible that some information is rendered inaccessible for some hosts, as storing locations might not be optimal or storing space inadequate. If network partition does not repair, it is possible that information is never again available in the other network partition. From the client perspective this is also equals to the removal of information, even though the information still exists on some hosts. This problem is caused by movement of hosts, and can be attacked by locally storing the most interesting information for a client.

7.4.3 Communication services

The primary role of communication services is to provide message passing between nodes of the Spontaneous Data Storage. It is not a trivial task to send messages in dynamic and heterogeneous environments. The ability to create connections between nodes is necessary, and it is beneficial to observe ongoing changes in the surroundings. Communication service is also the foundation for client and server services, and it provides the ability to establish and shut down services. Features and responsibilities are explained in more detail in this section.

R1. Providing of interface for messaging

The Communication service enables nodes to communicate with each other by providing message sending and receiving abilities. The main structure of each message is identical. The communication service encapsulates additional proprietary routing information into a message and sends it over any available protocol. The network topology and heterogeneity are hidden from the client and server services by providing two message types, unicast and multicast. The message model is discussed in more detail in Section 7.6.4.

R2. Preparing of nodes for connections

Heterogeneous devices offer diverse transmission protocols and addressing systems for communication. A communication service combines several technologies together and provides a uniform interface for client and server services. It is necessary to establish connections before it is possible to transfer any information. Unicast messages are inadequate for creating connections, as it is not possible to send a message without a destination address. Therefore, broadcast mechanisms are used to introduce the presence of local node to other. Communication is made possible by extending some existing transfer protocols with a proprietary addressing policy. Addressing and transfer techniques used in this study are described in Chapter 4. The communication service grants a globally unique location identifier for itself, and it also provides service identifiers for other services running on the same node. Furthermore, it observes environment and keeps up-to-date information about surrounding nodes and connection reliability.

R3. Providing of message forwarding and routing

To be able to send unicast messages it is necessary to know both ends of the transmission and the path between end-points. The path has to be determined if it is not known, or if it is broken. This is the primary role of communication services: to forward messages to the correct destination. A path is determined by broadcasting routing queries and receiving routing replies from a defined destination if it is reachable. This method does not provide every possible path, but it provides the fastest ones. The reason for this is the design of the communication service, which trashes messages that have already been received in every node. It is correct to assume that the reply that is the quickest to arrive, certainly is not a bad one, even though it might not be the best. After an adequate path is resolved from replies, it is possible to start forwarding messages. If the path is broken, which is quite a common situation in a dynamically evolving network, a new path must be searched. It is necessary to adapt to the environment, and if displacements and disconnections of nodes are too frequent, it is possible to use broadcasting for all messages. This consumes more resources in a static environment, but might be better if the level of dynamics is high. More detailed comparisons between unicast and multicast is depicted in Chapter 4.

7.5 Implementation approach

Without verification of the ideas and concepts with a concrete implementation, it is not possible to see flaws or overseen problems in the design. Therefore it is necessary to deliberately choose an approach that is practical for the specific task. Two different verification models are apparent for Spontaneous Data Storage: implementation into physical or simulated environment. Both of these have several advantages over each other, and a comparison of these approaches is presented in the following sections.

7.5.1 Real environment

It would be natural to create a concept and test it in the real world. It is the preferable choice, if the problem field is well organised. On the other hand, it requires a fair amount of resources and time, and is a bit stiff, as it might be difficult to arrange several test placements for hardware to simulate different network circumstances. The following list represent the major edges and flaws for testing the Spontaneous Data Storage in a real environment.

- Results are based on real environments
- Environments are fixed and not easily modifiable
- Environments require proprietary hardware and is expensive
- Requires expertise in diverse technologies and protocols

As this work focuses on data sharing over heterogeneous and mobile networks at a higher abstraction level, it is not necessary to decide upon a strictly specified applicable environment. This includes utilised technologies, the amount, the placement of hardware nodes and several other factors.

With a real environment it would be possible to derive more accurate results than with simulation, but probably not to a high degree in this case. A real environment eliminates some mistakes that are present in a simulated version, mostly related to performance, and leaves some others untouched. The same false assumptions and arrangements can be made for both approaches, and the same principal mistakes that are not prevented by physical limits of the system are apparent in both environments.

A large-scale implementation for the environment is not justified as a decent value for invested capital could not be guaranteed, and a small-scale environment is not a good basis from which to derive results. Furthermore, parameters and the environment are not easily modifiable after freezing the implementation design, and as the researched technology is in an evolving stage, there might be a need for radical changes, especially if some serious design flaws are faced. Therefore, it is necessary to consider another approach.

7.5.2 Simulated environment

The skipping of a real physical environment requires more verification at the theoretical level. It is necessary for a simulated environment to correctly simulate all of the relevant characteristics of the environment, or derived results are faulty. There are several factors that are left out purposefully, as they do not cross the line of significance. The following points introduce differences between a correctly implemented simulated environment and real ones.

- Results are not based on real environments.
- Environments are flexible and easily modifiable.
- Environments do not require proprietary hardware, and therefore it are inexpensive
- Environments do not require understanding of proprietary technologies.
- Environment is an approximation of a real environment.

Results are based on simulated events, in the belief that the environment is able to approximate the real world at an adequate level. This is the challenge of simulated environments, as some of the relevant factors might be overlooked.

For Spontaneous Data Storage the importance is not placed on performance, but on availability and intelligent data distribution, and thus, the real environment does not give a meaningful edge over a simulated one. This study is focused on the functionality of a system, and it is not necessary to have a perfect real world environment to verify important goals. Even better, the simulated environment offers many other favourable reasons to make it a preferable choice. For this work, a good simulation model is better, as it is more flexible and offers a possibility to quickly create different test cases and network topologies to draw conclusions. It is cheap, and simulation results are easily achievable. Therefore, the simulated environment is chosen.

7.6 Simulation model

For coordinated implementation, the architecture of simulation is divided into six layers, as presented in Figure 8. The two lowest of these layers simulate the real physical environment, and the transmission capabilities of different networks. The ones that are not shaded form the Spontaneous Data Storage, and on the top there is a base of simulated users as an application level.

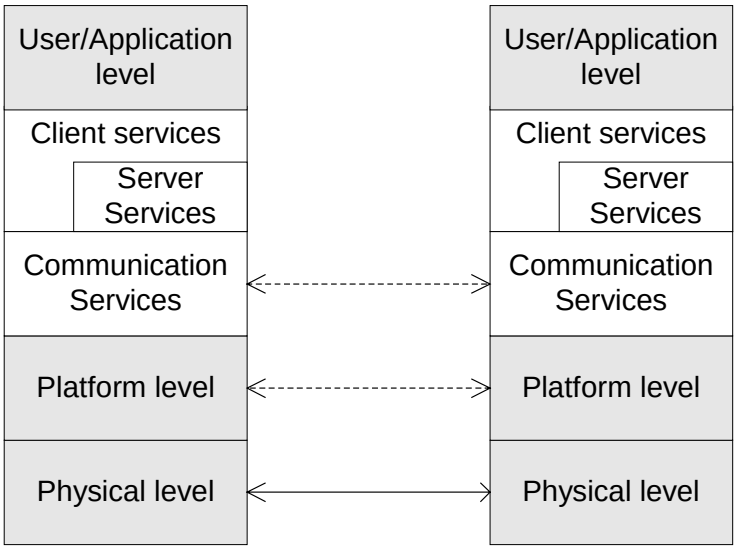


Figure 8. Layered architecture of the simulation model.

Before going deeper into detailed functional layers, it is worth noting that some functionality of the layers was left out from the final implementation, as the focus was

on analysing the fundamental ideas of Spontaneous Data Storage. In other words, analysing data distribution patterns and the viability of the data sharing approach. The programming language used to implement the simulation was Java, and the development environment was Borland's JBuilder.

7.6.1 Layers of the simulated environment

The simulated environment is partitioned into three components for easy implementation and reuse possibilities. The platform layer provides the functionality and information required from the host platform, the physical layer emulates the transmission medium and creates a network environment, and the application level simulates the data access utilisation of the created environment.

Physical layer

The physical layer simulates the transmission medium for data signals, for both wireless and wired mechanisms. Wired links have infinite range and good reliability, and wireless transfers protocols are distinguishable from each other by their different transmission radii. The physical layer broadcasts messages from a sender node to every possible receiver node at the range of one hop. The range of one hop means any reachable host without forwarding through other nodes. In other words, a broadcast message is sent to every neighbourhood node.

Two different node types are supported: fixed and mobile. Fixed nodes are able to use every transmission method, while mobile nodes have access only to different wireless connections. Table 20 on page 89 explains the availability levels of different transmission technologies for different nodes. As the list shows, the nodes can have access to different transmission technologies, and thus, messages have to be forwarded through different protocols and transfer mediums. This simulates platform independence and the option to establish connections between any two devices, if a compatible transmission method is found. Furthermore, communication can be routed through several nodes to reach the destination, and it is not necessary that the source and destination have compatible transmission technologies.

Messages are delivered through every available transmission channel and it is possible for nodes to receive the same messages on a multiple basis. This introduces unfortunate overhead, but provides good propagation of broadcasted messages to every host. The performance of different transmission technologies is not considered, as simulating of data transfers in ad hoc networks is very complex. It would be possible to make rough approximations and include some factors to determine network utilisation, but that is

not the key point in this study. Table 17 describes additional features that are necessary for simulations. The SDS needs some of the features, and the rest are to model the environment physically.

Table 17. Simulated features in the physical layer.

