



**University of
Zurich^{UZH}**

Towards less dependence on locality through erasure coding

Submitted by: Benedikt Steger
Matriculation number: 10-735-520

Date of submission: May 20, 2018

Module name: Master's Thesis - GEO 620
Supervisors: Dr. Haosheng Huang¹
Prof. Dr. Michael Böhlen²
Faculty member: Prof. Dr. Robert Weibel¹

¹Geographic Information Science (GIS)
Department of Geography
University of Zurich
Winterthurerstrasse 190
8057 Zurich - Switzerland

²Database Technology Group
Department of Informatics
University of Zurich
Binzmühlestrasse 14
8050 Zurich - Switzerland

Abstract

Future GIS systems need to distribute data storage and -processing over multiple computers not only due to increasing storage and processing demands, but also due to new applications requiring the possibility to participate in a decentralized fashion. In the context of democratization of GIS, such a collection of computers forming a peer-to-peer (P2P) system should be self-organizing in order to be of use for the average citizen.

Optimal manual or automatic allocation of fragments of data proves to be notoriously difficult. The key is to group data that is likely to be accessed together, also known as locality. This thesis presents a new method to lower the dependence on locality by creating an overlay enriched with redundancy through Reed-Solomon erasure coding where the amount of information needed to satisfy a request is less than 100% of the amount needed without the overlay.

By going through an example, a multidimensional range query using a distributed Quadtree and the node lookup protocol Chord as proposed by Tanin, Harwood, Samet, Nayar and Nutanong in 2006, it is shown how erasure coding not only can be used for traditional fault-tolerance, but how the presented optimization method using erasure coding can contribute to both the Quadtree and the Chord protocol. In order to calculate the profitability, this thesis contributes an analysis method respecting the history of recently made queries. The optimization method increases the node lookup efficiency of Chord to $O((1 - \frac{\beta}{\log N}) \cdot \log N)$; β nodes do not have to be contacted if the history of operations make the optimization method profitable. Applied to the Quadtree, which is applied to Chord by Tanin et al., the method improves index utilization since it is sufficient to find *just enough randomly selected* peers that are within a moving window for processing a spatial query, thereby reducing transmission costs in a P2P setting. As a consequence, the dependence on locality can be reduced.

Contents

1	Introduction	1
1.1	Towards the statement of the problem of this master thesis . . .	1
1.2	The problem statement of this master thesis	3
1.3	Structure of the master thesis	7
1.4	Motivation	8
2	Elaboration of the research questions in respect to the current state of the research	10
2.1	The case for a decentralized spatial database	10
2.2	The research context in relation to this thesis	13
2.3	Research questions	18
2.4	Overview of the research maximizing locality properties in GI-Science	19
2.4.1	Skip list, Skip graph, Skip tree	21
2.5	Overview of the redundancy research in Computer Science	22
3	Method	23
3.1	Analysis method: Introduction to the cost model	23
3.2	Original methods: Basis for the redundancy application	29
3.2.1	The Core services layer	30
3.2.2	The Connectivity layer	31
3.2.3	Optimizations in existing methods	35
	Optimization done by Tanin, Harwood, Samet et al. . . .	36
	Optimization done by Chord	38
3.2.4	Optimization through applying redundancy	39
	Optimizing the method of Tanin, Harwood, Samet et al. .	39
	Optimizing the Chord node lookup	41
3.3	Reed-Solomon erasure coding	43
3.3.1	Basic idea	44
3.3.2	Galois field algebra	45
3.3.3	Calculating and maintaining checksum words	47
3.3.4	Recovering from failures	49
3.3.5	Properties	51
3.4	Applying redundancy	52
3.4.1	Adapting the redundancy	54
3.4.2	Altering Chord in the Connectivity layer	55
	Node lookup	57
	Node update, node insertion and node deletion	59
	Overlaying the moving windows and scope expansion . . .	60
3.4.3	Altering the Core services layer	61
3.4.4	A typical insertion and search procedure	64
3.5	Excursus	69
3.5.1	Changing the spatial objects in the Core services layer . .	69
3.5.2	Synchronizing the f_{min} level in the Core services layer . .	69
3.6	Interim conclusion	71

4	Analysis	72
4.1	The cost model	72
4.2	Analyzing the profitabilities of the redundancy applications . . .	73
4.2.1	Relation to caching	73
4.2.2	Altering the Core services layer by introducing redundancy	74
4.2.3	Altering Chord in the Connectivity layer	77
4.3	Analyzing excursus	84
4.3.1	Synchronizing the f_{min} level in the Core services layer . .	84
4.4	General reflections	84
4.4.1	History h	84
4.4.2	Parallelism	85
4.4.3	Effects of data sharing between moving windows on ad- dress discovery	85
4.4.4	Effects of RSEC on overall performance	85
4.5	Conclusion	86
5	Outlook	87
6	Summary	88
7	List of figures	89
8	List of tables	91
9	List of terms	92
10	Bibliography	95
11	Personal declaration	100

1 Introduction

1.1 Towards the statement of the problem of this master thesis

In data processing there exists a general ideal: EVERYTHING - ALWAYS and EVERYWHERE.

This ideal can only be achieved in an ideal world where changes are not possible. One can think of universal libraries that are located everywhere. Each library would contain the whole knowledge of humanity. Since there are no changes, the status quo of this universal knowledge remains absolutely constant. But such a world without changes is only possible if time is standing still.

By allowing time in such a world, changes become possible. As a matter of fact, knowledge will increase. Consequently, there are three adverse possibilities:

Firstly, every library of this world (EVERYWHERE) would need to collect and provide all new books (EVERYTHING). Since such an update needs time, the ALWAYS, i.e. the availability, can not be guaranteed at all times.

Secondly, if it should be guaranteed that EVERYTHING is ALWAYS available, only a central library is feasible. In such a central library, the increase in knowledge is adapted continuously and in real time. The EVERYWHERE has become impossible.

Thirdly, if knowledge is distributed to all libraries in this world (EVERYWHERE) and if knowledge is ALWAYS made available, then the content of the libraries is reduced to conservative knowledge, since current knowledge can not be feasibly adapted by all libraries (not EVERYTHING). Contrary to changing knowledge, only canonical knowledge can be made available optimally EVERYWHERE. This does not mean that updates to knowledge of certain libraries is impossible, the knowledge is just not made available everywhere optimally.

This thought experiment reveals that the ideal of having all data (EVERYTHING) ALWAYS and EVERYWHERE available is not possible in a world with time and space - our world -, since one of the three requirements always remains unclaimed in favour of the other two requirements:

- Firstly, EVERYTHING and EVERYWHERE leads to NOT ALWAYS.
- Secondly, EVERYTHING and ALWAYS leads to NOT EVERYWHERE.
- Thirdly, EVERYWHERE and ALWAYS leads to NOT EVERYTHING.

If it is wished to get away from an unwanted possibility of this list, it is only possible to select from two other equally unwanted possibilities.

This trilemma exists because of locality. The distribution of updates of the ever increasing knowledge needs to overcome distances. The distribution of data in the digital world is equivalent to the distribution of data in the physical world (libraries). An event is always happening locally. The knowledge about this event remains local until it overcomes distances and is finally available as global knowledge. Our real world exists in space and time. Consequently, the problem of locality arises.

There are two possible ways to deal with locality: either 1. locality is ignored or 2. locality is taken seriously. There are 1. attempts to escape from the dependence of space and time. These attempts optimize communication lines or

focus on computers with always increasing processing speeds. These approaches most likely encounter limits, because technical progress is hardly likely up to the task of handling the ever increasing amount of data. If locality is taken seriously (2.), the ideal is reformulated as “optimal fragment allocation”. In other words, the knowledge is distributed optimally according to the interests. Of course, the “optimal fragment allocation” remains an ideal because the future interests regarding knowledge are unknown or difficult to forecast.

An understanding of the “optimal fragment allocation” can be reached with the help of Öszu and Valduriez [29, p. 114]:

Assume that there are a set of fragments $F = \{F_1, F_2, \dots, F_n\}$ and a distributed system consisting of sites $S = \{S_1, S_2, \dots, S_m\}$ on which a set of applications $Q = \{q_1, q_2, \dots, q_n\}$ is running. The allocation problem involves finding the “optimal” distribution of F to S .

Due to the introduction of applications, the example can be detailed more specifically: The example involving books and libraries distributed knowledge universally if possible since it was assumed that knowledge, that was taking part in education, was an end in itself. In the scope of applications, however, appropriated interest is introduced. Such an appropriated interest manifests itself in telephone books, for example. The set of fragments F can be understood as the set of telephone books. The telephone books are distributed to different humans at different places. In the example, applications are the relevant operations (Q) of an human operator in relation to telephone books. Two possible operations are *read* and *write*, this is finding an entry or updating an entry in a telephone book, respectively.

Since telephone books are for a specific purpose - the retrieval of a phone number of a certain person in order to call this person -, the desire for optimal distribution in relation to locality can be explained. It is highly probable that a person gives a call to another person who speaks the same language. As a consequence, telephone books listing German-speaking members should be distributed mainly to the German-speaking area, and not in English-language countries.

The optimization of this problem is a concern in data management. In a first step, I outline two branches of science and illustrate the main ideas with the telephone book example. In a second step I explain how the two branches of science can be interlinked, from which the main problems of this master thesis are derived.

The branch of science regarding locality is concentrated on the ALWAYS and EVERYWHERE of the availability of data, always with respect to the realistic objectives of the “optimal fragment allocation”. The distributed database research tries to achieve the ALWAYS and EVERYWHERE by minimizing the distance that data has to cover. As a result, efforts are made to reduce the amount of contacted nodes as much as efforts are made to reduce the amount of exchanged data. Two approaches to a solution are of particular interest: firstly the organization along a hierarchy and secondly the organization according to distributed hash tables, or skip lists.

The main idea of the optimization through hierarchization is the assignment of responsibilities along nodes of a search tree. Applied to the telephone book example: If a telephone number of a German-speaking member is requested in

an area where German is a foreign language, the request is redirected to the correct country, the appropriate state, and down to the matching municipality, which has the needed local knowledge.

The main idea of the optimization of the distributed hash table is an intelligent search along a line or along a single-dimensional identifier space. Thereby, the costs associated with the creation and maintenance of an hierarchy are avoided, and no chaos results. The principle of a search in a distributed hash table is explained with the telephone book example as follows: Every member of the telephone system has a private telephone address list. In order to contact another member who is located in Parkes, Australia, for example, the querying member contacts the person on his list who has assumingly most likely to do with Australia. This could be a person, for example, who has made holidays in Australia. This person has met an Australian, who himself is in contact with someone living in Parkes. For the sake of this example, it is assumed that the “bush telephone” is working, because it is assumed that everybody knows everybody in Parkes.

The branch of science regarding redundancy is concentrated on the EVERYWHERE and EVERYTHING of the availability of data, also always with respect to the realistic objectives of the “optimal fragment allocation”. The main focus is on redundancy, high availability and failure tolerance. These researchers try to guard against data loss with the help of following strategies: on the one hand by distributing copies widely, on the other hand by the mathematical calculation of checksums, which allows the reconstruction of the original data packet if a data loss is observed during the transmission of the data packet. In the case of the telephone book example, this is achieved by distributing replicas to as much persons as possible on the one hand, or by constructing telephone books with checksums in the appendix on the other hand. As a consequence, it has become possible to reconstruct pages that went missing due to a careless transport, because the checksums from the appendix could be used. This principle is called erasure coding.

It is remarkable that the two branches of science, which developed by focussing on locality and redundancy, are doing research mostly independently. My idea in this master thesis is to inquire - in the sense of an interdisciplinary dialogue - what can be won if the two research efforts are merged.

- Is it possible to reasonably apply redundancy (II) onto the principle of hierachization ($I\alpha$) or onto the principle of distributed hash tables ($I\beta$)?
- How is the benefit of such applications of redundancy (II) calculated and assessed?

1.2 The problem statement of this master thesis

It is not apparent that redundancy not only can be applied to data contents, but also to data structures.

Obviously, nobody has recognized the opportunities lying within the redundancy: At the moment, files are extended by files which contain checksums for an increase in reliability. As a consequence, there are more files afterwards. Now, my idea is to increase the files directly with checksums. As a result, there are as many files after applying redundancy as there were before, but the files are each enlarged by checksums. See figure 1.

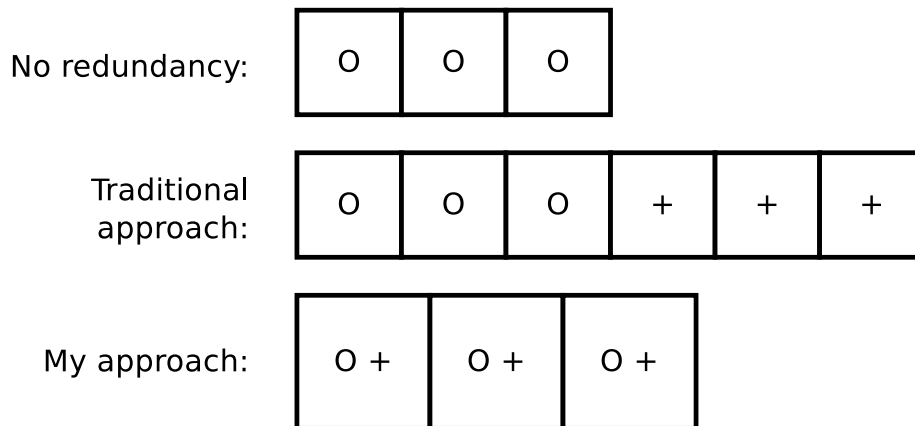


Figure 1: Redundancy (symbol +) applied to files (symbol O).

Up to now, it seems that redundancy has not been applied to structures, since almost no research can be found. If redundancy was applied to structures according to the traditional approach, additional nodes would be needed for the storage of the checksum symbols. See figure 2. With my approach of applying redundancy, no additional nodes are needed within the structure, because the checksums become integrated with the original structural elements.

With the traditional approach, the collection of needed information distributed over three nodes involves visiting all the three nodes. If one of the three nodes is corrupted, a fourth node containing a checksum needs to be visited.

With my idea of directly integrating the checksums with the original structural elements, the collection of needed information stored on the original three nodes involves visiting only two nodes. This is because the information stored on the third node can be calculated from the checksums that were additionally stored on the first two nodes. If one of the two visited nodes is corrupted, the third node has to be contacted.