Feature	**Description of the implementation**
Mobility	Locations of nodes are stated in two-dimensional coordinates, and the physical layer allows modification of the coordinates
Transmission medium	Wireless transmission is simulated with transmission radius. If two nodes are in the range of any transmission method, it is possible to communicate. See Figure 9 for an example
Environment creation	The physical environment includes a method of creating different environments. The method decides the amount and locations of the created nodes, and establishes services and technologies randomly
Time simulation	The time concept used in simulation is a discrete model and allows time increments in cycles. The model allows an evolving environment to be observed and data distribution step by step. It is possible to clear a send-and-receive buffer for every node in a clock cycle
Signal reliability	Wireless transmissions include field strength that is given to the platform layer. This determines the reliability of a connection

Platform layer

The platform layer simulates only the hardware interface to an environment. In this context, the platform refers only to the hardware and software running on top of it. It does not refer to an environment in which the platform is located. An instance of the platform layer is a node, and the resources available for a node are listed in Table 20 on page 89.

The platform layer provides methods to send and receive messages to the communication and physical layers respectively. It also stores general information about the platform and full details of the most important features are listed in Table 18.

Table 18. Simulated features in the platform layer.

Feature	**Description of the implementation**
Location of node	Location is presented in two-dimensional coordinate format, as can be observed from Figure 9
Available transfer protocols	Simulated node contains a list of available transfer protocols. Available protocols can be established when the node is created, or they can be modified later manually
Established services	It is possible to establish and terminate services on a node. Terminating communication services practically removes the node from a distribution network
Allocated storing resources	Simulated node contains a profile of available hardware resources for that node. This includes storage space reserved for server services to operate
Access to global cycle	Physical layer synchronises the operation of nodes in platform layer, and synchronised nodes provide time for other services
Dozing mode	Platform layer controls also the dozing mode of nodes. It is possible to put a node into a dozing mode at will, and this results as temporary disconnection from network. Global time is still updated to allow basic operation of the services

When dozing mode is enabled in a node, all incoming messages from the physical layer are trashed. A node is in a frozen state, and does not allow any modification but movement. Basically, all of the nodes are created automatically with environment creation methods, but it is possible to affect most of the parameters with manual intervention to customise an environment for a simulation run.

Application layer

After the environment has been created and the services have been enabled on nodes it is time to observe the behaviour of Spontaneous Data Storage. The application layer simulates arbitrary data access to SDS. The full list of simulated behaviour is listed in Table 19. In fact, the application layer is party responsible for creating the environment, as part of the data sharing environment is the stored data.

The application layer has access to client services that have been established, and to the features provided through them. The possibility to track data distribution patterns and variables through server services is also included for simulation purposes. This feature does not affect the data sharing system, and only provides behaviour information about SDS. The application layer is a simulation of the real world, with the test case patterns

to invoke different data requests. It requests to store random information during the simulations from arbitrary locations.

Table 19. Simulated features in the application layer.

Feature	**Description of the implementation**
Information storing	Utilises the interface provided by the client service to store information
Information tracking	Provides a graphical interface to browse various details of data items stored in the server services. This includes access to variables available for the Smart Data Distribution Algorithm
Information fetching	Provides ability to download data items through client service interface
Information searching	Utilises client service to provide full list of globally available data items

7.6.2 Layers of the Spontaneous Data Storage

This section gives a detailed description of what has been implemented into the Spontaneous Data Storage. The data storage consists of the client, server and communication services. The responsibilities of each service have been described earlier, and this section gives a featured list of the implementation. Full functionality of the previously described ideology of the SDS is not implemented, and the focus has been on the key elements. That is, distribution, storing and accessing of data. These features and simulation environments consist of approximately four thousand code lines. For a side note, simulation results are not derived from any of the described layers, but directly from graphical interface, and therefore, some screenshots are included.

The environment window and user interface of the simulation environment are presented in Figure 9. The right-hand side of the split panel provides a view of the network topology and environment. This screenshot contains a randomly generated portable group consisting of ten nodes. Light circles show the ranges for long-range and the dark ones for short-range wireless connections. It is possible to have several groups connected with different communication technologies to model real situations, while this example topology gives simplified example. Furthermore, several of the spontaneous networks that are created in the real environment are arbitrary, and therefore, it is not rational to categorise topologies according to whether they are real or not. The example has been created randomly in the way that all of the nodes belong to a

one dynamic cluster. When this cluster is split into two, it is possible to evaluate the availability of information in both clusters. This is the method used to evaluate SDS.

The left side of the split panel provides some general information on the environment, and specified information about the selected node. The title bar shows the global time, which is calculated in cycles. One cycle is one arbitrary time unit that has been used to provide easy simulation.

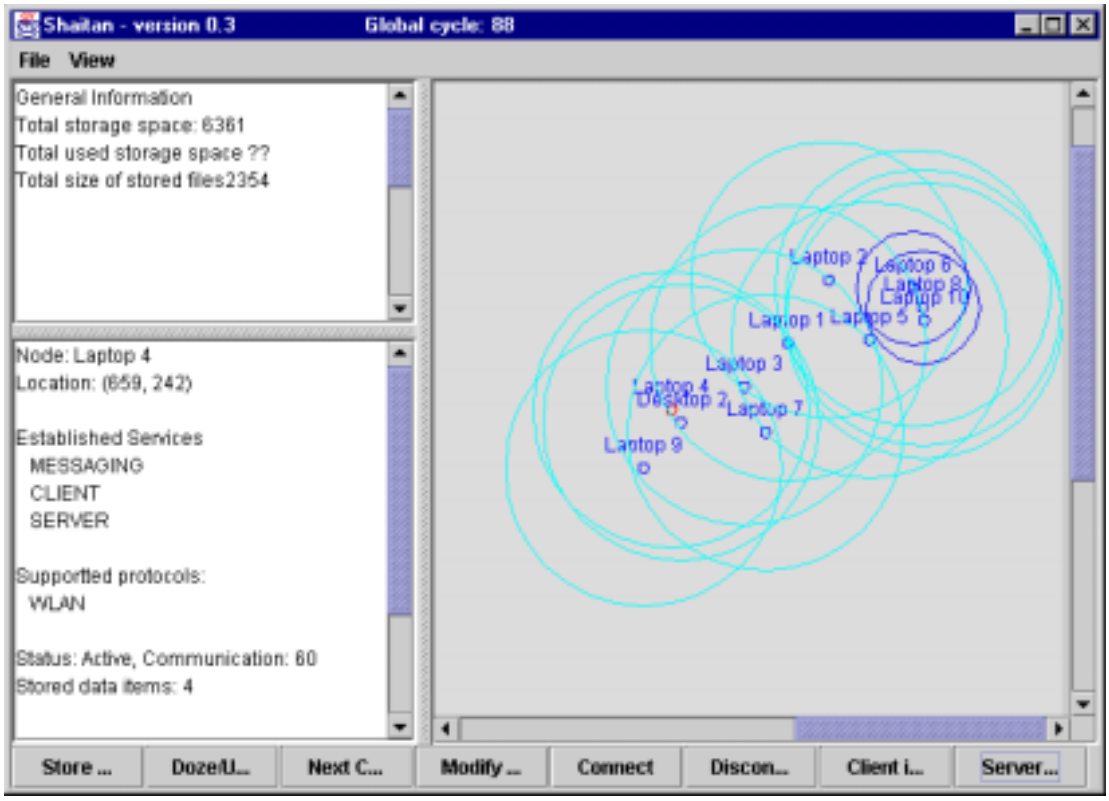


Figure 9. User interface.

Environment and data storing are usually created automatically, but it is also possible to manually modify the environment and store information. GUI in Figure 9 offers direct access to information on clients and servers to follow the distribution of replicas and to manipulate it.

Client services

It is not possible to access information that has been stored in the SDS without going through client services. This is the fundamental part of every handheld device that is interested in utilising information. Client services are not necessary for servers, as some servers might only be interested in serving information and not utilising it. The following services are offered to users:

- Data store. The client service sends an upload query into a network of nodes, and waits a certain time for replies. The best location for upload is chosen according to received replies, and the upload is performed. This is repeated as and when necessary, and after the client has verified that data it is stored into the data storage, or storing is failed, and then the user is informed.
- Data fetch. When a download request is invoked, the client service sends a download request and waits for replies. After a certain time the download is performed from the optimal location.
- Data list. The list is a small buffer of information that has been recently advertised in the data storage. It is possible to send an information search request to receive an up-to-date list of stored information. The data list is not updated actively but on demand.

Data update messages are not implemented, as it would require update integration mechanisms. The message type itself is supported but it would do nothing, and is not used. The client service is enabled randomly for simulation, and the primary contribution to the data distribution in simulation is the storing of information from several locations.

Server services

Server service contains the most important functionality needed for Spontaneous Data Storage to operate. It provides resources for data storing in the network, and utilises an advanced technology to manage and distribute information. Server services are autonomous units and act according to currently available information about surroundings, and are engaged in the nodes with enough computing and data transmission resources. They maintain the following services.

- Providing data management of stored data items. This is done with help of a data distribution algorithm detailed in Section 7.6.3
- Receiving data storing requests, and sending an accept reply to a client service if the data distribution algorithm shows enough interest towards the provided data item
- Providing a list of globally stored files. This list contains information on all the stored files, and not only locally interesting files as with the client services.
- Sending an advertisement broadcast from all locally stored files. This is done periodically, and all advertisement information is integrated into one single file.

A data distribution algorithm determines the primary behaviour of a server service. It is possible to utilise different distribution algorithms, as server services are autonomous units and act according to the needs of the user who has established them. The Spontaneous Data Storage utilises the same algorithm for all of the server services, to provide easier analysis of the system. According to the interest values given by SDDA, the server service replaces the least interesting items with others.