In sum: with my direct extension of the original structural elements by the checksums, less nodes need to be contacted in order to reach the full information case as opposed to the traditional approach. This comparison between the two approaches even holds in the case of corrupted nodes.

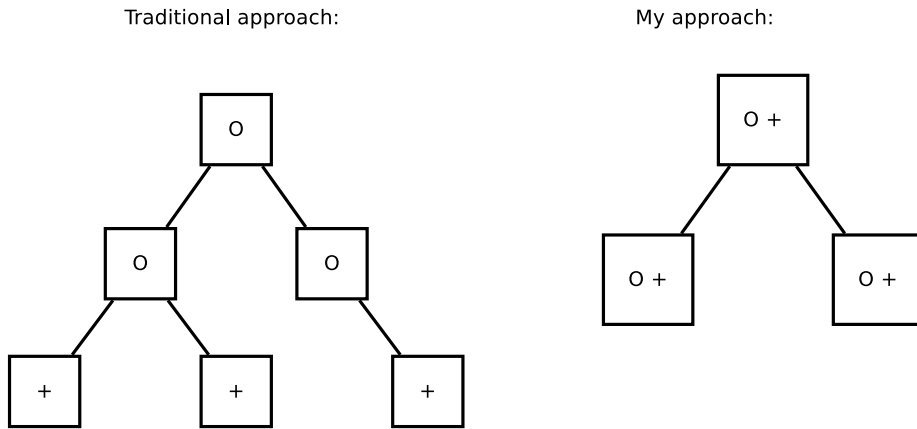


Figure 2: Redundancy applied to data structures: Redundancy (symbol +) applied to a set of elements (symbol O) participating in a data structure.

The two approaches to the application of redundancy can be exemplified in the telephone book example. It is assumed that a telephone book only contains one telephone number per page.

The traditional approach uses a single page for each telephone number and for each checksum¹. A telephone book containing the telephone numbers 43 33, 51 90 and 65 34 has following content:

Page 1:	43 33
Page 2:	51 90
Page 3:	65 34
Page 4:	15 47
Page 5:	39 78
Page 6:	0c ac

Table 1: A telephone book with an appendix of checksums.

My approach regarding application of redundancy appends the checksums directly to the telephone numbers. As a result, only the original pages are modified:

Page 1:	43 33 15 47
Page 2:	51 90 39 78
Page 3:	65 34 0c ac

Table 2: A telephone book with directly appended checksums.

¹In the following examples, the checksums were calculated using the program “Reed Solomon”, which is available at <https://github.com/tomerfiliba/reedsolomon>. Section 3.3.3 exemplifies how new checksums are calculated out of these telephone numbers. Section 3.3.4 describes how the original telephone numbers can be reconstructed out of the gathered telephone numbers and checksums.

If the telephone book has a low quality bookbinding, the pages fall apart. With the traditional approach, three pages need to be collected in order to successfully reconstruct the original content of the telephone book. With my approach, only two pages are needed for a successful reconstruction.

It is possible to argue that it is more probable to find three out of six pages, instead of two out of three. But the objection only affects the architecture of the example, which relates to figure 1 because an analogy has been made. It is easily possible to double the number of telephone book pages with my approach. As a consequence, both approaches result in telephone books with the same number of pages:

Page 1:	43	33	15	47
Page 2:	51	90	39	78
Page 3:	65	34	0c	ac
Page 4:	55	ec	ec	bf
Page 5:	07	0c	95	4c
Page 6:	ae	fb	4f	91

Table 3: A telephone book with telephone numbers with directly appended checksums and an appendix of checksums.

The traditional approach of applying redundancy still has to find three out of six pages in order to reconstruct the original telephone book. My approach continues to need two pages for successful reconstruction. As a consequence, the higher probability of reconstruction with my approach is evident.

Up to now, the telephone book example focussed on the data contents. The example can be transferred to data structures as follows:

Suppose that the single pages of a telephone book were attached to different advertising pillars. Assume that the original telephone numbers should be reconstructed from these attached pages, which is equivalent to a query of a computer. With the traditional approach, three advertising pillars need to be visited, in the case of a corrupted telephone page even four. If the pages were calculated according to my approach, on the other hand, only two advertising columns need to be visited. If the second page is corrupted, it is not necessarily needed to visit a third advertising pillar. If one half of the page was undamaged, there is enough information available for a successful reconstruction of the whole original telephone book.

By showing the possibility of an application to both data files and data structures, chances residing in redundancy are highlighted, chances that were not recognized so far to the best of the author’s knowledge.

Up to now, redundancy (II) was applied to to data contents. With respect to the elevated ideal, this is the application to the EVERYTHING, or the “what”. More concretely, it is the application of redundancy to the fragments F in the context of the optimal allocation of fragments.

The branch of science focussing on locality ($I\alpha$ and $I\beta$) related to the ALWAYS and EVERYWHERE, and subsequently on the “when” and “where”. Consequently, this branch of science optimizes the route that data takes.

This route that data takes was optimized without the application of redundancy up to now. With the help of my direct application of redundancy, the

two branches of science can be bridged, because redundancy can be applied in order to optimize the route that data takes.

The advantages of such an application can be conjectured as follows:

Assume a query, during which data has to be gathered. If contacting all nodes is not needed anymore, because contacting the most local nodes is sufficient, the risk of network congestion is minimized since data flow is more optimally spatially distributed.

Spatial dependence, which has to be taken seriously and which is taken seriously by both highlighted branches of science, is not an obstacle anymore, but also an opportunity for a reduction of the dependence on locality. Being at one place at a time is no longer only an disadvantage, but also an advantage in the sense of a locational advantage, since locational advantage is distributed on as much places as possible. Like it is possible to fight fire with fire, it is possible to largely offset the disadvantages of locality by the advantages of locality. Every place becomes attractive precisely because it is a stationary place.

This idea is the subject of the investigation in this thesis. This thesis examines the real applicability and benefit of such an application.

- Is it possible to reasonably apply redundancy (II) onto the principle of hierachization ($I\alpha$) or onto the principle of distributed hash tables ($I\beta$)?
- How is the benefit of such applications of redundancy (II) calculated and assessed?

The structure of this thesis is derived out of these main problems.

1.3 Structure of the master thesis

In order to explain how the two main questions are handled by this thesis, the structure is explained. The application of redundancy (II) to the principle of hierachization ($I\alpha$) or to distributed hash tables ($I\beta$) is verified by the section concerning the method (section 3). The verification is done by effectively applying redundancy to real systems which employ the principles. The benefit of the application of redundancy is then calculated and assessed in the analysis with the help of a cost model (section 4).

Section 3 is divided into different subsections. In the first part, the analysis method and the cost model are introduced. The second part is concentrated on describing the textbook knowledge of foreign methods and on the basic description of the application of redundancy. In the third part, an existing method which is able to calculate and apply redundancy (II) is described (section 3.3).

The fourth part of the section 3 describes how I apply redundancy (II) to the existing methods. Section 3.4.1, contains the description of the history h . Namely, section 3.4.2 describes how redundancy is applied to the distributed hash table ($II \rightarrow I\beta$). This modification also performs scope expansion. In the telephone book example, scope expansion was performed when I doubled the number of pages to 6. Section 3.4.3 explains how redundancy is introduced to an hierarchical structure ($II \rightarrow I\alpha$). In this modification, no scope expansion was performed. An equivalent alternation in the telephone book example would be the modification to existing telephone book pages by appending checksums to each existing page. The last part goes through a running example (section 3.4.4).

The section 3.5 continues with two excursus. In the first excursus in section 3.5.1, the distinction between affecting locality through altering the structure (only $I\alpha$) and affecting locality through the introduction of redundancy ($II \rightarrow I\alpha$) is shown. In the telephone book example, this distinction can be explained by comparing optimization of the process of retrieval of a telephone number through the optimization of the original pages versus the optimization through redundancy. In the second excursus in section 3.5.2, I apply redundancy to a large set of nodes. The last section answers the first main question (section 3.6).

In the analysis (section 4), a cost model (section 4.1) is presented. My cost model contains relevant parameters and allows a quantification of the profit. The specific research questions RQ1, RQ2 and RQ3 - elaborated in section 4.2 - regarding the benefit of the application of redundancy are answered by the sections 4.2.1, 4.2.2 and 4.2.3, respectively. Section 4.2.1 analyses whether the method from section 3.3 outperforms other forms of redundancy, namely making plain copies. In the telephone book example with advertising pillars, this distinction is explained as follows. In the caching approach, no checksums are calculated. But existing original telephone book pages are copied and distributed to advertising pillars. In the other approach, however, checksums are calculated. Section 4.2.1 then analyzes the different implications from the two approaches. Section 4.2.2, on the other hand, analyzes how adding checksums to existing structural elements, or how adding checksums to existing telephone book pages influence the dependence on locality ($II \rightarrow I\alpha$). The application of redundancy is compared to the method of Tanin, Harwood, Samet et al. [21, 49, 48, 47], who employ full redundancy, or who copy each full telephone book to each location. Section 4.2.3 is concerned with the modification which has to perform scope expansion in order to fit reasonably into a real structure ($II \rightarrow I\beta$). Not only are there implications resulting from applying redundancy, but also resulting from the scope expansion that has to be performed. This section derives whether the application of redundancy remains profitable and what the exact allowable share of updates in the recorded history made on this set is. In section 4.3 and 4.3.1, the limits of applying redundancy to a large set are made clear in an excursus. The remaining sections within section 4.4 are concerned with general implications. Afterwards, section 4 concludes (section 4.5).

Section 2 is important because it contains the requirements for a next-generation GIS (section 2.1), the research context in relation to this thesis (section 2.2), the research questions (section 2.3), the overview of the research in GIScience (section 2.4) and, finally, the overview of the research in Computer Science (section 2.5).

The thesis concludes with an outlook (section 5) and a summary (section 6).

1.4 Motivation

According to the declared values and the explicit mission statements that are enlisted in the mission statement of the Faculty of Science of the University of Zurich [12],

- We have the technical expertise to identify, implement, and successfully develop new and relevant research topics at an early

stage.

- We advance knowledge for the benefit of society.
- We actively foster national and international networks and support the transfer of innovations and discoveries developed by us into applications that are useful to society.
- At all levels of study, we place considerable emphasis on optimal guidance, interdisciplinarity, and the application of innovative teaching methods.

the university lecturer of the course GEO 661 has led us students to independent scientific research.

In a first step, we had to identify and define important and unsolved problems of society that can be solved with the help of GIScience.

In order to ease the identification of the problems for us students, “Drivers of Change” of Arup Foresight [3], part of the Arup University, and the 2030 Agenda for Sustainable Development of the United Nations [36] were recommended.

In a second step we had to demonstrate our competence in GIScience through tackling a research challenge. For this, we had to embed a research challenge in the current state of the research in order to weigh up on the different proposed solutions.

I evaluated the 2030 Agenda [36] as a “GIScience scientist” with knowledge in computer science. I noticed problems that I can tackle and solutions which I can contribute to interdisciplinary:

A desiderata in our global world is the *public access to information* (goal 16.10 in [36]). This is achieved by employing a system that gives access to data to all citizens, also disaggregated by geographic location. GIScience has a similar intention since research regarding participation of large masses of humans with the aim to exchange information is being done. The support for participation aims for community-defined goals, and is therefore different to the requirements for a traditional Geographic Information Systems, which are geared to the needs of corporations with bottom lines driven by profits.

The development of such a distributed spatial database supporting participation was hindered by the problem of locality, as it was set out in detail above.

The concrete placement in the research and the exact elaboration of the resulting research questions follows (section 2).

2 Elaboration of the research questions in respect to the current state of the research

2.1 The case for a decentralized spatial database

A substantial amount of challenges faced by humanity is listed in the 2030 Agenda for Sustainable Development of the United Nations [36] in form of goals and targets. The ambitious agenda proposes targets such as eradication of poverty and hunger, universal primary education, gender equality, the empowerment of women and environmental sustainability amongst other targets [36].

In order to assess how GIScience can contribute to those challenges it is important to lay Geographical Information Systems (GIS) into the societal context. According to John Pickles [30], the classic discussion amongst geographers sees either GIS from a technicist, instrumentalist or from a positivist point of view, or, on the other hand, sees absolutely no implications in the use of GIS at all. Since GIS are developed according to certain interests, GIS have an important impact. This is prevalent even though GIS is used as a tool, where the social implications are dependent on how the tool is used [30]. For example, geodemographic information systems suppose socially homogeneous neighbourhoods, and therefore allow the construction of individual profiles out of neighbourhood-derived profiles [30]. In the course of democratization of GIS, such as the GIS-2 in [30], certain challenges must be tackled. The historically grown requirements of GIS as defined by large public or private institutions are according to John Pickles surveillance, continuous monitoring, ownership and control [30]. A democratized GIS, on the other hand, makes access possible, allows participation and involves the community in the GIS, leading to informed citizens [30], public participation, as mentioned by Sui and Goodchild [45] and, subsequently, feedback, as outlined by Janssen, Charalabidis, Zuiderwijk [25]. Therefore, GIS-2 must be of use for a wider range of users, and, consequently, must be more accessible, cheaper and more flexible [30]. By exploiting the emerging possibilities of technology, such an accessible GIS-2 will be able to integrate multiple data components in one single interface [30].

The goals stated by the United Nations are similar to GIS-2 in spirit. The United Nations state in the preamble of the 2030 agenda [36] that they focus on the weakest since they do not want to leave anybody behind. Education is the key medium for communication in order to provide technologies [36]. GIScience must be usable and available for the least developed countries. This is achieved by closing the digital divide, that is, providing universal and affordable access to the Internet and providing information and communication technology, as mentioned in goal 9.c and 17.8 of the 2030 Agenda [36]. The call "to strengthen the scientific and technological capacity to move towards more sustainable patterns of consumption and production", as mentioned in goal 12.a, is aiming for the introduction of access to science and therefore promotes the usability of GIScience, too. This introduction of access to science, as mentioned in goal 17.6 [36], is envisioned to be done with the "Technology Facilitation Mechanism", as mentioned in §70 [36], which will host a website that provides science technologies [36].