Server services are used to track the data distribution patterns and parameters in implementation, and the information that can be observed from a server service is depicted in Figure 10. Under the title bar there are labels to provide general information about the local host's storage space allocation, and the number of messages that the server service has handled. The left split panel shows a list of globally stored files that are locally known, the number of their locations, file sizes and locally calculated interest values. It is possible to check a file from the global list to get additional information about interest parameters in the right split panel. File (9) is checked for demonstration and additional information including storing locations, time from the last update for corresponding location, distance in hops, connection reliability and the file size are shown on the right. These parameters, and some others that are not shown, are used to calculate the local interest for data items. Locally stored files and their interest are shown in the lower part of the horizontal split.

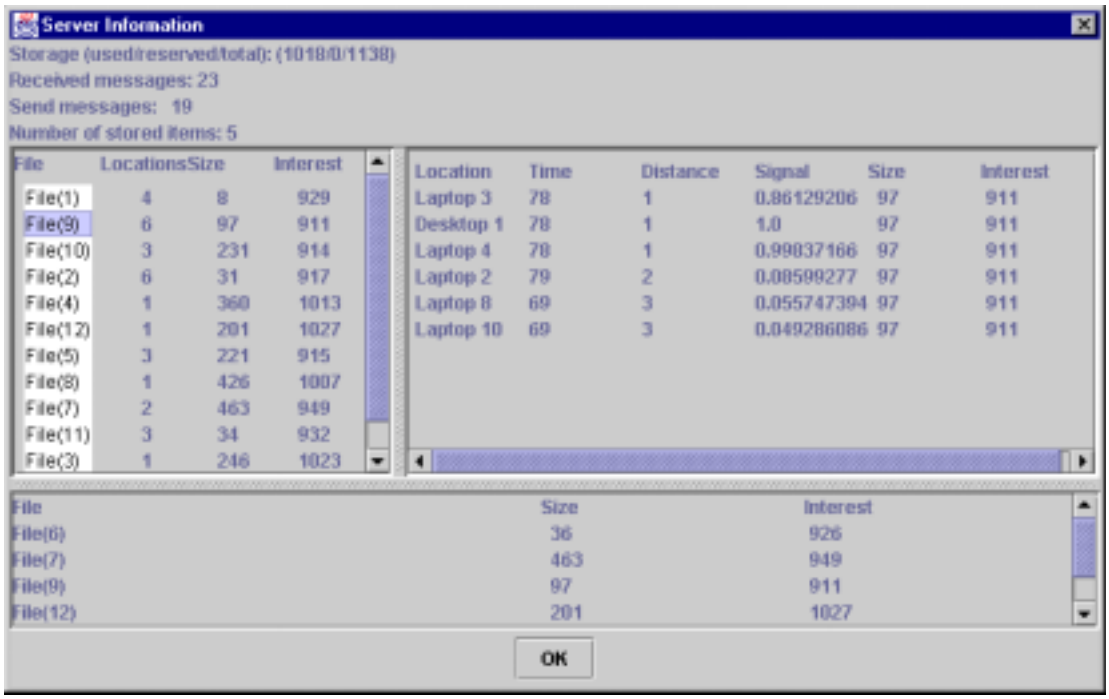


Figure 10. Server information.

As can be seen from the screenshot, information locations have not stabilised yet (during the simulations it became apparent that they will not), as there are several global files that have higher interest values than those locally stored. The allocated storage space is fully consumed (used storage 1018 and available 1138) and it is necessary to drop some locally stored files (for example, file 9) in favour of some more interesting global files, as with file (12).

Communication services

The communication services form the lower layer of the SDS. It establishes a foundation for the client and server services to function by providing message routing capabilities. It also observes the environment and is aware of every node that has a direct connection to it.

Information that is necessary for the SDS to establish itself is collected by the communication services from the platform it is running on. This includes a real time clock, signal qualities of data transmissions and other variables that are needed by the client and server services. The communication services are necessary for every host participating in the data sharing system, as it enables platform independent communication between nodes. These are the primary features that are provided:

- Globally unique address is necessary for every node participating into the SDS. It provides the possibility to absolutely identify hosts from each other and enables point-to-point transfers.
- Messaging and connections. Unicast and multicast messages are supported and delivered according to destination. Periodic broadcasts are used for advertisement messages. All of the messages share the same structure as described in Section 7.6.4, and the messaging model prevents receiving of duplicated files. The presence of neighbourhood nodes is observed with ‘hello’ messages that are broadcasted periodically to allow dozing of handheld devices.
- Global time. A real time clock or some other time source is necessary for client and server services to operate. Time is important to track time stamps of files and versions, and for the data distribution algorithm. For this study, a discrete time model of the global time cycle is used.
- Connection reliability is calculated from the sender node and is calculated at the moment of message retrieval, and the reliability field is updated to the message. The connection reliability is an important factor in determining data distribution locations.

This model provides a functionality to send and receive messages in dynamic networks, and supports the function of client and server services by providing the described features. It is possible to use advanced routing mechanisms to increase performance and message delivering reliability, but it is not necessary. Performance evaluation of the SDS is not a focus area for this study.

7.6.3 Smart data distribution algorithm

Data distribution systems require a method to choose locations to store data items, if resources are inadequate to store all of the data items in every server. This is the case with heterogeneous and portable computing devices. It is possible to have a centralised or decentralised approach to controlling data distribution patterns. In a centralised model, one master server has data distribution control over the others in a cluster. This master status can be given, or it can be claimed spontaneously in network partitions. The master server has knowledge of all the other servers in the cluster, and it is easy to choose locations in which to store data items reliably.

This study presents a decentralised approach, where is no central control. Additionally, the target environment is crucially different from classical approaches. Therefore, a unique data distribution algorithm is presented to provide an alternative way to manage information storing. The goal of the algorithm is to create data distribution patterns that meet the data access requirements described in Section 7.2. This study presents the following list as the key parameters in providing good data distribution patterns in the target environment of Spontaneous Data Storage.

- Distance. The more distant the information, the more interesting it is to store locally, as network partitions might render information unavailable.
- Redundancy. The amount of currently stored replicas in the network relates to general availability. It is preferable to fetch data items for local storing for files with low redundancy.
- Probability of removal. To prevent the extinction of arbitrary data items, it is necessary to have a hint of the probability of information being removed from other hosts; some nodes might drop a data item. The persistency of replicas increases their potential redundancy, and has to be considered.
- Reliability of connection. If connections to replicas in the network are not reliable, it might be preferable to locally store them and serve into the neighbourhood.
- File size. For efficient resource utilisation, it is not practical for minor server nodes to store relatively large files. The relative file size of allocated storage space has to be taken into account.

- Time stamps of advertisements. It is rational to remove advertised information from the history if updates have not been received within a certain time. This is necessary because server services do not inform others of local data item removals. Informing is not practical when network partitions are common.

Not all parameters from advertised data items are stored. For example, it is not necessary to store the distance to every data item, only to the closest ones. Trade-offs between complexity, required computing power and the size of advertisement history need to be made on algorithm implementation. It is possible for server services to utilise differently weighted parameters, or even totally different distribution algorithms. After all, server services are autonomous units and act according to the needs of the user who has established the service. The described parameters are used to calculate interest values towards different data items. Every server service calculates its own interests towards each data item, and makes storing decisions locally. All of the available storing resources are locally utilised, to provide maximum redundancy. If the algorithm is successful, the data distribution patterns are decent.

The developed Smart Data Distribution Algorithm is described with equations and explanations as follows. Information on the data items that are stored in the SDS is distributed with periodic broadcast advertising messages. Every server service sends advertisement messages on all locally stored data items. The messages consist of the following fields.

Data_item_advertisement =
{ distance, probability_of_removal, connection_reliability,
time_stamp, location }

A server service collects information from incoming advertisement messages, and stores one message to every replica it is aware of. If information about a replica that an advertisement message refers to has already been stored, the information is updated, and otherwise it is stored as new. Data_item_information_object stores all described information regarding to a piece of information replicated in the Spontaneous Data Storage. The redundancy refers to the amount of replicas of a data item in the reachable network.

Data_item_information_object =
{ file_size, redundance, data_item_advertisement[redundance] }

The local interest value of a data item is calculated with the following equation. The letters from ‘A’ to ‘G’ present functions that are used to create a numerical value of subinterest. The cumulative sum of subinterests equal a local interest value that is used to determine if it is worth storing a referred data item locally. Variables from ‘a’ to ‘f’

are determined to shorten the outlook of Equation 1. This equation provides the interest value that is used in data distribution management.

In SDDA, the interest value is

$$I = A(a) + B(b) + G(\sum_{i=1}^{a} C(c) + D(d) + E(e) + F(f)) \quad (1)$$

where

a is redundancy
b is file_size
c is Data_item_information_object[i].probability_of_removal
d is Data_item_information_object[i].connection_reliability
e is Data_item_information_object[i].time_stamp
f isData_item_information_object[i].distance

Figure 11 and Figure 12 demonstrate a possible undesirable situation in a data distribution system. The description of the picture is similar to the ones presented in Section 7.3 and Figure 7. In the pictures, a disjointed line represents a removed connection. If the node in the centre is removed then the network is divided into two clusters, as shown in Figure 12.

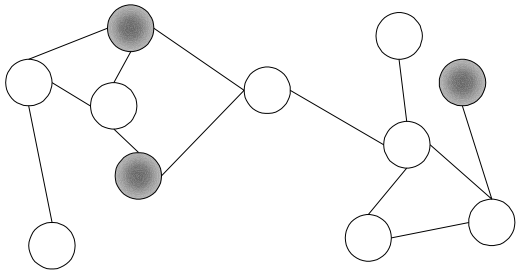


Figure 11. Initial DDP.