Interestingly, the UN are interested in private business activity since this leads to economic growth and job creation, as mentioned in §67 [36]. According

to Pickles [30], GIS is influenced by demands of corporate interests and has unclear forms of access to information that systems promote and deny. The question is how a win-oriented business can be supported by the UN only because it promises productivity. The UN state in the same paragraph that several standards and agreements must be met to alleviate the negative consequences that are introduced by the necessity of win-orientation. In the context of GIS, this will enable access for the general public in the best case, or, in the worst case, promote established behaviour of private businesses, if Human Rights are not enforced as it is required in §67 [36].

In line with the envisioned GIS-2 [30], the United Nations are calling for *public access to information*, as mentioned in goal 16.10 [36]. The democratization of data is also pushed by the UN in goal 17.18 [36]. In the context of capacity-building, there should be a general availability of high-quality data, including the possible disaggregation by geographic location. As a matter of fact, the UN want to strengthen the capacity of developing countries of "statistical offices and data systems to ensure access to high-quality, timely, reliable and disaggregated data" [36]. A special focus is laid on the contribution of earth observation and geospatial information, as mentioned in §76 [36]. The development and implementation of the goals are controlled by the UN by using evidence which is also disaggregated by geographic location, as mentioned in §74.g [36].

The call for public access to data as a means for transparency should not result in blind publication of all data, including personal data. With the illustration of the universal libraries, which are located everywhere, in the mind, it should not be the case that all diaries and photographic albums of each person should be accessible to the public. First and foremost, the mentioned goal 16.10 [36] explicitly calls for "protection of fundamental freedoms", and links thereby to both the copyright and privacy debate. The UN want to have such a protection "in accordance with national legislation and international agreements", hinting to the Berne Convention [36]. Furthermore, the idea of unrestricted dissemination is, as stated by Janssen, Charalabidis and Zuiderwijk, a myth that surrounds open data [25]. Not only that such a distribution of public data may effectively bring no benefits, in addition there would be a complete lack of quality control [25].

Regarding myths surrounding open data, pure publications do not beget the targets such as engagement of the public [25]. It is necessary to close the loop between the government and the governed people by introducing a feedback mechanism [25]. Within the context of the universal libraries, which are located everywhere, it should be the society which decides which diaries and photographic albums are a common property and, subsequently, publicly accessible. For example, the diary of Anne Frank is valuable to humanity. Similarly, a photographic album of Joseph Beuys had to be a common property, because he claimed: "Every human being is an artist!". On system level, this means that it should be possible to contribute back and that frequent changes are accepted. These changes are then available as basis for further changes, thereby closing the loop [25].

A possible way to achieve general availability of high quality data, as mentioned in goal 17.18 [36], would be the distribution of the data. This is both an organizational issue as well as a technical one. From the organizational point of view, there are arguments for and against applying decentralization as an answer to guaranteeing access and participation [5]. The key arguments revolve

around the benefits of distribution versus loss of mandatory control, even if the control is in the hands of a large public group [5, 25].

The technical aspect is largely decoupled from the organizational one. For example, it is possible to provide a system on a central server that allows collaboration, participation, crowdsourcing et cetera [6]. On the other hand, it is possible to outsource authoritative spatial data in the context of Location Based Services on clients with mechanisms that verify the integrity of data [7, 22, 54, 23].

In this scientific context, this master thesis selects a decentralized system without the control of a single organization since the system should be capable of organizing itself [30]. The crucial step for the integration of multiple data components in one single interface [30, 39] is done in this master thesis by focussing on the location of data alone and organizing computers optimally in relation to the geographical location of data and to the access patterns in the case of a GIS. This master thesis includes a decentralized index over continuous spatial data, which differs from linked data where ontologies link entities together [43]. Such an index supports both feature-based and field-based geospatial datasets.

By making the organization of spatial data in a decentralized environment as an automatic process, Sui and Goodchild [45] argue that a step is made towards the convergence of the three categories of people and organizations: 1. the ones who create data, 2. the ones who collect data and 3. the ones who have the expertise to analyze the data [45]. This way, they argue, that social and political issues of corporations with bottom lines driven by profits are mitigated by reintroducing the concerns of the public good [45]. The goals of geographers to engage the public and possibly change the world in meaningful ways becomes more possible by introducing a mechanism that enables resource discovery and data insertion [45]. Such a mechanism eases the illustration of issues ranging from earthquake relief and environmental disasters to human rights abuse [45]. By applying such a mechanism, more prominence of the development of spatial intelligence in education is less needed [45]. Especially for developing countries, access to the technology GIS allows the public to get an impression of what is happening in the country - the typical control function of maps. This not only helps to prevent crimes hindering sustainable development, but also nourishes and promotes the possibility for a sustainable, functional democracy. By allowing open read and write access to geographical data for the general public, democratic societies, as well as public participation, as well as community-defined goals are supported [6, 30, 36].

Summarizing, a GIS-2 requires on the technical level that a Peer-to-Peer infrastructure can be used. Such a system is only feasible if the system is efficient enough for the average user.

2.2 The research context in relation to this thesis

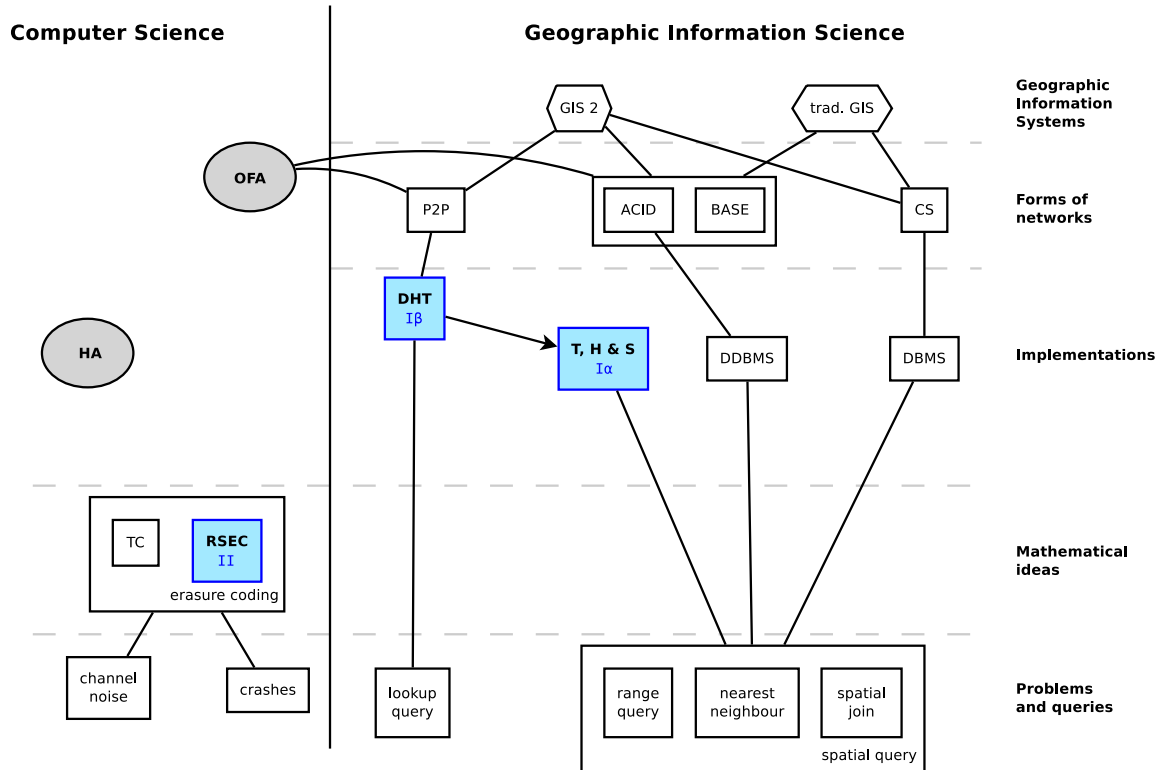






Figure 3: Structured overview on the relevant research.

Legend:

   	<p>Relation</p> <p>Application</p> <p>Influence</p> <p>Solution for</p>	<p>GIS: Geographic Information System</p> <p>trad GIS: Traditional Geographic Information System</p> <p>GIS-2: GIS-2 (John Pickles)</p> <p>CS: Central server</p> <p>BASE: Distributed servers, outsourced GIS (eventual consistency approach)</p> <p>ACID: Parallel cluster (guaranteeing ACID properties)</p> <p>P2P: Peer-to-Peer</p> <p>DBMS: Database Management System</p> <p>DDBMS: Distributed Database Management System</p> <p>T, H & S: Method of Tanin, Harwod & Samet et al.</p> <p>DHT: Distributed Hash Table</p> <p>RSEC: Reed-Solomon erasure coding</p> <p>TC: Tornado codes</p> <p>HA: High Availability, caching</p> <p>OFA: Optimal fragment allocation</p>
--	---	--

In Geographic Information Systems, following three spatial queries are supported: spatial join queries, nearest neighbour queries and range queries.

Spatial join queries support the selection of relevant spatial features based on their location or attributes and foreign location or attributes. For example, it is possible to select all restaurants within Zurich.

Nearest neighbour queries support the selection of the nearest spatial features in relation to a point. The nearest neighbour query may be limited by the maximal distance or the maximal number of nearest features. For example, a nearest neighbour query finds all restaurants that are 500 metres away from a given location, or finds the 50 nearest restaurants.

A range query defines a range. In the one-dimensional case, a range window specifies the interval within which all requested values lie. For example, all objects within a specified interval on a street are returned. In the two-dimensional case, the range query specified a two-dimensional window. All spatial features lying within the query window are returned. For example, all spatial objects of a city are returned. In the three-dimensional case, the range is a box, and the airspace of the city is included.

All queries are accelerated if the data structure is ordered according to specific rules that are geared towards the expected type of queries. An hierarchical, recursive tree is an effective choice for a large number of query types since it is possible to request roughly only the parts within a data structure that are effectively needed. All unneeded objects are quickly identified as unneeded, and not considered furthermore. This is called pruning.

Tree structures are well supported in a Database Management System (DBMS) on a central server (CS). Distributed database management systems consist of a collection of computers that are centrally organized. Such a collection may be in the form of a parallel cluster, which is supporting ACID properties, or in the form of decentralized servers or outsourced GIS, which rely on eventual consistency (BASE). There are existing solutions regarding distributed storage and querying of spatial objects.

Peer-to-Peer system (P2P) are much more difficult to support. The introduction of Distributed Hash Tables (DHT) enabled the effective processing of lookup queries, this are queries that can fetch spatial objects based on a given search key. Chord [44] is a lookup protocol operating on a DHT.

Spatial queries are more complex queries than lookup queries. While it is possible to support more complex queries by performing a sequence of more complex basic queries, such an approach is not feasible since it is inefficient.

Tanin, Harwood, Samet et al. [21, 49, 48, 47] tackle the problem of supporting complex queries with basic queries. They apply an hierarchical tree supporting complex queries to a line, a DHT. Since efficiency is a major requirement, especially in the context of P2P systems, Tanin, Harwood, Samet et al. [21, 49, 48, 47] and Chord optimize queries. In the context of a functioning smart city project with a high amount of participants and data movement, these implementations should be improvable in relation to the optimal fragment allocation (OFA).

The optimal allocation of fragments results in the fact that data is distributed in relation to the expected queries. In the concrete case, this means that the actual location of a computer or server is considered, as well as the location of a client. As a result, data is distributed amongst the network in such a way that the data is placed to the requests as local as possible. The data

is placed where the probability is highest that the data is requested.

As a matter of fact, data is made available to the location where the data is needed most. Computer science also strives for availability by introducing high availability, that is, striving for complete datasets if possible. High availability (HA), however, has roots in methods that are concerned with different objectives.

During the transmission of data over a noisy channel, data loss occurs. During the storage of data, crashes may happen and again, data loss occurs. Motivated by the problems of data loss, Reed and Solomon [35] enriched signals mathematically with redundancy in order to filter out noise. The main idea consists of providing a context, within errors such as a verbal error may be corrected [15]. If the context of a conversation is about radio telescopes, it is possible to replace wrong spelled words such as “periscopes” with the correct word “telescopes” [15]. This is called erasure coding.

The communication with the space probe Voyager II occurred over a noisy channel. The Voyager II space probe enriched the messages with redundant bits. As a matter of fact, the communication became faster as a whole, even though more information was sent at a time. Since the mathematical context provided enough information, less subsequent requests for retransmissions had to be made.

Erasur coding has the roots in the reconstruction of data. It has become possible to restore damaged data and it has become possible that failures are handled reliably. This situation lead to the question whether crashes and channel noise could be used for high availability. Dealing with errors requires the calculation of additional information, and, subsequently, one has to deal with additional information. For the case of high availability, this additional information is not erased and, as a consequence, is usable for further applications.

My question is, whether it is possible to use the high availability principle in order to gain an advantage in relation to locality.

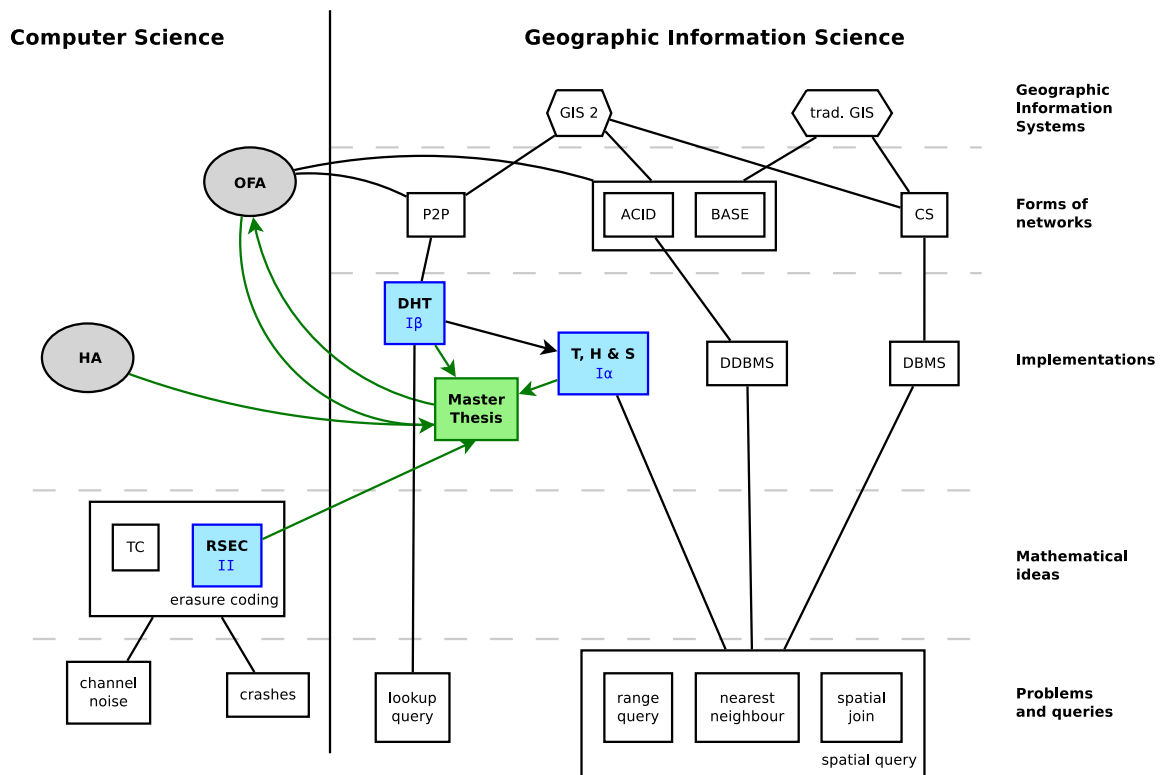


Figure 4: Context of this master thesis.

Legend:

	Relation	GIS:	Geographic Information System
	Application	trad GIS:	Traditional Geographic Information System
	Influence	GIS-2:	GIS-2 (John Pickles)
	Solution for	CS:	Central server
		BASE:	Distributed servers, outsourced GIS (eventual consistency approach)
		ACID:	Parallel cluster (guaranteeing ACID properties)
		P2P:	Peer-to-Peer
		DBMS:	Database Management System
		DDBMS:	Distributed Database Management System
		T, H & S:	Method of Tanin, Harwod & Samet et al.
		DHT:	Distributed Hash Table
		RSEC:	Reed-Solomon erasure coding
		TC:	Tornado codes
		HA:	High Availability, caching
		OFA:	Optimal fragment allocation

Efficiency is a key requirement of GIS-2. Existing solutions, namely the method of Tanin, Harwood, Samet et al. [21, 49, 48, 47] and Chord [44], optimize for locality in order to become efficient. I optimize again, but this time by enriching existing data structures with redundancy.

By doing this, the idea of high availability is applied in this master thesis: redundancy is not only used during emergencies or failures, but as an improvement during normal operation. This improvement is made in order to make spatial queries running within the context of a decentralized spatial database operating in a P2P network more efficiently.

It is a justified hope that the taking of the optimal allocation of fragments seriously does not only result in the fact that my method has contributed to more efficiency, but that this master thesis has made a contribution, which allows to move towards less dependence on locality through erasure coding.

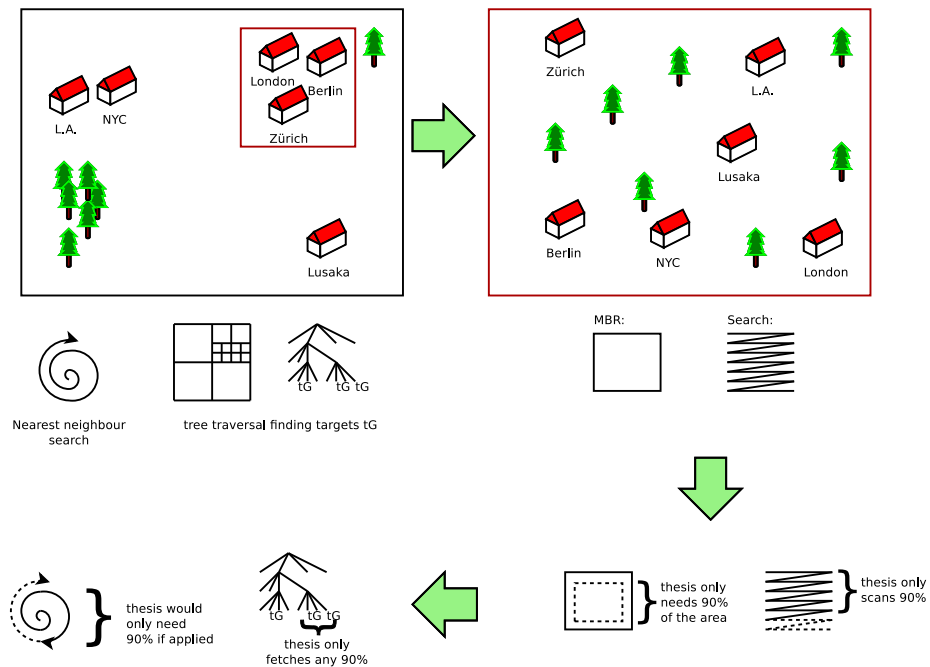


Figure 5: The range query (red) has to be performed. Assume that spatial dependence is destroyed completely. A method that achieves to perform a spatial query better than full linear scanning is needed. If the method improves over no locality successfully, it also performs better on any method that it piggybacks to, subject to the quality of adaption. Therefore, any underlying method that has few locality properties can gain locality properties from the piggybacking method. Even though the underlying method groups spatial data suboptimally, by applying the method of this thesis, spatial data is grouped more optimally since it is possible to find any 90% nodes that are enough to answer the spatial query 100%. In any case, there is no need to find all suboptimally grouped spatial objects.

2.3 Research questions

The architecture employed by Tanin, Harwood, Samet et al. [21, 49, 48, 47] is the Open P2P Network architecture (OPeN) consisting of the *Application layer*, the *Core services layer* and the *Connectivity layer*. The application logic is confined in the Application layer [47]. The Core services layer keeps consistency and performs the actual range search and relies on the lowest layer to perform key-based routing [47]. The Connectivity layer has the task of routing, dealing with object replication and migration, handling the introduction and departure of peers called churn, and providing connectivity amongst participating peers [47].

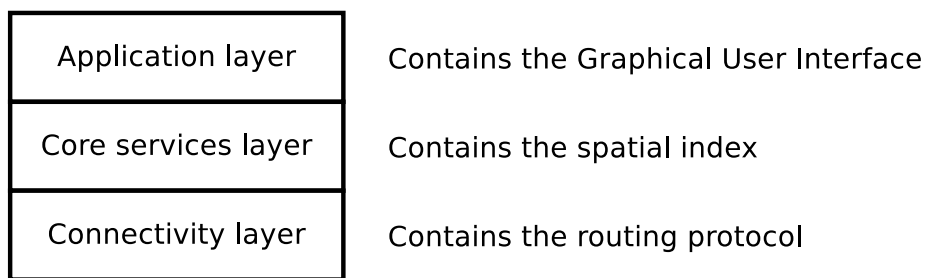


Figure 6: The layers of the Open P2p network architecture (OPeN). Adapted from Tanin, Harwood, Samet [47].

A core service adapted to a certain application has an interface [48]. All services make use of the underlying layer in a consistent manner, allowing the replacement of the lower Connectivity layer [48]. The participants in the system running a core service can answer queries stated by the Application layer [48]. For example, range queries, insertion and deletion of spatial objects are handled by the Core services layer.

Supporting multi-dimensional queries over P2P systems consists of partitioning and routing [14]. Since the OPeN architecture abstracts the P2P protocol away through a layered approach, it achieves a separation of the spatial index and the routing protocol [48]. Still, the Core services layer is concerned with routing to a certain degree. Routing through the tree is handled by the Core services layer and routing to nodes is handled by the Connectivity layer. This explains why the routing cost in the analysis of Tanin, Harwood, Samet et al. [48] is affected by both the Core services layer and the Connectivity layer.

The thesis is concerned with placing redundancy strategically and analyzing the effects of piggybacking such a redundancy in the spirit of RSEC. Applying RSEC is a success if the total number of contacted nodes is smaller, even in the light of updates. There are three important concerns that need to be dealt with.

The first concern is about whether RSEC alone is worthwhile regarding the number of contacted nodes. Does RSEC reduce the amount of contacted nodes? Is RSEC better than the alternative caching?

The second concern: It is interesting whether RSEC, which is known for its high repair costs, can lower the costs of the method of Tanin, Harwood, Samet

et al. [21, 49, 48, 47]. They use full replication, and all elements are affected by a change, too. The introduction of RSEC is only worthwhile if updates are not too expensive.

The third concern: Updates to the sets prove to be the decisive factor regarding profitability of RSEC. It is interesting to analyze how RSEC changes the required ratios of read operations and update operations in order to remain profitable. This is done for a situation where the integration of RSEC also affects the underlying data structure.

- RQ 1** Does the introduction of redundancy result in a lower number of contacted nodes during read operations than without? Is copying elements of the set to multiple participating nodes more efficient in terms of contacted nodes than Reed-Solomon erasure coding?
- RQ 2** Assuming the method of Tanin, Harwood, Samet et al [21, 49, 48, 47] as an underlying method with nodes in a set P , each holding one copy of a spatial object: Does the introduction of RSEC-calculated devices over the nodes in P result in less contacts for a given history h of read operations and update operations than without?
- RQ 3** How is the exact maximal share of update operations of a given history h determined so that the introduction of RSEC is still worthwhile, even if the identified set has been expanded and is distributed over more nodes than before?

2.4 Overview of the research maximizing locality properties in GIScience

There are a lot of studies aiming at reducing the number of nodes that have to be visited, or aiming at grouping data that is likely accessed together during range queries in a distributed setting. The approaches try to minimize the query latency.

Globe [51] stores objects in a distributed tree. These objects are situated on multiple levels. The level is dependent on whether the object is mobile or not, that means, whether the object has updates to its location. The addresses of the object are maintained amongst the participating nodes. Pointer caches within the hierarchical search tree lower query latency. Globe performs location management and therefore maximizes locality properties.

Karapiperis et al. [26] and Gupta et al. [19] use locality preserving Locality Sensitive Hashing (LSH), and thus optimize the above-mentioned locality requirement number $??$. The idea is to preserve spatial dependence during the conversion of higher dimensions to lower dimensions. Similarly, Ganesan et al. [14] employ SCRAP, which maps multidimensional data to single-dimensional data using a space-filling curve. The problem is that it is always possible to find points that are simultaneously near in the multidimensional space and far apart in the single-dimensional space [8].

Tanin, Harwood, Samet et al. [21, 49, 48, 47] all build on the idea that the overlay of a spatial data structure can maximize locality properties in an environment of distributed hash tables that destroy any spatial dependence by hashing the keys. The proposed indexes are an adapted MX-CIF Quadtree or

its three-dimensional variant, the MX-CIF Octree. Resilience against network failures, for example, is handled by the underlying layer. Chord [44] is used as the underlying method, but the authors stress that the underlying method can be exchanged easily in order to keep up with recent developments.

Gu et al. [18] also support a distributed tree. Contrary to the above-mentioned tree overlay [21, 49, 48, 47], which uses recursive decomposition as the space partitioning strategy, the proposed tree of Gu et al. has a direct mapping. The method of Tanin, Harwood, Samet et al. [21, 49, 48, 47] has a sensitive underlying topology originally designed for the exact key query [18]. Gu et al. [18] propose the Hierarchically Distributed Tree that supports multidimensional range queries at the structured P2P overlay layer [18]. As a matter of fact, locality properties are maximized by the application of a structure geared towards the expected type of queries [18]. Interestingly, Gu et al. [18] state in 2013 that the recursive decomposition as an alternative to direct mapping only supports range queries by relying on the three structures Chord [44], the Content-Addressable Network [34] or skip graphs [18].

The approach of Tanin, Harwood, Samet et al. [21, 49, 48, 47] exhibits similarities to PGrid and the shower protocol both mentioned in the analysis of Blanas and Samolads [9]. As the method of Tanin, Harwood, Samet et al. [21, 49, 48, 47] is based on a volume-based topology, it belongs to the faster group in the analysis of Blanas and Samoladas [9], as opposed to data-balanced topologies such as those employing kd-tree, like MURK employs [14]. The MX-CIF Quadtree can also support non-spatial data in higher dimensions, as long as the range is similar [37].

Methods having the locality awareness property connect to near nodes by considering the location of the nodes [50]. Dynamic Prefix Average Distance (DPAD) identifies nodes that become possible landmarks. These landmark nodes cluster other nodes under them if the other nodes have the same prefix. A joining node measures the lowest latency, which is also dependent on distance because of the finite speed of light, and prepends the prefix of the fastest node to its key. Therefore, the nodes are grouped according to the geographical location [50]. The method of Toda et al. [50] minimizes the dependency on landmarks, while still exhibiting the locality awareness property. By assigning membership vectors that are close to other nodes that are geographically close to each other on a number line, and by implying that this improves the performance, the authors assume the validity of Tobler’s law implicitly [50].

The method of Plaxton et al. [33] is also locality aware. In fact, the method is concerned with answering the queries as locally as possible, with the help of replicas. Caching is a dominant strategy: each node caches information gained from visiting queries.

Bisadi et al. [8] reference more recent work of customized indexing supporting multiple dimensions. The trend is towards lower number of contacted nodes during queries and lower per-node state. The focus is on the design of the structures. For example, Sioutas et al. [42] employ a D^3 -Structure.

Spatial decomposition can also be done with Voronoi overlays, that can be hierarchical or not [13]. Voronoi-based systems have to deal with the potentially large number of possible neighbours.

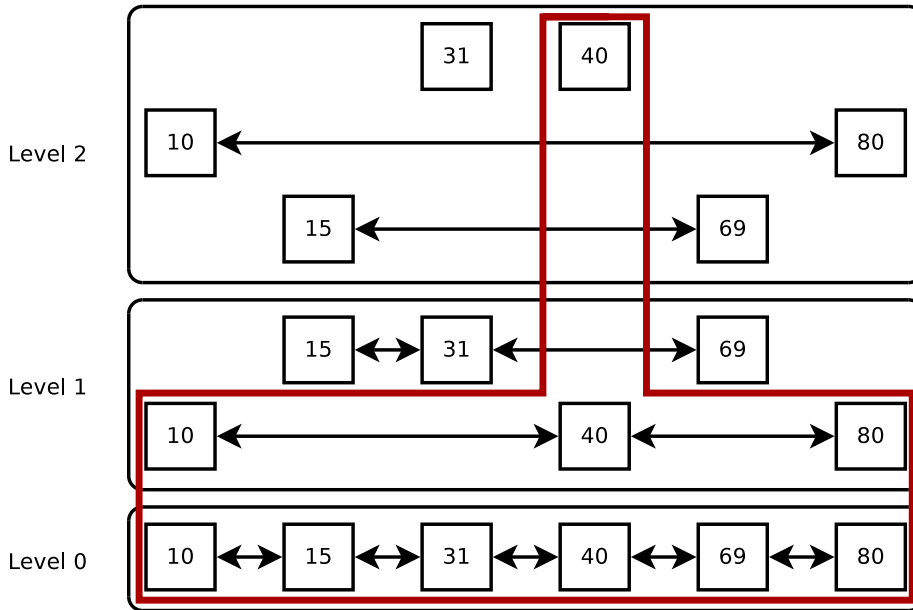


Figure 7: Skip graph with an highlighted skip list (red) adapted from [50] and, apparently, from [24].