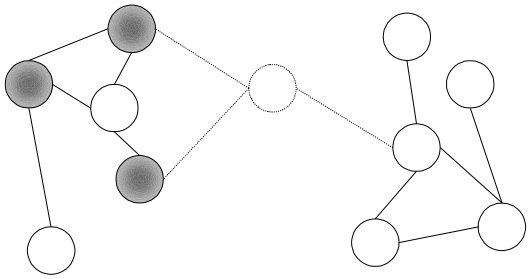


Figure 12. Eventual DDP.

Figure 11 visualises the initial data distribution pattern, and Figure 12 the eventual DDP. The DDP in Figure 12 is reached before the removal of the centre node. The figures show clearly that the observed data items are available for all of the nodes in the initial DDB, but are only available for the left-hand side of the network in the eventual DDP. The described behaviour is not desirable, as it prevents the right-hand cluster from accessing the interesting data item for as long as the centre node is absent. This example is over simplified when compared to real network environments, but it shows clearly the importance of smart data replication.

The undesirable behaviour visualised in Figure 11 and Figure 12 is not possible when data distribution is managed with the unique Smart Data Distribution Algorithm. Data items flow around the network according to dynamic interest values. Static situations, where data items are in the optimal locations, are not possible, as described in Chapter 8. The initial situation is shown in Figure 11, and the eventual situation before the removal of the central node is in Figure 13. A similar situation to thats in Figure 12 is highly improbable when utilising SDDA, when assuming that Equation 1 is decently implemented. It is not possible for information to drift into one end of the network, when the distance to the closest data item is one of the driving factors in the designed algorithm.

Not only does SDDA prevent the visualised undesirable behaviour, it also provides an even more desirable situation. The situation after the elimination of the central node in Figure 13 is not safe, which can be seen by the nodes in the right-hand cluster. A high probability of arbitrary removal of a node in spontaneous networks does exist, which leads one to the conclusion that information availability is under threat. One of the nodes in the right-hand cluster decides that it wants to download and serve a replica of the observed data item. This node is shaded differently in Figure 14. A broad line represents an utilised communication route to access the replica. As described earlier, all Spontaneous Data Storage resources are utilised whenever possible, which is the situation in Figure 13. Therefore, it is necessary for the differently shaded node to delete some of the locally stored files in order to achieve enough space for a replica of the observed data item.

This section provided the ideology behind the SDDA, and demonstrated the functionality of its implementation. The example pictures used in this section are not arbitrary. They are snapshots of simulations that have been run when the implementation was being tested. Figure 11 and Figure 12 demonstrate the early behaviour of the implemented SDDA. In the early stages the functions from ‘A’ to ‘G’, as explained in Equation 1 were not efficient in modelling the ideology behind SDDA. A lot of tinkering had to go into functions to provide decent operation in the simulated environment.

The SDDA is far from perfect, but it is a step towards allowing true decentralised data distribution in spontaneous networks. Figure 13 and Figure 14 demonstrate the successful operation of the SDDA in an overly simplified network. The simulations that were run were not as simple as the examples. Most of the time was spent using wireless networks where nodes are able to move from one cluster to another, doze and add to the general complexity. This complexity is the reason why it was not possible to do any statistical analysis of SDDA behaviour. The simulated network topologies and situations were radically different from each other, and in a limited time frame it was not possible to collect enough information to create a reliable analysis of different data distribution patterns.

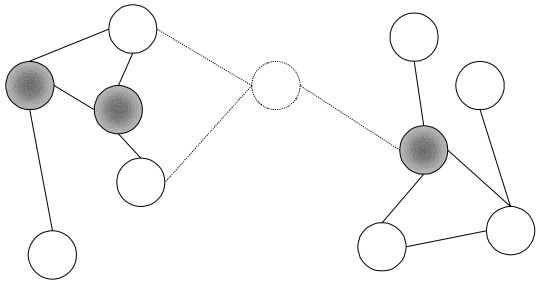


Figure 13. DDP with network split.

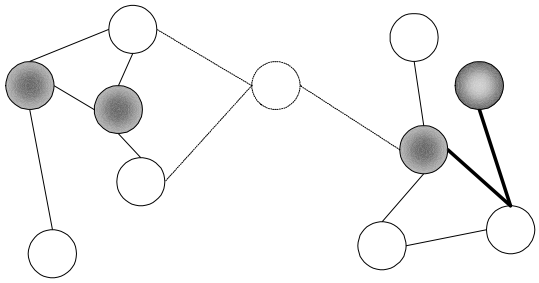


Figure 14. SDDA increases redundancy.

7.6.4 Message model

Communication between services of the Spontaneous Data Storage is established with messages. Messages consist of different fields as shown in Figure 15. A size of each block in the diagram gives some idea about relative field lengths. The destination address and routing path fields are optional, and the rest are mandatory. Not all of the fields are visible for every service. Routing path and history are visible only for communication services, and are replaced by source address for servers and clients.

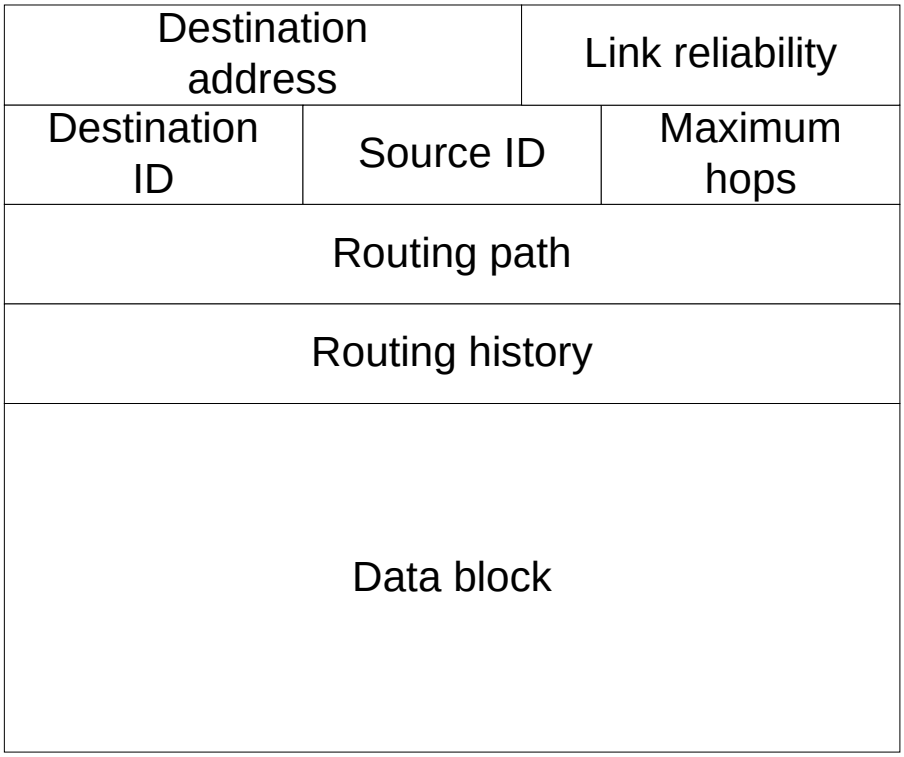


Figure 15. Fields of the message model.

Routing information between different communication services:

- The *destination address* is a unique string that identifies the destination node exclusively. It is optional, as it is possible to use in broadcast messages where a target group is not specified.
- The *routing path* determines the route that a forwarded data packet will follow. It is possible to be determined with route finding module, or it is possible to check a buffer for recent route path histories. The latter method does not utilise network resources, but it might provide an invalid route, as the network topology might have changed. This field is also optional, as it is possible to use the broadcast method to deliver messages. The communication services alternate due to different transfer methods according to dynamical level of a network.
- The *routing history* stores the address of each node in its hopping path. The first address that is stored is the source address. This is important to prevent the sending of duplicates of data items. When broadcasting is used, it is possible that messages will arrive from several different locations.
- The *maximum hops* determines the maximum distance that a packet can travel without being trashed. To prevent exponential explosion of transferred packets all over the network, it is important to define a default maximum range in hops.

Routing information from the communication layer to server and client services:

- The *destination ID* specifies the target service or services when the destination address is received. If the destination address is not determined, then the message is forwarded to the upper levels and neighbourhood nodes.
- The *source ID* specifies the identifier of the source service. It is necessary if a delivered message invokes a reply back to the source.

Information of the other fields:

- The *link reliability* is a unique feature that is necessary for the SDDA. It gives a picture of how reliable the point-to-point connections are. Every hop has a negative impact on value, which reduces reliability with a multiplier between 0 to 100%.
- The *data block* contains information that has been send from a source to a destination.

8. Validation

This chapter provides validation for the design and implementation of the Spontaneous Data Storage and introduces problems that have manifested themselves during the study. Research problems were described in the introduction, and the theory supporting design decisions has been thoroughly covered. The first part evaluates the simulation method and analyses results derived from several simulation runs. The second section provides a general evaluation of the data sharing approach, and how the various challenges to mobile distributed systems are opposed. The last section deals with some afterthoughts that have arisen during the study.

8.1 Simulation

The implemented Spontaneous Data Storage was validated with simulations. The following sections provide an insight into how these validations were performed, and what results were derived from them.

8.1.1 Simulation runs

Simulations were run in different environments, which were created randomly. Section 7.6 gives more information about the simulation environment and technology. These random environments consist of several groups of wireless and wired nodes with random storing resources according to a node type. An example environment is shown in Figure 9. The figure shows a large group of portable laptops, with mostly medium range wireless connections. This particular environment is not of great value for simulation, as relations between nodes are complex. The random creation of environments leads to several environments that are not usable, or relevant. On the other hand, example figures in Section 7.6.3 are over simplified, but they are usable to some degree. Therefore, manual modification of a network topology and nodes in the simulation environment is both possible and important.

Three types of nodes were used: handheld, portable and fixed. Some of the features that were used in simulations were dozing, mobility and different connection technologies. These connection technologies included short and medium range wireless protocols, and wired high-speed connections. These node types and technologies are adequate for modelling simple versions of target environments. Target environments are described in Chapter 7.

After the environment was randomly created and data items stored into the system, it was possible to track events in the data storage. During simulations, most of the interest was directed towards data distribution patterns, and the tuning of the factors of the distribution algorithm. This was done with automated scripts and manual tailoring. Section 7.6.3 provides two example cases of simulations. The central node in the examples has been manually removed at an arbitrary moment in time. The information required for the DDP snapshots has been recorded manually. Figure 10 shows the interface used to collect the information. It is possible to click any of the server nodes in Figure 9 to provide the server information screen.

8.1.2 Functionality of the implementation

The basic functionality of the implementation has been verified with the method described in Section 8.1.1. The examples in Section 7.6.3 are taken from real simulations and prove the basic functionality. Basic functionality and proof of it are listed in the following bullets.

- Data can be stored into SDS. One proof of this is the fact that some data items have existed in an established SDS during the simulations. A further proof is the fact that it is possible to store a new data item in SDS, even if all of the resources are utilised. Chapter 6 provides information as to why it is worth utilising all of the available resources whenever possible.
- Data can be searched and fetched from an established SDS. This can be done through a client service interface. Information storage is also performed through the same interface.
- The technology used in SDS provides good data distribution patterns, and thus, increases data availability. This can be proven with the examples in Section 7.6.3. Several other simulation runs have been performed while customising SDDA, and this tailoring has enhanced data distribution patterns.

This basic functionality *is not self-evident.* The unique data sharing approach that has been used when designing the Spontaneous Data Storage is revolutionary. The whole technology is based on the one key idea: “How can you choose a location to store information?” It is not possible to store information without first deciding on a storage location. In a truly decentralised approach, where all the nodes are equal, it is necessary to have a method for choosing a location. Negotiation between nodes is one of the possible approaches, which is viable in reliable networks. According to the technology survey in Chapter 4, the optimal alternative in mobile networks is periodic broadcasting. With periodic broadcasting it is difficult to establish negotiations to store information. How does this technology compare to other similar systems? It does not compare, as

other similar systems do not exist. Nevertheless, it is possible to question the viability of the designed technology.

One of the problems that has been introduced by this unique data-sharing concept is the preservation of stored information, which has been described in Section 7.4.2. Arbitrary loss of information did occur during simulations. The information flows dynamically and eternally from one host to others, as according to simulations it is not possible for SDS to settle for optimal data storing locations, even if the network topology stays consistent. During this dynamic flow, it is possible that some data items are lost. This behaviour scales with stored information, as storing new data items decreases the redundancy of previously stored data items. Autonomous nodes in a decentralised system cannot take full responsibility for data items if they are not able to store them all. Therefore, a minor loss of information is characteristic for this data sharing approach, in environments where the resources of servers differ on a grand scale.

As mentioned earlier, all of the available storing resources are utilised whenever possible (enough information to store). Storing space is freed up when it is required. It is required when a client service tries to store a new data item into SDS. This means that the redundancy of stored information is high if a low amount of information is stored, and low if a high amount is stored. Therefore, the more data storing resources a node has, the more replicas it will hold. This practically determines the storing locations in heterogeneous networks if storing capacities differ greatly. This leads to the conclusion that even when utilising SDS, the heterogeneous resources have a high impact on data access availability. On the other hand, SDS does not have predefined roles for nodes, which allows operation in spontaneous environments. The basic functionality has already been proven.

8.1.3 Simulation results

Section 8.1.2 proves the basic functionality of Spontaneous Data Storage. The technology cannot be compared to other similar technologies, as none exist. Therefore, it is difficult to make any perfect judgement of the viability of the developed technology. Spontaneous Data Storage can be used for mobile information sharing. It is not a reliable technology, but it is the only one that has been developed for extremely hostile environments. There have been attempts to extend classical databases to work over mobile environments, which have been too restricting. And an easy method is the local caching of information for mobile end users, which does not provide as rich possibilities as SDS does. The primary points of these technologies have been covered in earlier chapters.

It is possible to modify data distribution patterns, and hence data availability, by modifying the functions that contribute to Equation 1. Whether SDDA is able to provide high data availability remains to be seen. Simulations have been promising in environments with low complexity. When competing technologies emerge it will be possible to perform a real comparison. SDDA was able give the best results in environments that consisted of uniform nodes.

In simulated situations when an arbitrary node was removed from the network, removal did not have much effect on the other nodes in the network, but only on the removed node. Local caching has been the traditional key to overcome this, but this study presents an alternative approach. The removed node has to store the most interesting data items locally, and therefore, should map the interest patterns of the user. This factor can be implemented with the multicast subscription that has been briefly described in Section 7.4. This factor has to be reflected in Equation 1 to be in effect. It was not included in the implementation.

The following results are based on the assumption that every node has an established server and client services.

- Some *data items can be permanently lost* if the total size of stored data items exceeds the storing resources of the most capacious server node. This is normally an uncommon event, but the probability increases if the total allocation of the stored data items approaches that of the total resources allocated to data storing.
- Simulations showed that information flows from one node to others all the time. This behaviour did not change, even if the network had been in a static network topology for a long period. Therefore, a conclusion can be drawn that *it is not possible to achieve a static situation where data items are in optimal locations*. Too many variables have an effect on the data distribution patterns.
- The optimal approach for replica distribution is *to utilise all of the available data storing resources to increase availability*. The storing of new files decreases redundancy and availability, as the necessary storing space is achieved by removing some of the least interesting replicas.
- The optimal data distribution patterns are not uniform over the network. As not every node is able to store all of the available information, it is necessary to *store the most interesting information locally* to provide availability in network partitions. In other words, for good availability it is necessary to *track interest patterns of local users and applications* and to inform server services to increase interest towards specific information.

- Computing platforms differ greatly in their available resources. If the total information size stored into the data storage exceeds the storing capacity of a local host to a major extent, the local node will not be able to be of much value to a general data sharing community. This leads to a situation whereby some of the nodes take greater responsibility for data storing and management, and the others' primary value is to provide availability for local applications in network partitions. The distinction between server and client nodes is mercurial. Nevertheless, *all nodes are servers to some extent*.

8.2 Evaluation of the data sharing model

This section discusses how different challenges are answered in the data sharing approach presented here. The challenges of mobile distributed data sharing are described in Chapter 2. The evaluation approach analyses Spontaneous Data Storage at the conceptual level, and includes some references to simulations as well.

8.2.1 Scaling and heterogeneity

According to simulations, all of the devices were able to store and serve data items. Scaling was simulated with three different node types: Handheld devices, portable laptops and desktop servers. A comparison of the technology of these different nodes is in Table 20. Bluetooth is a short range and WLAN a medium range wireless transmission technology, and ADSL is a high-speed wired technology. The implementation of these technologies is described in Section 7.6.1. Different transmission technologies were assigned randomly to emulate real situations.

Table 20. Node types and included technologies.

Node type	**Storage space**	**Bluetooth**	**WLAN**	**ADSL**
Handheld	Low	Default	Rarely	Never
Portable	Average	Usually	Default	Rarely
Desktop	Excellent	Rarely	Randomly	Default

Scaling over different technologies was not painless, as was verified in Section 8.1.3. The possibility to choose between established services enhances the ability for handheld devices to take part in data sharing. On the other hand, it also restricts data access on a major scale. In a disconnected operation where none of the other nodes is in the neighbourhood, it is possible to access only locally stored information. If a server service is not engaged locally then it is not possible to have disconnected data access. If

a server service is not established, it is still possible to access data items that are in the neighbourhood, which is a benefit. The nature of SDDA and the ability to choose which services to establish, provide good adaptability to different environments. It cannot be said for certain which are the minimal hardware resources required to take part in the Spontaneous Data Storage. Establishing all of the services is highly recommended for the viability of SDS.

The functionality of data sharing services is guaranteed to be independent of heterogeneous computing platforms and transmission technologies. This has been guaranteed with the communication services, a layer between other services and communication technologies. Wrappers between communication technologies and communication services have to be created for every target platform, which is natural. Interaction between services, applications and communication technologies is shown in Figure 6.

Any device with or without storing resources has the ability to utilise data sharing services to some extent. The level of services that are available is dependent on the resources of the host accessing the services. Increased storage and transmission resources increase the availability of information to local, neighbourhood and to some extent distant hosts. This is true for target environments, which have access to multicast mechanisms. Broadcasting is a technology that allows communication without a destination address, as described in Chapter 4. The following example demonstrates the behaviour of SDS towards heterogeneous nodes.

It is possible to communicate over heterogeneous transmission technologies between nodes if any connection between different nodes to the desired data item can be found: for example, if the desired information is located in a laptop computer with WLAN support, and is desired by an application on a PDA with Bluetooth support. It is not possible to directly connect these devices, but routing through any device that has support for both the technologies provides a chance for data access. This is possible with the addressing scheme introduced in Section 7.3.

8.2.2 Data availability

The available copies method was chosen for data management, as it provides maximum availability. Further details about the technology can be found in Section 5.4.4 and in the conclusions in Section 5.4.7. According to the available copies method, every reachable data item is accessible, regardless of which version. The used ideology is “any information is better than zero information”. This ideology is the reason why it is

not possible to implement a reliable database solution based on Spontaneous Data Storage. The concept of consistency is discussed in Section 8.3.