2.4.1 Skip list, Skip graph, Skip tree

A P2P system is a large structure where pointers cross machine boundaries [4]. In the case of a Distributed Hash Table (DHT), the effectively resulting pointers are controlled by a hash function [4]. A DHT with hashed values does not prove to be a performant choice for supporting range queries, since only lookups on single nodes are possible [4, 17]. The introduction of the skip graph allows for nearest neighbour- and range-queries over ordered datasets [4, 17]. Skip graphs are increasingly popular amongst researchers [50].

A skip graph can be viewed as a distributed extension of skip lists, and therefore have a lot in common [17]. A skip list and a skip graph consist of levels containing increasingly sparse doubly linked lists [17], as shown in figure 7.

Each node has a membership vector consisting of an infinite amount of bits [17]. The bit at position o defines whether the node is participating in a doubly linked list at level o or not [17]. In the case of skip lists, there is only one list on each level, in the case of skip graphs, there are 2^o such lists on each level [17]. Only nodes residing on level o qualify as members for a list on level $o + 1$ [17]. In the case of a skip list, there is ultimately a single node at the highest level [17].

Aspnes et al. [4] try to maximize locality properties by grouping data accessed together on the same node. The main idea is to truncate the skip graph such that there is only a certain amount of intermachine pointers as opposed to Chord [4].

The skip tree from Alaei et al. [2] maximize locality properties by ordering the nodes by a tree. Range queries are supported with more efficiency than

alternatives with the equal amount of per-node state.

2.5 Overview of the redundancy research in Computer Science

As mentioned already, Reed and Solomon [35] have introduced a mathematical solution for noisy channels. This mathematically complex paper was explained by Plank by creating a guideline [31]. As a result, the PAR specifications and programs such as Parchive emerged. This insightful paper enlightened the method of Reed and Solomon. A correction of the former paper, the paper by Plank and Ding [32] referred to the advantage of RSEC in providing the freedom to choose *n closest* blocks. By doing this, Plank and Ding identified the advantages that are inherent in high availability regarding the dependence on locality. Geisel [15] is helpful for the understanding of Reed-Solomon erasure coding, because the mathematical basics are explained logically.

RSEC can also be used for hiding information. Xu and Bhalerao [52] distribute the pieces resulting from the coding procedure amongst cloud providers. Similar ideas are found in the work of Abu-Libdeh et al. [1] and Chen et al. [11]. The idea is that the secret original input is unknown to a single cloud provider if less pieces are stored on a certain cloud provider than are needed for successful reconstruction. The authors of [52] oversee that they also distribute data pieces, which are not secret. If security is preferred, a solution such as the related Shamir's Secret Sharing Scheme (SSSS) can be used [41].

Sathiamoorthy et al. [38] reduce the high repair cost that is associated with RSEC by introducing a method that supports locally repairable codes.

3 Method

In this section, the analysis method is presented first, because it is important to know that the redundancy method has to be profitable in relation to the underlying method (section 3.1). As a matter of fact, the cost model is introduced in its basic forms. The parameters of the basic formula of the cost model are specified during the whole section 3.

The introduction of the two underlying methods (section 3.2) is needed out of two reasons. Firstly, in order to know to what redundancy has to be applied to. Secondly, in order to know against what the new redundancy method has to be compared to. This allows to calculate the profitability. The description starts with the Core services layer of the method of Tanin, Harwood, Samet et al. [21, 49, 48, 47], and then continues with the lowest Connectivity layer containing the Chord lookup protocol, because the upper layer informs the requirements of the lower layer.

After is explained *to what exactly* within the underlying methods redundancy has been applied to, it has to be explained *what* is applied during the application of redundancy, i.e. RSEC (section 3.3). Finally, redundancy is effectively applied technically to the underlying methods (section 3.4). This is first done for the lowest layer, and then for the Core services layer as a part of a bottom-up approach.

3.1 Analysis method: Introduction to the cost model

A database management system stores facts in form of records. In order to process a read query successfully, certain records are needed. For the decision, whether a record is to be used or not, a database management system would have to go through the whole list of records if no use of any optimization is made.

A fundamental optimization is achieved by a tree-like index. It becomes possible to decide more quickly whether certain records are relevant to a query or not. With the help of the tree, a read query can decide what categories of records do not come into question at all, and that the remaining candidates are possibly important records. There exist different index structures. For example, it is possible to organize records along a line instead along a tree. This is not further relevant for the moment.

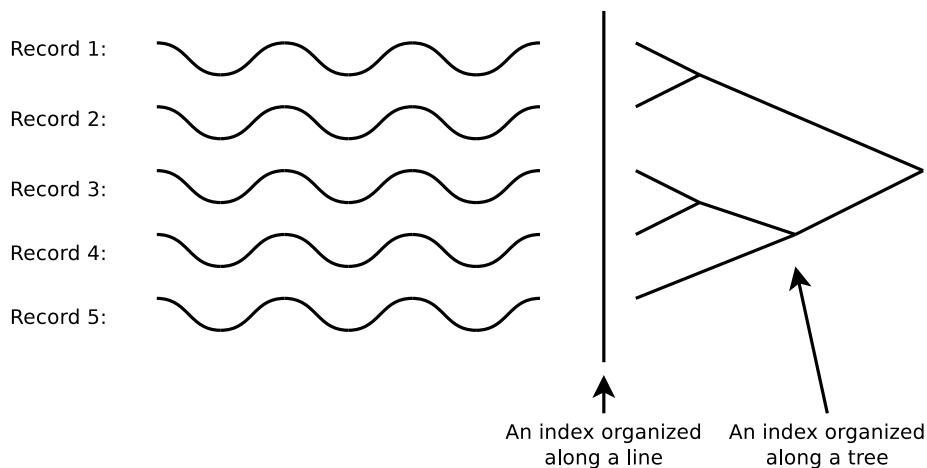


Figure 8: Some records in a database and two possible index approaches.

Assume that a certain read query needs a certain set of records, i.e. the content of each record must be made known to the read query for a successful completion of the read query. I call this set of records *group*. I enrich this group with redundancy. By doing this, I do not increase the group by adding new elements, but I replace existing records with a mix of existing fragments of records and newly calculated checksums. I replace the original records by distributing this mix evenly onto the original places of the participating records. As a result, less modified records are needed during a read query.

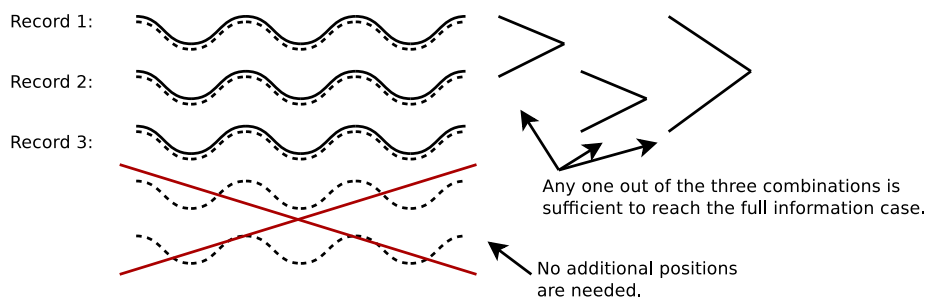


Figure 9: Applying redundancy (dashed line) to existing records. Note the similarity to figure 1.

In a distributed database management system, records are spread over nodes. Nodes are computers or servers, which are interconnected by a network infrastructure. The completion of a read query is only possible if all records of a group are found; but the records of such a group may be distributed over multiple nodes. For example, the read query fetching the needed records has to contact three separate nodes. With my redundancy method, the read query has to fetch less records within a group. This means that less nodes have to be contacted within the context of distributed systems.

Records can change their values. Dependent on the type of query² and the

²Queries include, but are not limited to, read queries and update queries. The formulation

system used, the number of affected records varies. Either only one record is changed, independently of the other records within a group, or multiple records at once, or possibly all records at once.

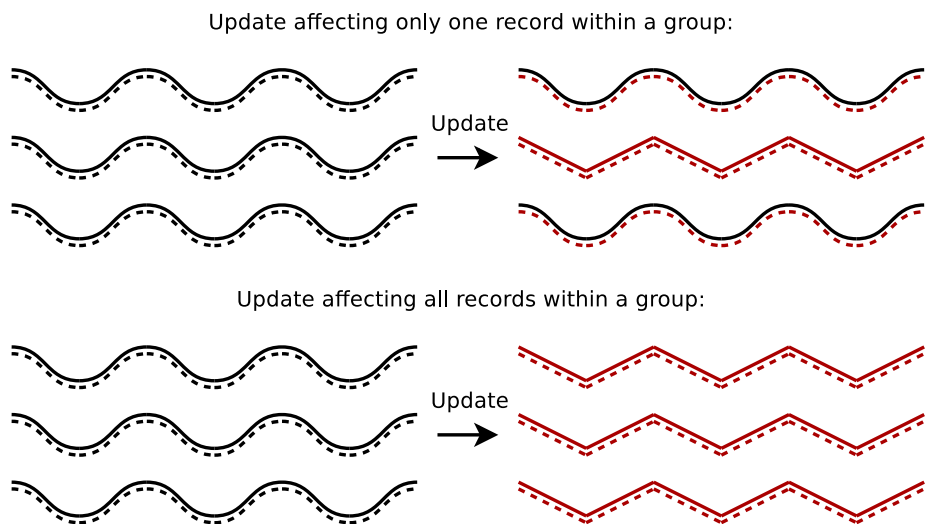


Figure 10: How updates are handled by the original method and my redundancy method: Updates in the original method may affect only one or all elements within a group. Updated records are symbolized as jagged lines, and changed values are in red.

My application of redundancy uses a mathematical method which has the characteristic property that updates are affecting the checksums of the whole group. Even if only one record gets updated, each participant of the group has to update its redundancy information, too. This is also the case if more than one record is changed in the original method or if all records are changed in the original method.

I do not only apply redundancy to the record-storing nodes belonging to a group. Sometimes, I increase the number of participating nodes by including otherwise unaffected nodes. This is done in order to increase the benefit of the current location of a requester. By doing this, it is achieved that a query may reach records more quickly, because the required information is distributed over more nodes. But this advantage of my extension comes at the cost of more contacted nodes during updates in relation to the original method. If the mix consisting of fragments of original data and checksums is distributed over 4 nodes instead of originally 3 nodes, then 4 nodes instead of originally 3 nodes need to be contacted during an update. This is always disadvantageous if two contacted nodes are not enough for a read query in order to reach the full information case. In other words: my extension for the benefit of less dependence on locality only makes sense if the application of my extension results in less contacted nodes during a read query compared to the unmodified original situation.

describing the optimal fragment allocation ideal conceived read queries and update queries as applications. Compare page 2 containing the definition.

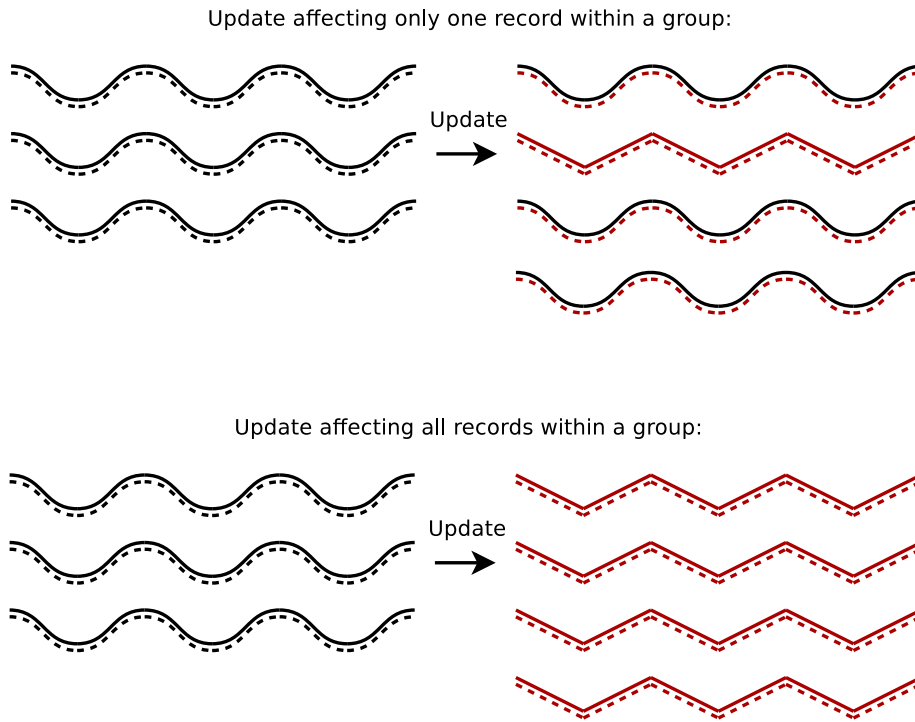


Figure 11: Advantages and disadvantages of an extension of a dataset: Storing information on more nodes than originally has advantages regarding locality and disadvantages regarding updates. If a read query only needs 3 nodes out of 4 nodes, the overall application of redundancy makes no sense because not only the same number of nodes are contacted during read queries, but additional nodes have to be contacted during update queries. Consequently, a read query must contact less nodes than a read query of the original method would.

It is possible to track the number of contacted nodes during a read query and the number of contacted nodes during an update query, both for the original method and for my redundancy method. The values for my redundancy method differ from the values for original method. If full redundancy is applied, a read query only has to contact one node within the group. If the amount of redundancy is chosen to be smaller, a fraction of the share of nodes needs to be contacted, or a large share of nodes needs to be contacted, depending on the amount of redundancy. During an update query, all nodes within a group need to be contacted. All nodes need to have the possibility to update their mix of fragments of records and checksums.