The eventual consistency scheme is the used replication model, but it is not the only variable affecting data availability. A lot of effort has been directed towards smart data distribution. Section 7.6.3 describes and demonstrates the functionality of SDDA. Furthermore, local caching is a technique that provides an opportunity to access information without a connection to a central database. The SDS enhances this technique by also providing the locally stored information to every other node in the neighbourhood. Therefore, SDS provides better data availability than local caching when the device is not alone in an environment.

It is not possible to define an absolute level of data availability because comparison against non-existent technology is difficult. The general philosophy used in the design of Spontaneous Data Storage trades reliability for availability. In theory, the concept should provide as high data availability as possible. The eventual consistency scheme provides non-restricted data access, and smart data distribution tries to provide optimal locations for storing data.

8.2.3 Reliability

The presented data sharing approach provides high data availability, as described in Section 8.2.2. It is not enough on its own, and it is necessary to maintain the semantics of available information. It is also important that information is reliably stored. These two factors contribute to reliability.

Eventual consistency requires advanced update integration methods to integrate different versions of data items. Manual intervention can be used to resolve update conflicts in some circumstances, but clearly, it cannot be used as a primary conflict resolving mechanism. Current technology does not provide a mechanism for solving all of the arbitrary update conflicts. Consistency is important for current computing systems, as discussed in Section 8.3. Therefore, Spontaneous Data Storage provides poor reliability if updates to data items are allowed.

Section 6.2 provides a unique idea of data granulation. According to this idea, all of the received updates are integrated into a revision history that is used for conflict resolution. The idea provides possibilities and challenges, and it is too early to say which are stronger. The idea requires more refining, and might be worth further research.

If a network is split into two clusters, it is possible that some data items are not available for a certain time period in both of the clusters, if ever again as the split might not repair. This is a problem caused by a decentralised data sharing approach, as well as the weakly connected environment – and cannot be prevented. Furthermore, the persistency of stored information has been discussed in Sections 7.4.2 and 8.1. The designed data distribution algorithm is not perfect, and allows data items to be lost even without network splits. This behaviour is extremely uncommon, but it does exist.

The Spontaneous Data Storage provides poor support for update conflict resolution. The problem is not with the implementation, but with the chosen replication scheme. Current technology does not provide effective mechanisms for eventual consistency, even though it is a recognised consistency approach. Additionally, SDS is guilty of faulty operation, as it is possible to lose data items without network splits. In other words, the proposed data-sharing model provides poor reliability.

8.2.4 Mobility and displacement

The designed Spontaneous Data Storage does not place any restrictions on the mobility of portable devices. The devices are free to migrate between networks and environments, and disconnected operation is supported to some degree. In SDS, it is not possible to determine whether the network or the node is moving. And in a decentralised model it is not possible to determine if a device is disconnected from a network, or if a network is disconnected from a device.

One of the goals for the design has been that network partitions should have minimal effect on the functionality of the system. This has been achieved with a network of autonomous nodes. All of the nodes have equal possibilities and responsibilities, the roles of the nodes change only according to locally available resources.

Displacements to the devices lead to changes in the network topology and/or network partitions. A change in a network topology is dealt with routing technology as described in Section 7.4.3. Network partitions cannot be prevented, and SDS has been developed to consider these as natural behaviour. Unlimited mobility of devices is a feature of Spontaneous Data Storage.

8.2.5 Data sharing over weak connections

A decentralised approach to data management and sharing is justified for circumstances where centralised data services are not available. Many portable devices, if excluding

mobile phones, do not have access to long-range transmission mechanisms. And thus, they cannot access centralised data services. Furthermore, this study has shown that high data availability over weak connections is not possible with conservative replication schemas.

A decentralised approach with optimistic replication is successfully used in Spontaneous Data Storage to provide basic functionality for data sharing. The basic functionality has been discussed in Section 8.1.2. SDS is also designed to provide high availability to the target environment, as described in Section 8.2.2. The target environment is defined in Section 7.2. On the other hand, the SDS provides poor reliability, as explained in Section 8.2.3.

Spontaneous Data Storage provides basic functionality with high data availability for small-scale spontaneous networks. It allows mobility and disconnections with minimal impact on functionality. This is true for read only access. The SDS provides poor reliability for update access.

The SDS is a viable alternative for mobile data sharing with the following exception: Update requests cannot be allowed because current technology is inadequate in solving update conflicts with the eventual consistency schema. Research in the field of optimistic replication focuses on this problem, and major advancement in that field could remove the greatest obstacle faced by Spontaneous Data Storage.

8.3 To be consistent, is it really worthwhile?

Replication over a network introduces the possibility of inconsistency between data items if update transactions are allowed simultaneously from several locations, which is the path taken in this study. The inconsistency problem scales with replication redundancy, as it is possible to request update to several different replicas in the same file. Another viewpoint is that if redundancy is increased, then there is a higher probability that the correct information will survive. This is an interesting assumption, as it is generally not possible to determine which of the replicas provides the correct semantic for information.

The brilliancy of this viewpoint is that it leads to the conclusion that the previous sentence is fundamentally flawed. It generalises a personal opinion into a fact. It could be said that it is not possible to argue against a fact, but it is possible to argue against an opinion. There is no such thing as an absolute fact! Though, there is such a thing as an opinion supported by general consensus, which could be called an accepted theory. Theories are similar to information, and are continuously in a state of evolution. A

single person can contribute to evolving information, and a single client can send an update for a file. These two things have a clear analogy. The difference is that information contributed by a person melts into an information base, while an update to a file replaces it according to classical principles. The idea behind this thinking game is to provide insight into the problem field we are wrestling with. We are striving for absolute semantics of information for computers, while we do not have access to the absolute semantics of information outside the computers.

This leads to the conclusion that when striving for absolute semantics of information, it is easier to solve update conflicts if the level of redundancy is kept low. Therefore, the original idea for the data sharing approach came from the belief that it would be possible to guarantee consistency if the amount of replicas could be kept low, and the design focused on good data distribution patterns to provide availability. In the end it became quite clear that it is not possible to guarantee instantaneous consistency over weak connections without severely restricting availability. It is possible for network partitions not to repair, and thus, some of information will not return to the original cluster. In network partitions, every cluster is disconnected from some other clusters and therefore, disconnected operation is natural behaviour and is allowed. From this we can conclude that availability and consistency are mutually exclusive in weakly connected networks, if update transactions are allowed.

Strict consistency requirements seem to provide an unsupportable burden on mobile distributed systems. Consistent operation is impossible over weak connections, if an adequate level of data availability is required. It would be a breakthrough in the data replication field if it became possible to offer eventual consistency with technology to integrate arbitrary updates. If a peek into the real word is taken, it is unreasonable to assume that there will be no conflicts. A good example is any living being operating in its native environment. Optimistic replication is based on the idea to allow update conflicts and focus on providing a method to solve them. Whether it will be possible for applications with a high level of intelligence to integrate different update conflicts remains to be seen.

Consistency in itself is not necessary. It is a tool for providing the correct operation of applications. If the correct functionality could be provided by some other means, then consistency would not be a problem. According to previous considerations, the path taken in this study was to sacrifice reliability in order to attain good availability. Current technology might only be ready for this approach in some limited fields, for example, in documentation editing and programming. But some day it might be, in a similar way that people are able to resolve their daily conflicts.

8.4 Evaluation framework

The framework for communication and replication evaluations was presented in Chapter 3. According to the expectations stated in Section 3.1.2 periodic broadcast technology was chosen to deliver control messages in Spontaneous Data Storage, and it proved to be a successful choice.

Replication technology is more important that the communication technology and therefore, the evaluation framework for replication is more in-depth. The numerical evaluation method that was used made it easy to choose the available copies method as the optimal replication technology. The greatest emphasis in the comparisons was placed on data availability, which was the right choice. Choosing the classical data distribution path would not have made any contribution to the smart data distribution field. And when looking at the results of this study, it would not have been possible to implement Spontaneous Data Storage without an optimistic replication scheme. Therefore, the evaluation framework for both technologies was successful.

8.5 Future direction

This study has proven the functionality of the data sharing approach presented here with a question mark as far as viability goes. The implementation that was presented did not include update transactions, as the technology required for update integration does not exist. The integration of update conflict belongs to the field of optimistic replication, but nevertheless, this study provides a unique contribution for that field at idea level.

Data partitioning has been presented as a new approach to reduce network traffic and to offer version history for data items (see Section 6.2). The version history, or a similar method may play an important role in resolving update conflict in the future. It could also be used to offer a customised version of the data item to a user. Furthermore, the presented idea can also reduce network traffic.

Enhancements to Smart Data Distribution Algorithm are important in order to offer maximum availability. The implemented SDDA assumed that all of the nodes are interested in all information. This is not the case in the real environments. Therefore, the most important research target *is subscription for interesting multicast message groups*, which allows the server service to tailor data distribution patterns to suit the desires of local applications and users. This is the factor that allows disconnected operation, as a local server service is more able to focus on storing locally interesting items. Removing data from the SDS is not a problem when utilising the subscription, as the least interesting data items are replaced with more interesting ones. The removal of the data

items has not been discussed before, and it is worth giving a reason for this. The removal of all the replicas of a data item is impossible if the network splits, and never recovers. The least interesting data items drop automatically from SDS. SDDA needs the described subscription method to have a picture of locally interesting files. Also, if it were possible to derive geographical information from GPS-receivers or with any other means, it would be possible to generate far better data distribution patterns and addressing mechanisms.

Striving for highly available data access for mobile and weakly connected environments requires advancements in several fields. Routing, update integration and unique data sharing concepts, such as the one introduced in this study, are needed to satisfy the requirements of mobile computing environments.

9. Conclusion

This work introduces a unique data-sharing concept for weakly connected small-scale networks. The basic data sharing functionality of the implementation has been validated with simulations. The work is based on two technology surveys and on a personal contribution.