The number of contacted nodes during a read query in the original method is represented in the variable a . The number of contacted nodes during a read query in the context of my redundancy method is represented in the variable a' . Within the scope of an update query, the equivalents are b for the original method and b' for the redundancy method, respectively.

My redundancy method results in a profit during read queries, because less nodes than originally are contacted. This is not true, if the amount of redundancy is $r = 0$. If the update query of an original method contacts all nodes, then

the usage of my redundancy method does not result in loss. But if an update query of an original method only changes a single record, and subsequently only contacts a single node out of the participating nodes within a group, then my redundancy method is disadvantageous in comparison to the original method. This is because my redundancy method contacts more nodes than are contacted originally, because all nodes out of a group need to be contacted.

In order to answer the question whether an application of redundancy effectively reduces the amount of contacted nodes in relation to the original method, an analysis of the type of queries has to be done. If a group only experiences read queries, the application of redundancy is worthwhile because each read query has to contact less nodes than originally. If only update queries are made, the redundancy method is not worthwhile if only a single record is changed in the original method. Otherwise, an update query that contacts all nodes within a group does not result in a disadvantage in relation to the original method. It is conceivable that there is an original method that is capable of changing multiple records simultaneously, without having to update all records necessarily. Such a case is not treated in this thesis: Chord [44] only changes one record within a group, and the method of Tanin, Harwood and Samet et al. [21, 49, 48, 47] change all records within a group. As a consequence, the profit of my application of redundancy only has to be determined in relation to origin methods which only change one or all records at once within a group.

My redundancy method introduces the history h , which records the executed query types over the group. The history h lists the 50 last executed query types: read query (R) and update query (U). If the history h is filled completely because 50 elements are listed, the oldest entry is removed in order to make space for the new element. I treat a group as a totally new group whenever a record is added or removed from the existing group. As a consequence, the history h is not affected by the insertion or deletion of records.

$$\begin{aligned} h_0 &= [R] \\ h_1 &= [R, R'] \\ h_2 &= [R, R', R''] \\ h_3 &= [R', R'', U] \end{aligned}$$

Figure 12: History h through the time: The contents of a history h , which is limited to size 3, after three read queries (upper three entries) and after a subsequent update query (lowest entry). Note how the oldest element is automatically removed.

It is possible to determine, whether the application of redundancy has resulted in a profit, a zero-sum situation or in a negative profit, that is a loss. Illustrated on an example: assume a history of queries that led to $h = [R, R, U]$. Assume a group consisting of three nodes and full redundancy. As a consequence, each node stores all records of the group ($r = 1$). A read query in the context of my redundancy method only needs to contact one out of the three nodes within the group ($a' = 1$). The original method needs to contact all three

nodes in order to obtain all records ($a = 3$). Consequently, the profit of my redundancy method compared to the original method is $a - a' = 2$ during a read query. Assume that the original method only contacts one node during an update query, because only one record within the group is changed originally ($b = 1$). Because of the properties of my redundancy method, it is needed to inform all participating nodes about the change ($b' = 3$). As a matter of fact, the profit during an update query is negative, specifically $b - b' = 2$. Because history shows that the group experienced two read queries and one update query, the total profit is $2 + 2 - 2 = 2$. In fact, for the history $h = [R, R, U]$, the application of redundancy paid off. According to section 4.1, the profit over history h is quantified as $p(h)$. The formula

$$p(h) = |R| \cdot (a - a') + |U| \cdot (b - b')$$

only adds up the profits according to history h .

In effect, it is possible to establish whether an application of redundancy is worthwhile for a given history h . It is possible to determine how many update queries U in history h are tolerable, such that $p(h) \geq 0$. This depends on the amount of redundancy actually, because the amount of redundancy determines the profit of the read queries R . By setting $p(h) = 0$, it is possible to derive the exact ratio between update queries and read queries, with the result that an application of redundancy does not result in a loss. With $p(h) > 0$, the application of redundancy is worthwhile.



Figure 13: The maximal allowable share of updates for a given redundancy can be determined by solving for $p(h) = 0$. The share of updates is smaller if there is overall profit, or $p(h) > 0$ (assuming same r for all cases).

It is possible to find out where the border between being profitable and not profitable lies, always dependent on the chosen redundancy. By assuming the best case, that is full redundancy ($r = 1$), one can state for sure that an application of redundancy is *not* worthwhile if the share of update queries exceeds the allowable share of update queries. My redundancy method respects the maximally allowable share of update queries and it is my recommendation to abandon an application of redundancy during adverse conditions, i.e. if $p(h) < 0$ for $r = 1$.

As a matter of fact, full redundancy is a waste of storage space. Even though storage space is not represented in the cost model, I jump at the chance given

by the gap between the maximally allowable share of update queries and the actual share of updates, because my contribution aims to reduce the dependence on resources as much as possible.

During an update query, all nodes need to be contacted anyway. This is why I seize the opportunity to adapt the amount of redundancy in relation to the history h . On the one hand, in order to reduce the damage as much as possible by setting full redundancy ($r = 1$), if the share of update queries has crossed the maximally allowable share of updates. On the other hand, in order to make economical use of storage space, which is achieved by adapting the redundancy to the needs. This could be a reduction of the amount of redundancy, if necessary.

So, the amount of redundancy is adapted during an update query. It would be possible to adapt the amount of redundancy during a read query, but this would result in the fact that redundancy has reduced itself ad absurdity: the main idea of redundancy is exactly that the number of contacted nodes required to reach the full information case is reduced. In the case of an adaption of the amount of redundancy during a read query, there would be no reduction. As a consequence, I abstain from an adaption of the amount of redundancy during read queries. Consequently, the profit of a read query stays the same, until the amount of redundancy is adapted during an update query.

The cost model is simplified to the effect that old read queries are evaluated with a global redundancy value. It is possible to enhance the model, which would allow a more accurate determination of the maximal allowable share of update queries.

The profit of read queries is effectively dependent on the amount of redundancy, and the definition of a new amount of redundancy is recursively dependent on itself. A recursive definition requires the use of complex mathematics. The benefit of a more accurate adaption is out of proportion to what the simplified model provides. In the simplified model, I use a default value if the maximal allowable share of updates is not overstepped. This default amount of redundancy is the ratio of read queries in the history h , and not a fixed value. Otherwise, I employ full redundancy ($r = 1$), as already described.

My master thesis can determine the exact maximal allowable share of update queries in the history h for the simplified case of full redundancy. By doing this, I can guarantee a cap on expenses. The optimal amount of redundancy, which is dependent itself on the amount of redundancy, is not employed, because I make use of default values.

Not reaching the optimum, however, is not a serious shortcoming, because the redundancy is adapted within the winning zone. A zone which differentiates storage space utilization, which does not take part in the simplified cost model. The master thesis can satisfactorily answer whether an application of redundancy is worthwhile or not.

3.2 Original methods: Basis for the redundancy application

I applied the application of redundancy to two layers within the method of Tanin, Harwood, Samet et al. [21, 49, 48, 47]. In a first step I explain the two layers, in a second step I explain how I apply redundancy to those two layers.

3.2.1 The Core services layer

Tanin, Harwood, Samet et al. [21, 49, 48, 47] use the MX-CIF Quadtree, henceforth referred to as Quadtree. Regarding Quadtree, the discussion is within the context of the method by by Tanin, Harwood, Samet et al. [21, 49, 48, 47]; and without optimizations at first. Tanin, Harwood, Samet et al. [21, 49, 48, 47] call a node within a Quadtree “control point”. Every control point has four children. The first child A is responsible for the upper left quadrant of the square for which the parent control point is responsible. The child B is responsible for the upper right quadrant, the child C for the lower left quadrant, and the child D for the lower right quadrant.

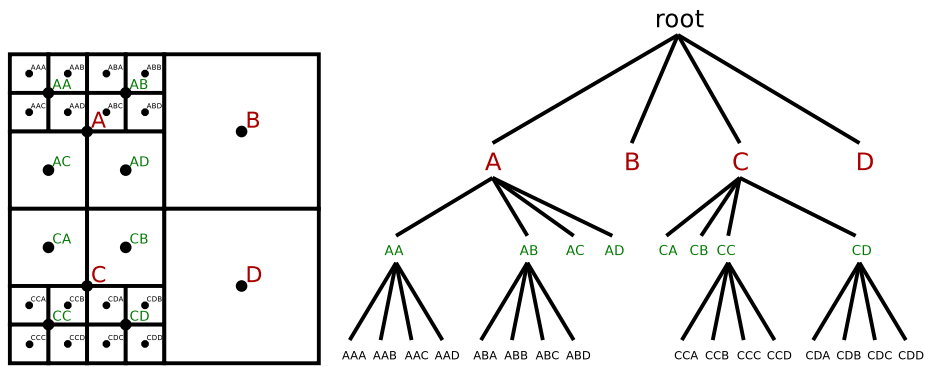


Figure 14: A MX-CIF Quadtree: The quadrants A, B, C and D on each square and the corresponding MX-CIF Quadtree. The control point is the centroid of a square.

Every two-dimensional spatial object has a minimum bounding rectangle (MBR). In three dimensions, the equivalent is the minimum bounding box (MBB). The three-dimensional tree dealing with the MBB is called Octree. In the following the explanations for the two-dimensional plane are analogous for the three-dimensional space. All operations on the Quadtree only use the MBR. During the insertion of a spatial object in the Quadtree new children are initialized if the MBR of the spatial object only covers one quadrant of an uninitialized control point. The initialization of a child is tracked by the parent by incrementing a counter, which itself is initialized to 0. The insertion is continued until the MBR overlaps more than one quadrant. As a consequence, each spatial object is stored at one control point.

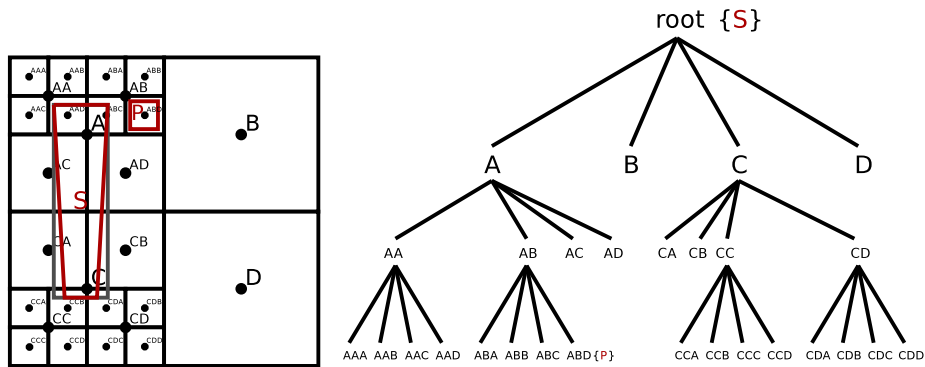


Figure 15: Some example objects in the Quadtree. The MBR of the spatial object S is shown in grey.

Consider figure 15. The responsible control point for the spatial object S is $root$, since S covers the quadrants A and C . The responsible control point for the spatial object P is ABD , since P covers multiple quadrants rooted at ABD . Assume that each control point is a node in a distributed system. Within the context of the method of Tanin, Harwood, Samet et al. [21, 49, 48, 47], responsible nodes store the whole spatial objects and its MBR in the metadata, whereby the MBR can be recalculated from the spatial object at any time. The node responsible for $root$ therefore has to store S and its MBR, and the node responsible for ABD has to store P and its MBR.

A lookup query running over the Quadtree knows the MBR of the wished object. Consequently, the lookup query knows the responsible control point where the spatial object has to be stored. If it is known that the spatial object was inserted, and if it is possible to directly contact the node responsible for the identified control point, then the lookup query can contact the responsible node instantly. Otherwise, the Quadtree has to be traversed because the child may be uninitialized. In this case, the Quadtree is traversed by following pointers to children until the wished control point containing the geometry of the spatial object is reached if possible. The traversal always begins at the globally known entry point $root$.

The original method of Tanin, Harwood, Samet et al. [21, 49, 48, 47] does not mention update queries. Update queries can be simulated by the sequence of a deletion of a spatial object, followed by the insertion of a new, modified object. As a matter of fact, it is possible to modify the original method by introducing the possibility of an update query. The update query is very similar to a lookup query. Instead of the download of the spatial object, the node issuing the update query pushes the new spatial object, or only the difference, to the node which is storing the spatial object.

3.2.2 The Connectivity layer

Tanin, Harwood, Samet et al. [21, 49, 48, 47] apply the hierarchical Quadtree to a line, this is a Distributed Hash Table (DHT). Chord [44] is used as the lookup protocol over the distributed hash table.

A Distributed Hash table is a number line with a limited number of possible

identifiers or keys, the identifier space. The parameter t specifies the size of the DHT. A DHT with $t = 8$ has space for $2^8 = 256$ keys, the keys 0 to $2^8 - 1$. $t = 8$ Bits are sufficient for the declaration of a key position.

A DHT is subject to modular arithmetic. This means that all calculations made over keys are subject to the modulo operation. As a matter of fact, no operation made over keys can result outside the interval $[0, 2^t - 1]$. This is why the line can be represented as circle. Similar to a clock, if the end is overstepped (rightmost part of the line), counting continues at the beginning (leftmost part of the line).

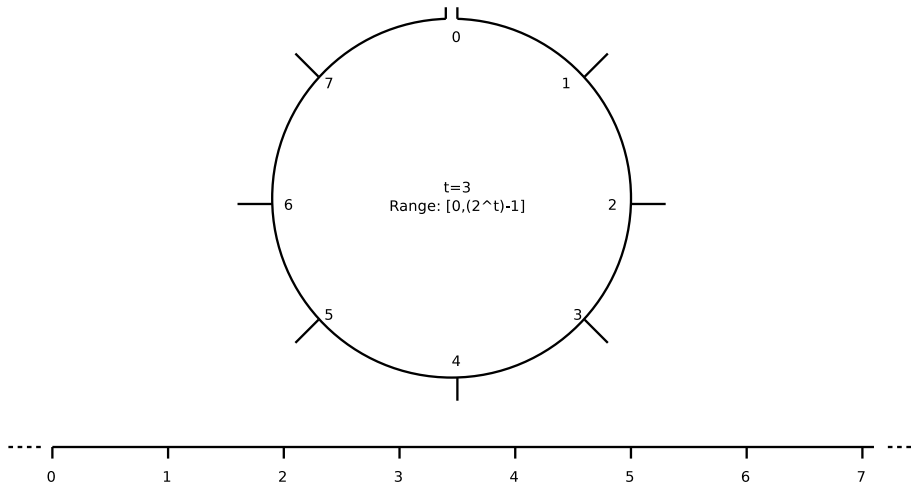


Figure 16: The DHT of Chord.

In the context of the method of Tanin, Harwood, Samet et al, a DHT can include both control points and nodes. Control points are included as logical elements. The nodes of a P2P-network, i.e. computers, are active participants in the DHT. Both the control points and the computers need to be mapped to a key. This is done with a hash function. A Hash function has two relevant properties: for one thing, the output of a hash function has always the same size, irrespective of the size of the input. For another thing, a hash function spreads the output evenly and with the same probability, independent of the similarity of the input. This property is named avalanche effect and results in the fact that a similar input leads to a completely different output with high probability.

The control points get mapped to the DHT as follows ($t = 160$):
 $\text{key} = H(\text{ABD}) = 164791f3ecd0bcf3387a67a7f1d43f2baf1fde19$. The nodes are also mapped on the DHT, i.e. they receive a key. The key is the hash of the IP address of the node.



Figure 17: Chord Example: A DHT with three registered nodes κ , λ and μ and seven logical elements, the control points ABD, root, BCA, CDA, BBD, CD and CAD. Note that κ is responsible for the control point BCA.

Every node is responsible for a range in the identifier space. The range reaches from the key of the preceding node +1 to the own key [53]. Consequently, it is clear what node is responsible for what control point. In figure 17, κ is responsible for ABD, **root** and BCA, and λ for CDA and BBD. Taking responsibility means that κ has to store all spatial objects that are connected with control point ABD, for example. Furthermore, κ has to process all queries pertaining to the quadrant in question. It is possible that a node is responsible for multiple control points. This is to be expected if there are less nodes than control points. This is most certainly the case for a central server or a smaller cluster of computers. On the contrary, it is possible that there are so many nodes, that some nodes are effectively not responsible for any control point. This is imaginable for P2P systems, see section 2. As a stress test for my application of redundancy, I assume the worst case throughout this thesis that any node is maximally responsible for one control point.

Every node in a DHT has to maintain a pointer to the IP address of its predecessor and a pointer to the IP address of its successor. By doing this, it becomes possible to find the responsible node for a key, for example the key of the control point ABD, which is $H(ABD)=164791f3ecd0bcf3387a67a7f1d43f2baf1fde19$. For the moment, the optimization called finger table is ignored. A message containing the IP address of the originator and target key $H(ABD)$, is handed over to the successor of each participating node, until the identifier of the currently processing node is larger than, or equal the key $H(ABD)$. The currently processing node (κ in figure 17) is the responsible node for the key $H(ABD)$. This responsible node (κ) contacts the originator of the node lookup.

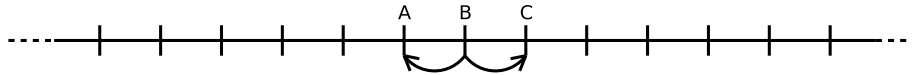


Figure 18: Chord neighbours: Node B maintains a pointer to its preceding node A and a pointer to its successor C. By interlinking neighbouring nodes, line traversal becomes possible. Only nodes are shown, logical control points are not shown.

If it is known that a certain spatial object was inserted, a lookup query can now contact the responsible node directly with the help of Chord. Otherwise, the lookup query has to traverse the Quadtree in the hope to find the control point in question initialized, starting with the control point **root**.

The method of Tanin, Harwood, Samet et al. [21, 49, 48, 47] has applied the Quadtree successfully to the DHT. As a matter of fact, the traversal in the Quadtree corresponds to repeated node lookups in the DHT. Consequently, it becomes possible to perform range queries with the help of the node lookup procedure in Chord, which constitutes the main idea of method of Tanin, Harwood, Samet et al. [21, 49, 48, 47].

A range query is started by a user in the application layer, which is one out of three layers in the method of Tanin, Harwood, Samet et al. [21, 49, 48, 47]. The user draws a query rectangle. The user expects to discover all spatial objects whose MBR intersect the query rectangle. The application layer calls the service in the interface between the Application layer and the Core services layer. The Core services layer then starts the range query.

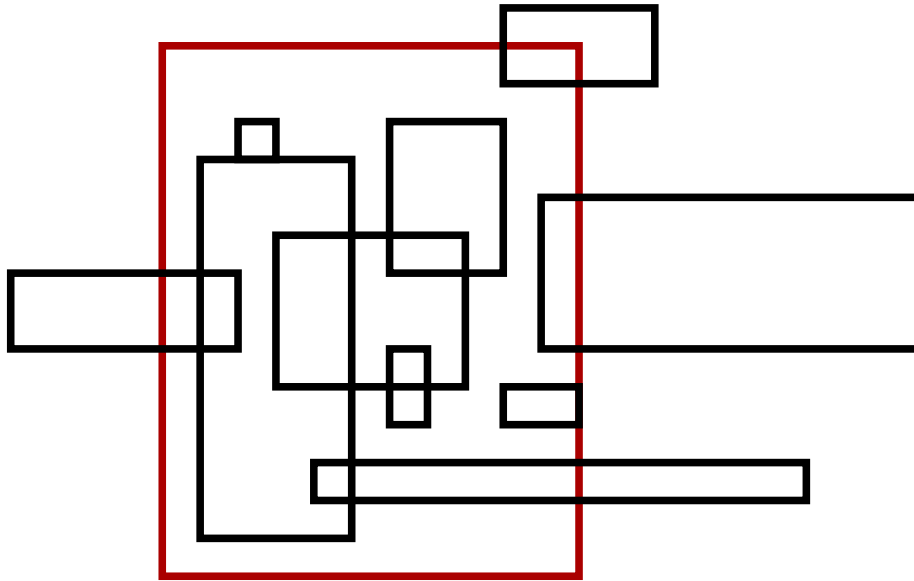


Figure 19: A range query: All spatial objects (black) that intersect the query window (red) have to be returned.

The range query message containing the query rectangle and the IP address of the requester is sent to the root control point. The root control point compares the query rectangle to the MBRs of the spatial objects that the control point is responsible for. Every intersecting spatial object is uploaded to the IP address of the requester, this is the computer from which the range query originates. Afterwards, the root node compares the MBR with its four quadrants. If the search rectangle intersects a quadrant, and if the quadrant is initialized since the counter is > 0 , the root node relays the range query message to the node which is responsible for the quadrant of the child control point. Each child proceeds exactly as described for the root control point. That means that each control point receiving a range query has to compare the query rectangle to the MBRs of the spatial objects that the control point in question is responsible for. Afterwards, each control point checks whether the query rectangle intersects an initialized quadrant and relays the range query message in such a case.

It is possible that a control point stores spatial objects whose MBRs do not intersect with the query rectangle. These spatial objects are not within the range and are not sent.

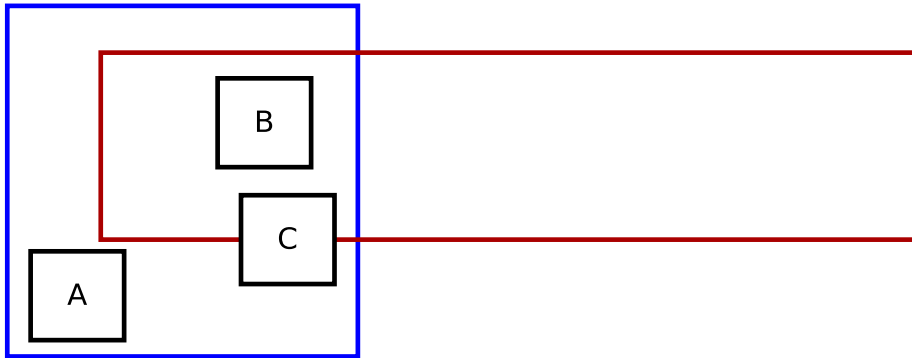


Figure 20: Query rectangle and its objects: A node which is responsible for a certain control point, whose area is depicted in blue, only has to return spatial objects that intersect the query rectangle, depicted in red. For this figure, the node has to return spatial objects B and C. A is not within the range and is not sent.

Whenever a parent control point has to contact one of its child control points, the method of Tanin, Harwood, Samet et al. [21, 49, 48, 47] have to locate the responsible node for the child control point in the DHT.

3.2.3 Optimizations in existing methods

As a matter of fact, a range query in the described system would be possible, but this endeavor is inefficient without any optimization. Each time a responsible node for a control point has to be looked up in unoptimized Chord, it is statistically expected that the query needs to contact half of all participants ($\frac{N}{2}$) due to the arithmetic mean (see figure 21). Additionally, the node responsible for the root control point has an unfair share of the total load in the system, because every range search has to start at the root. As a matter of fact, applying hierarchical trees in a distributed system constitutes a challenge to the requirement that load should be balanced. Additionally, the node responsible for the root node may become a central point of failure, since no tree traversal may be started without a root node [9].

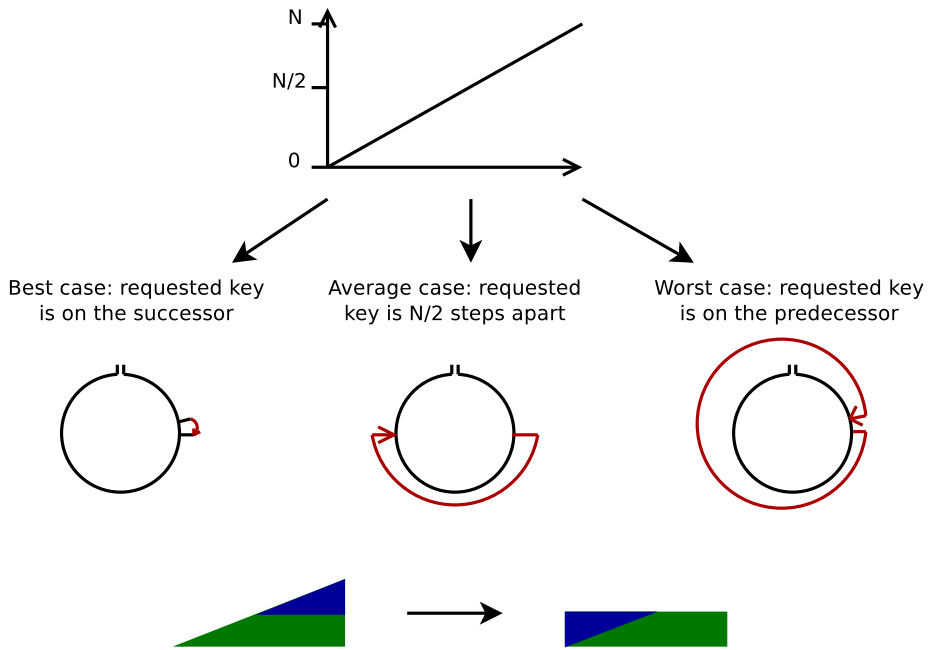


Figure 21: Chord lookup without optimization: Assume that all possible starts and ends of searches are equally probable. As a consequence, all possible distances listed in the array $\{1, 2, 3, \dots, N\}$ are equally possible. The sum of this array divided by the size of the array results in $\frac{N}{2}$.

Both Tanin, Harwood, Samet et al. [21, 49, 48, 47] and Chord each make use of an optimization. The first optimization concerns the variant of the Quadtree used by the method of Tanin, Harwood, Samet et al. [21, 49, 48, 47]. The second optimization affects Chord.

Optimization done by Tanin, Harwood, Samet et al.

Tanin, Harwood, Samet et al. [21, 49, 48, 47] introduce the globally known and immutable parameters f_{min} and f_{max} .

f_{max} defines the maximal height of the Quadtree. It becomes impossible to store any spatial object at a level l deeper than f_{max} , even though a spatial object only intersects one quadrant.

f_{min} defines the minimal allowable level of the whole Quadtree. Instead of a root control point there are $4^{f_{min}}$ control points. The equivalent for the Octree would be $8^{f_{min}}$ control points. Consequently, the Core services layer has to calculate the starting control points for a given query rectangle at the start of a range query. This is achieved by the method called **Subdivide**, as described by Tanin, Harwood, Samet et al. [21, 49, 48, 47]. The nodes responsible for the control points at level f_{min} proceed normally, as it was described. During insertion, however, spatial objects that would be stored at a level $l < f_{min}$ need special consideration. In fact, such a spatial object is subdivided into parts. The spatial object is no longer associated with its minimum enclosing Quadtree quadrant [37]. Actually, the **Subdivide** method is called and the intersecting

control points at level f_{min} are determined. This list of control points at level f_{min} for a certain spatial object is called G_{min} throughout this thesis. These control points can then proceed normally, even though the MBR is not wholly contained within the area for which the control point is responsible for.

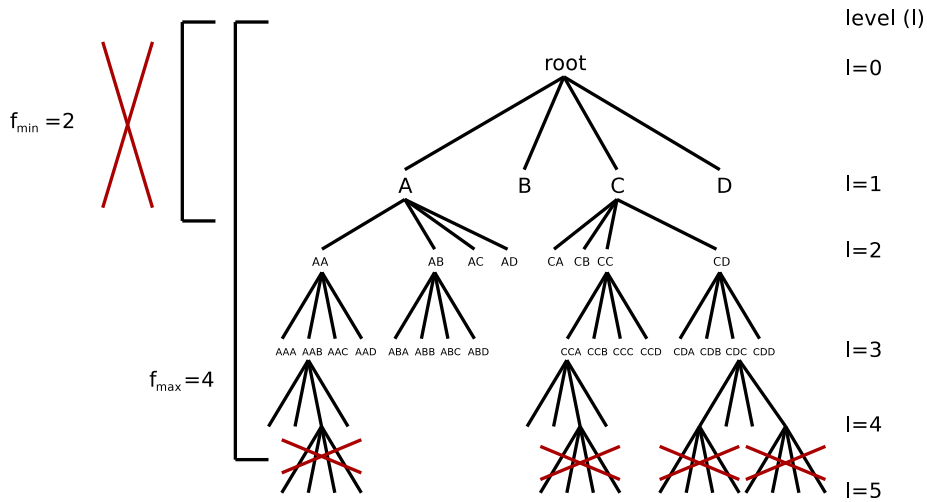


Figure 22: The effects of the parameters f_{min} and f_{max} in the MX-CIF Quadtree. No spatial object is stored on level $l > f_{max}$. Any object that would be stored on level $l < f_{min}$ has to be subdivided. The tree has now multiple roots, i.e. AA, AB, AC, ...