The first survey focused on data transfer in mobile networks, which proposes that periodical broadcast messages are important to allow non-restricted functionality in handheld devices. The viability of these messages has been verified with simulations. The technology creates energy saving features for battery powered devices, reaches all nodes in a cluster and provides information about network partitions.

The second survey focused on data replication technologies. According to the survey, the best replication technology for attaining good data availability is the available copies method. This study proves that data availability and consistency are mutually exclusive in weakly connected networks. Therefore, to provide high data availability, it is necessary to sacrifice reliability. Furthermore, this study proves that it is not possible to attain a high level of availability with current replication technology if update accesses are allowed in weakly connected networks.

The greatest contribution made by this study is towards smart data distribution. Several different variables that can be used in determining suitable storage locations for data items are introduced. A unique algorithm based on these variables is introduced for decentralised data sharing systems. The algorithm is verified to provide basic data sharing functionality with simulations. It is clear that suitable storage locations for data items are important. This is especially true for decentralised approaches – and even to some degree for centralised systems.

A unique idea about data granulation is presented in this study. The viability of the data granulation concept has not been verified, but it might provide major possibilities in different fields. It could be used to provide, for example: data update integration, customised files for users and savings in network resources.

The idea behind this work has been to create a data sharing system for mobile distributed systems. The goal has been to provide maximum data availability. Several technologies have been studied, and overall, the technology choices have been correct. Centralised service providers are unable to offer services during disconnections, and therefore, a decentralised approach provides better adaptability to the environment. Without high adaptability, it is not possible to operate in dynamic environments. The

unique data-sharing model that was introduced utilises a decentralised control model, optimistic replication and smart data distribution.

These technologies are the optimal choices to fulfil the requirements described in Section 7.2. This study proves that those requirements cannot be fulfilled with current technology, even though an enormous effort was directed towards smart data distribution for the target environment. Mobile data sharing with update transactions is not possible with current technology in a general case – at least, on the scale that was originally envisioned in this study. On the other hand, it is possible if update transactions are not allowed. On a limited basis with text editing and calendar updates, it is possible if applications provide support. The ability to update anywhere with support for disconnected operations is really beneficial for users, and might be possible with a refined version of the technology introduced here.

References

[1] Martinlassi, M., Niemelä, M. & Dobrica, L. (2002). Quality-driven Architecture Design and Quality Analysis Method. VTT Publications 456, Espoo, 128 p.

[2] Lerner, M., Vanecek, G., Vidovic, N. & Vrsalovic, D. (2000). Middleware Networks – Concept, Design and Deployment of Internet Infrastructure. Kluwer Academic Publishers, Boston, 400 p.

[3] Pitoura, E. & Bhargava, B. (1999). Data Consistency in Intermittently Connected Distributed Systems. IEEE Transactions on Knowledge and Data Engineering, 11(6): 896–915.

[4] Douglas, B. T., Theimer, M. M., Petersen, K. Demers, A. J., Spreitzer, M. J. & Hauser, C. H. (1995). Managing Update Conflicts in Bayou, a Weakly Connected Replicated Storage System. In Proceedings of the Fifteenth Symposium on Operating Systems Principles, ACM Press, pp. 172–183.

[5] Froese, K. (1995). File System Support for Weakly Connected Operation. In Proceedings of the Seventh Annual Graduate Symposium on Computer Science, Department of Computer Science, University of Saskatchewan, Saskatoon, pp. 229–238.

[6] Feeney, L. M., Ahlgren, B., Westerlund, A. & Swedish Institute of Computer Science (2001). Spontaneous Networking: An Application-Oriented Approach to Ad Hoc Networking. In IEEE Communications Magazine, No. 6.

[7] Hadzilacos, V. & Toueg, S. (1993). Faul-Tolerant Broadcast and Related Problems. In Sape Mullender, editor, Chapter in: Distributed Systems, ACM Press, pp. 97–138.

[8] Moser, L. E., Melliar-Smith, P. M., Agarwal, D. A., Budhia, R. K. & Lingley-Papadopoulos, C. A. (1996). Totem: A Fault-Tolerant Multicast Group Communication System, Communications of the ACM, 39(4): 54–63.

[9] Franklin, M. & Zdonik, S. B. (1998). "Data in Your Face": Push Technology in Perspective. ACM SIGMOD International Conference on Management of Data, Seattle, Washington, pp. 183–194.

[10] Miles, S. (14.05.2002). Networks strained by push. URL: http://news.com.com/2100-1023-278697.html.

[11] DeJesus, E. X. (14.05.2002). The Pull of Push. URL: http://www.byte.com/art/9708/sec6/art4.htm.

[12] Acharya, S., Franklin, M. & Zdonik, S. B. (1997). Balancing Push and Pull for Data Broadcast. In Proceedings of the ACM SIGMOD, Tuscon, Arizona, pp. 183–194.

[13] Weijia, J., Zhou, W. & Kaiser, J. (1999). Anycast Group Algorithms for Mobile Multicast Communications. In Proceedings of IASTED Parallel and Distributed Computing and Systems, Boston, pp. 423–428.

[14] Hadzilacos, V. & Toueg, S. (1994). A Modular Approach to Fault-Tolerant Broadcasts and Related Problems. Technical Report 94-1425, Department of Computer Science, Cornell University, Ithaca, New York, 86 p.

[15] Aksoy, D. & Franklin, M. (1998). Scheduling for Large-Scale On-Demand Data Broadcasting, Proceedings of IEEE INFOCOM Conference, San Francisco, pp. 651–659.

[16] Stanoi, I., Agrawal, D. & Abbadi, A. E. (1998). Using Broadcast Primitives in Replicated Databases. In Proceedings of ICDCS, Amsterdam, Holland, pp. 148–155.

[17] Yan, T. W. & Garcia-Molina, H. (1995). SIFT – A Tool for Wide-Area Information Dissemination. In Proceedings 1995 USENIX Technical Conference, pp. 177–186.

[18] Imielinski, T. Viswanathan, S. & Badrinath, B. R. (1994). Energy Efficient Indexing on Air, In Proceedings of the ACM Conference on Management of Data (SIGMOD), Minneapolis, pp. 25–63.

[19] Acharya, A. & Badrinath, B. R. (1996). A Framework for Delivering Multicast Messages in Networks with Mobile Hosts. In ACM/Baltzer Journal of Mobile Networks and Applications, 1(2): 199–219.

[20] Obraczka, K. & Tsudik, G. (1998). Multicast Routing Issues in Ad Hoc Networks. In Proceedings of Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking, Dallas, pp. 181–190.

[21] Coulouris, G, Dollimore, J. & Kindberg, T. (1994). Distributed Systems: Concepts and Design, Second Edition. 644 p.

[22] Guy, R., Reiher, P., Ratner, D., Gunter, M., Ma, W. & Popek, G. (1999). Rumor: Mobile Data Access Through Optimistic Peer-to-Peer Replication. In Proceedings of the 17th International Conference on Conceptual Modeling (ER'98), pp. 254–265.

[23] Demers, A., Petersen, K., Spreitzer, M., Terry, D., Theimer, M., & Welch, B. (1994). The Bayou Architecture: Support for Data Sharing Among Mobile Users. In IEEE Workshop on Mobile Computing Systems and Applications, Santa Cruz, California, pp. 2–7.

[24] Alonso, R. & Korth, H. F. (1993). Database System Issues in Nomadic Computing. In proceedings of the 1993 ACM SIGMOD Conference, Washington, D.C. pp. 388–392.

[25] Pitoura, E. & Bhargava, B. (1994). Building Information Systems for Mobile Environments. In Proceedings of the 3rd International Conference on Information and Knowledge Management, Guithesburg, pp. 371–378.

[26] Bernstein, P. & Goodman, N. (1981). Concurrency Control in Distributed Database Systems. Computing Surveys, 13(2): 185–221.

[27] Heidemann, J. S., Page, T. W., Guy, R. G. & Popek, G. J. (1992). Primarily Disconnected Operation: Experiences with Ficus. In Proceedings of the Second Workshop on Management of Replicated Data, pp. 2–5.

[28] Fischer, M. J. & Michael, A. (1982). Sacrificing Serializability to Attain High Availability of Data in an Unreliable Network. In Proceedings of the ACM Symposium on Principles of Database Systems, Los Angeles, California, pp. 70–75.

[29] Pitoura, E. & Bhargava, B. (1995). Maintaining Consistency of Data in Mobile Distributed Environments, Proceedings of the 15th International Conference on Distributed Computing Systems, Vancouver, pp. 404–413.

[30] Heddaya, A., Hsu, M. & Weihl, W. (1989). Two Phase Gossip: Managing Distributed Event Histories. Information Sciences, 49:35–57.

[31] Reiher, P., Heidemann, J. S. Ratner, D., Skinner, G. & Popek, G. J. (1994). Resolving File Conflicts in the Ficus File system. In USENIX Conference Proceedings, pp. 183–195.

[32] Ashvin, G. (1996). View Consistency for Optimistic Replication. Master's Thesis, University of California, Los Angeles, Available as UCLA Technical Report CSD-960011, 104 p.

[33] Liskov, B., Ghemawat, S., Gruber, R., Johnson, P., Shrira, L. & Williams, M. (1991). Replication in the Harp File System. Proceedings of 13th ACM Symposium on OS Principles, Pacific Grove, California, pp. 226–238.

[34] Agrawal, D., Egecioglu, Ö. & Abbadi, A. El (1997). Analysis of Quorum-Based Protocols for Distributed (k+1)-Exclusion. In IEEE Transactions on Paraller and Distributed Systems, 8(5): 533–537.

[35] Gifford, D. K. (1979). Weighted Voting for Replicated Data. In Proceedings of the ACM Symposium on Operating Systems Principles, Pacific Grove, California, pp. 150–159.

[36] Dolev, D., Keidar, I. & Lotem, E. (1997). Dynamic Voting for Consistent Primary Components In Proceedings of 16th ACM Symposium of Principles of Distributed Computing, Santa Barbara, California, pp. 63–71.

[37] Bernstein, B. A. & Goodman, N. (1984). An Algorithm for Concurrency Control and Recovery in Replicated Database Systems. ACM Transaction on Database Systems, 9(4): 596–615.

[38] Zaslavsky, A. (17.05.2002). Data Replication in Distributed Systems & Oracle. URL: http://www.csse.monash.edu.au/courseware/cse5200/ddb-l-06s/ sld001.htm.

[39] Adar, E. & Huberman, B. (17.05.2002). Free Riding on Gnutella, In First Monday, Vol. 5, No. 10. URL: http://firstmonday.org/issues/issue5_10/ adar/index.html

[40] Acharya, S. & Zdonik, S. B. (1993). An Efficient Scheme for Dynamic Data Replication. Technical Report CS-93-43, Department of Computer Science, Brown University, Providence, 25 p.

[41] Pitoura, E. (1996). A Replication Schema To Support Weak Connectivity in Mobile Information Systems. 7th International Conference on Database and Expert Systems Applications, Springer Verlag, pp. 510–520.

[42] Agarwal, S., Keller, A. M., Wiederhold, G. & Saraswat, K. (1995). Flexible Relation: An Approach for Integrating Data from Multiple, Possibly Inconsistent Databases. In IEEE International Conference on Data Engineering, pp. 495–504.

Published by

Vuorimiehentie 5, P.O.Box 2000, FIN–02044 VTT, Finland
Phone internat. +358 9 4561
Fax +358 9 456 4374

Series title, number and report code of publication

VTT Research Notes 2162
VTT–TIED–2162

Author(s)
Hongisto, Mika

Title

Mobile data sharing and high availability

Abstract

This work introduces a model for decentralised and platform-independent data sharing for highly dynamic networks. The goal is not to deliver a solution that can be considered similar to database systems. Communication and replication challenges are presented and examined from the perspective of nomadic computing. The aim has been to provide highly available data sharing over features over mobile distributed systems.

The primary functionality of the data-sharing model introduced here has been validated with simulations. A fully functional data-sharing model has been implemented and tested in a simulated environment. A simulated environment was chosen over real network conditions in order to offer data distribution pattern tracking in several different network topologies and environments.

This study provides understanding of the data sharing factors that are important in decentralised distribution models. These factors include location determination for replica storing and control message propagation over weak connections. Smart data distribution is crucial for weakly connected networks to offer access to shared resources. In heterogeneous environments, it is also necessary to consider possible transmission resource savings, and an idea for partitioning data into small elements is introduced. A data distribution algorithm based on this understanding is designed and evaluated.

According to technology surveys and the evaluation of data sharing implementation, it is clear that the optimistic replication schema is the choice for data distribution over weakly connected networks. This requires conflict solving for eventual consistency, or any equivalent method to provide the correct operation for applications. A decentralised data management model is the best alternative to operate in weakly connected networks, as it is able to function regardless of network partitions.

The data sharing approach presented here provides full adaptability to different network topologies and computing platforms, and it is able to offer data sharing services for any device to some degree. A decentralised data sharing approach introduces new challenges; these are discussed, and their viability for implementation is estimated.

Keywords
weak connections, mobile replication, smart data distribution, mobile distributed systems

Activity unit
VTT Electronics, Kaitoväylä 1, P.O.Box 1100, FIN–90571 OULU, Finland

ISBN	Project number
951–38–6081–7 (soft back ed.) 951–38–6082–5 (URL: http://www.inf.vtt.fi/pdf/)	E2SU00041

Date	Language	Pages	Price
September 2002	Engl., Finnish abstr.	102 p.	C

Name of project	Commissioned by
PLA-programme	

Series title and ISSN	Sold by
VTT Tiedotteita – Research Notes 1235–0605 (soft back edition) 1455–0865 (URL: http://www.inf.vtt.fi/pdf/)	VTT Information Service P.O.Box 2000, FIN–02044 VTT, Finland Phone internat. +358 9 456 4404 Fax +358 9 456 4374

Julkaisija

Vuorimiehentie 5, PL 2000, 02044 VTT
Puh. (09) 4561
Faksi (09) 456 4374

Julkaisun sarja, numero ja raporttikoodi

VTT Tiedotteita 2162
VTT–TIED–2162

Tekijä(t)
Hongisto, Mika

Nimeke

Tiedonhajauttaminen langattomassa ympäristössä

Tiivistelmä

Tässä työssä esitellään malli tiedonhajautukselle, joka on suunnattu dynaamisiin ja heikosti toisiinsa kytkettyihin tietoverkkoihin. Kantavana ajatuksena ei ole luoda perinteiseen tietokantasovellukseen rinnastettavaa toteutusta, vaan alustariippumaton malli, jonka tärkein kohdeympäristö on langattomat ja mukana kulkevat laitteet. Kyseiseen toteutukseen liittyen tarkastellaan kommunikointi- ja replikointi-teknologioita erilaisten toteutusmahdollisuuksien arvioimiseksi. Suoritettuun arviointiin perustuen esitellään valitut toteutusteknologiat, joilla pyritään korkeaan tiedon saatavuuteen ja käytettävyyteen kannettavissa päätelaitteissa.

Pääasiallinen toiminnallisuus esiteltävässä tiedonhajautusmallissa on varmistettu simulaatioilla. Täysin toimiva tiedonhajautusmalli on implementoitu ja testattu simuloidussa ympäristössä. Simuloitu ympäristö valittiin todellisen maailman verkkoympäristön sijasta, jotta pystyttiin paremmin seuraamaan tiedonhajautusjakaumia ja tilanteita erilaisissa verkkotopologioissa ja ympäristöissä.

Tämä diplomityö tarjoaa ymmärtämystä keskittämättömän tiedonhajautusmallin toteutuskonsepteihin ja ominaisuuksiin. Tärkeimpänä osana tätä mallia on älykäs tiedonhajautusteknologia, joka esitellään yksityiskohtaisesti. Älykäs tiedonhajauttaminen on tärkeää keskittämättömässä ja heterogeenisessa ympäristössä, kuten työssä tuodaan esille. Myös kommunikaatioteknologia on tärkeässä roolissa, joten erilaiset tiedonsiirron lähestymistavat ovat vertailussa. Tiedonsiirtoresurssit ovat hyvin vaihtelevia mobiileissa ympäristöissä, minkä johdosta esitellään uniikki ajatus tiedon paloittelemisesta pieniin osiin.

Tämä työ osoittaa, että optimistinen tiedonreplikointimalli on oikea lähestymistapa, kun pyritään korkeaan tiedon käytettävyyteen heikosti kytketyissä verkoissa. Kyseinen replikointimalli vaatii seurakseen teknologian, jolla tiedon semantiikan säilyminen voidaan taata sovelluksille. Keskittämätön kontrollinhajautus on valittu toteutusteknologiaksi siksi, että se kykenee toimimaan verkkojen jakaantumisista huolimatta.

Esitetty tiedonhajautusmalli tarjoaa hyvän mukautumiskyvyn erilaisiin verkkoympäristöihin riippumatta tiedonkäsittelyalustoista. Tavoitteena on taata mahdollisuus tiedon käyttämiseen ja muokkaamiseen ilman yhteyttä mihinkään tiettyyn keskitettyyn palveluntarjoajaan. Esitetty hajautusmalli nostaa esiin uusia haasteita ja ongelmia, jotka vaativat teknologialta harppauksia. Lopuksi, esitetyn hajautusmallin käyttökelpoisuus arvioidaan.

Avainsanat
weak connections, mobile replication, smart data distribution, mobile distributed systems

Toimintayksikkö
VTT Elektroniikka, Kaitoväylä 1, PL 1100, 90571 OULU

ISBN			Projektinumero
951–38–6081–7 (nid.) 951–38–6082–5 (URL: http://www.inf.vtt.fi/pdf/)			E2SU00041
Julkaisuaika	Kieli	Sivuja	Hinta
Syyskuu 2002	Englanti, suom. tiiv.	102 s.	C

Projektin nimi	Toimeksiantaja(t)
PLA-programme	
Avainnimeke ja ISSN VTT Tiedotteita – Research Notes 1235–0605 (nid.) 1455–0865 (URL: http://www.inf.vtt.fi/pdf/)	Myynti: VTT Tietopalvelu PL 2000, 02044 VTT Puh. (09) 456 4404 Faksi (09) 456 4374

VTT TIEDOTTEITA – RESEARCH NOTES

VTT ELEKTRONIIKKA – VTT ELEKTRONIK – VTT ELECTRONICS

Tätä julkaisua myy	Denna publikation säljs av	This publication is available from
VTT TIETOPALVELU PL 2000 02044 VTT Puh. (09) 456 4404 Faksi (09) 456 4374	VTT INFORMATIONSTJÄNST PB 2000 02044 VTT Tel. (09) 456 4404 Fax (09) 456 4374	VTT INFORMATION SERVICE P.O.Box 2000 FIN–02044 VTT, Finland Phone internat. + 358 9 456 4404 Fax + 358 9 456 4374

ISBN 951–38–6081–7 (soft back ed.) ISBN 951–38–6082–5 (URL: http://www.inf.vtt.fi/pdf/)
ISSN 1235–0605 (soft back ed.) ISSN 1455–0865 (URL: http://www.inf.vtt.fi/pdf/)