As a consequence, a spatial object, that had to be subdivided, is effectively stored on multiple control points. Applied to the telephone book illustration, the spatial object is a complete telephone book, and the original method replicates the telephone book to each participant.

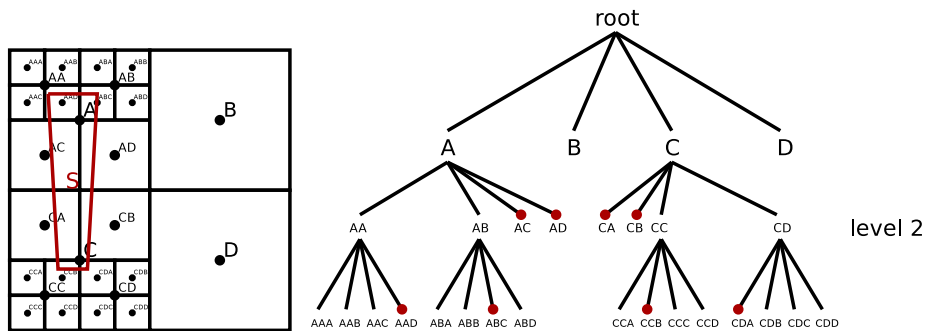


Figure 23: An example of a divided spatial object in the Quadtree: The spatial object S to be inserted has to be subdivided to $G_{min} = \{AA, AB, AC, AD, CA, CB, CC, CD\}$ ($f_{min} = 2$). Afterwards, it is processed in the tree. The spatial object is finally replicated to the control points $G_{eff} = \{AC, AD, CA, CB, AAD, ABC, CCB, CDA\}$, since the parts can not fall down further than level 2 (AC, AD, CA, CB) or further than level 3 (AAD, ABC, CCB, CDA). G_{eff} is marked in the Quadtree as red points. Each node stores the whole spatial object and the MBR. Adapted from Tanin, Harwood and Samet [47].

Optimization done by Chord

Every node has to maintain a finger table with t entries, which is comparable to a private telephone contact list. A so-called finger is a pointer to the IP address of another node in the DHT which is responsible for a chosen key. A finger is comparable to an entry in the private telephone contact list: the name and address of the person is the key and the telephone number is the IP address. The fingers to succeeding nodes are chosen with an exponentially increasing distance, or more formally, the entry i contains the link to the successor node responsible for position $(ownPosition + 2^{(i-1)}) \bmod 2^t$ [44]. As a consequence, each node maintains a large number of pointers to succeeding IP addresses that are near, and less and less pointers to succeeding nodes far away.

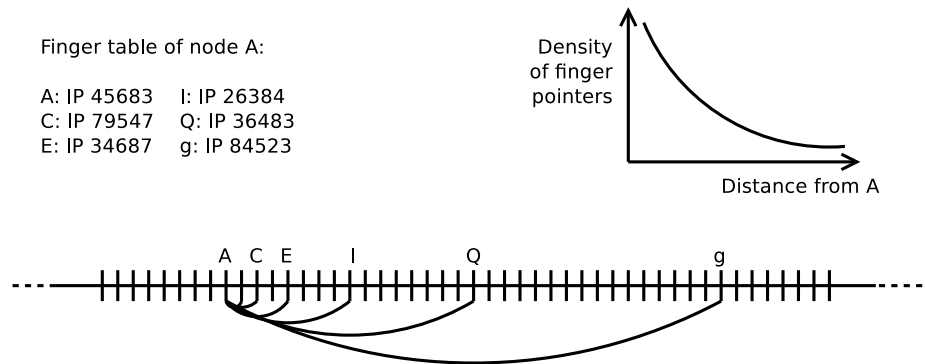


Figure 24: A finger table in Chord, the exponential distribution in dependence on the distance and the visual representation of the pointers.

Any node can start the Chord node lookup. During a lookup of a responsible

node for a given key, finger tables are used if possible. The node which initiates the node lookup can relay the search coarsely into the target zone. The contacted node can relay more accurately to the target zone, because the target key is nearer to the contacted node than to the initiating node. The contacted node maintains more pointers around the target zone than the initiating node since the contact node maintains more pointers for nearer succeeding nodes. The search message is relayed. After each relay, the target key is approached more accurately. At some point, the responsible node for the target key is found. The paper describing the Chord node lookup protocol [44] shows that the effort for a node lookup is $\log(N)$; N is the number of participants in the DHT. Assuming 1'000'000 participants, the number of contacted nodes is reduced from 500'000 on average to approximately 20. With the help of my piggybacking method, this value can be underbided.

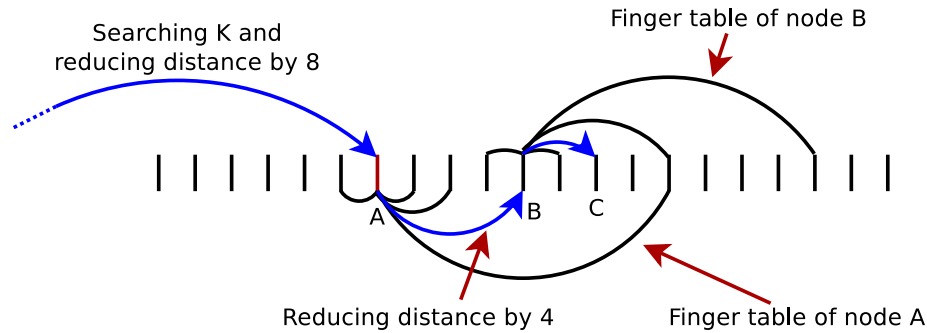


Figure 25: Search using finger tables in Chord: Search (blue) arriving at the target C, that is responsible for K . The arcs to the right of the nodes take part in the finger table.

3.2.4 Optimization through applying redundancy

This master thesis identifies groups of commonly accessed elements and enriches the group with redundancy.

Optimizing the method of Tanin, Harwood, Samet et al.

The method of Tanin, Harwood, Samet et al. [21, 49, 48, 47] replicates the spatial object to every responsible node for a control point listed in G_{eff} . I identify these nodes as a group and replace the spatial object with a mix of fragments of the serialized spatial object and checksums. Thereby, it becomes possible to contact only a subset of nodes that are responsible for a control point listed in G_{eff} . The fragments and checksums from the subset of nodes are sufficient now for the successful reconstruction of the spatial object. Using the telephone book illustration: I enrich the telephone book pages with redundancy and distribute the pages evenly amongst the participants.

It is possible to object, that because the original method of Tanin, Harwood, Samet et al. [21, 49, 48, 47] copies the spatial object to every node which is responsible for a control point, only one single node is enough for the complete download of the spatial object. Because this is also the case with my redundancy method for $r = 1$, one could think that nothing is gained.

Setting $r = 1$ is actually a waste of storage space. My redundancy method additionally provides the possibility to use a lower amount of redundancy, which is not possible in the original method. Consequently, my first strength of my redundancy method is that less storage space is wasted.

Parallelism is an advantage for the method of Tanin, Harwood and Samet et al. [21, 49, 48, 47], because spatial objects from different search instances are returned from multiple nodes. But there is the disadvantage that the same spatial object may be returned multiple times. In a variant used in the method of Tanin, Harwood and Samet [47], this disadvantage is alleviated by only storing the spatial object at the “owner node”, and all nodes in G_{eff} point to the owner node. For my method, however, every additional source for a spatial object is actually an advantage.

With my redundancy method, the requester receives fragments of responsible nodes. Consequently, all parallelly downloaded informations are usable and new knowledge. Additionally, it becomes possible to stop downloading, as soon as enough fragments (telephone book pages) are gathered. Therefore, this is the second strength of my redundancy method: it becomes possible to download from multiple sources, and a complete download is not needed. This reduces the overall download time and prevents network congestion. Compared to the telephone book illustration: Because less complete telephone books have to be sent around, since only pages are available, the postal office has less workload.

The third strength: with the help of my redundancy method, the fastest nodes can be used. It seems evident that speed of light determines that the fastest nodes are the nearest ones. But if there are complex reasons why the most local nodes are not the fastest to respond, those reasons are not relevant to my redundancy method, because my redundancy method always profits from the optimum. As it happens, only the most local nodes on a cost surface are needed, and the reasons do not have to be quantified, because it is affordable to be blind to those reasons, even though it can be assumed that the nodes listed in G_{eff} residing at the deepest level in the Quadtree are likely the last ones to respond, because every traversal of a level begins with a Chord node lookup.

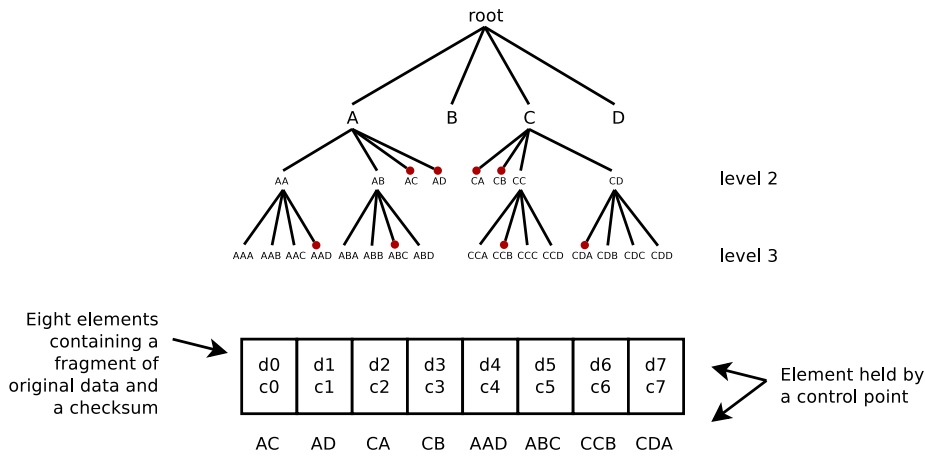


Figure 26: Enriching a spatial object in the Quadtree: Instead of replicating the spatial object, the serialization of the spatial object is divided into data fragments $\{d_0, d_1, \dots\}$ and checksums are calculated ($\{c_0, c_1, \dots\}$). Each control point receives a data “device” and a checksum “device”. The nodes $\{AC, AD, CA, CB\}$ at level 2 are most likely the fastest to respond, since these nodes do not suffer from the additional delay of a further Chord node lookup, as compared to the nodes at level 3. Four downloads from four different nodes are enough for the successful reconstruction of the original spatial object. Any other combination is also tolerable.

The number of contacted nodes during an update is the same for both the original method and my redundancy method ($b = b' = |G_{eff}|$). For both cases, the changes must be made on all participating nodes. During read queries and $r = 1$, I contact as much nodes as the original method ($a = a' = 1$), this is the fastest node that responds. If a lower amount of redundancy is used during read queries, more than one node need to download their mix of fragments and checksums to the issuing node. Thereby, I support a lower redundancy than $r = 1$, which is not supported by the original method.

Optimizing the Chord node lookup

A search in Chord first makes big steps, and the steps become smaller as the search is nearing the target key.

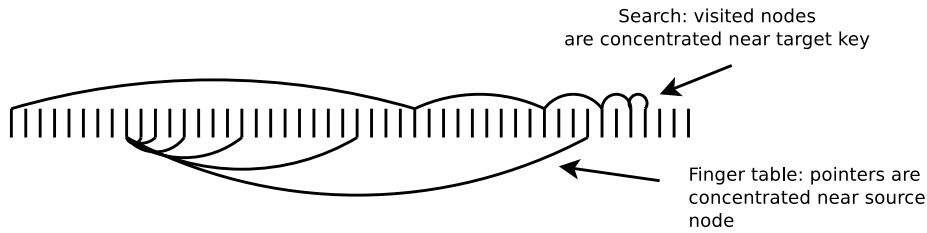


Figure 27: How a search compares to finger tables. The visited nodes during a search are concentrated near the target key. The pointers of each node are concentrated near the node in question.

Because the most steps are made just in front of the target key, I have introduced a moving window just in front of the target. See figure 33 and figure 28 for an example of a moving window. This moving window is rooted at the target node and, as a consequence, contains the target node itself. The moving window contains the IP addresses of all participants. Assume full redundancy ($r = 1$). As soon as the search encounters a moving window of which the target is a participant of, it becomes possible to jump directly to the target node, because all IP addresses within the moving window become instantly known ($a' = 1$). If the amount of redundancy is smaller, more than one step within the area of the moving window is needed (z steps). But the total amount of steps z is still lower than without application of redundancy. In this case, $a' = z$.

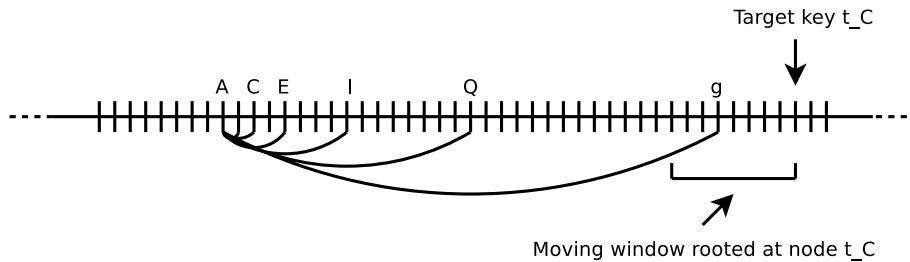


Figure 28: Introducing a moving window: This particular moving window is rooted at node t_C . Similar to finger tables, *each* node has a moving window rooted at the node in question.

Because the contacts of the node lookups are distributed randomly in front of the target, it is impossible to determine a certain set of contacted elements. The size of the moving window is therefore larger than the the amount of normally contacted nodes during an average search. If the size of a moving window is s , a search normally contacts $a = \log s$ elements. This means that I made a scope expansion ($\log s < s$). I enlarged the group of commonly accessed elements. This and the fact that only a single element in the group is changed during an update of an IP address results in the fact that an update is cheaper in the original method ($b = 1$) in comparison to my redundancy method ($b' = s$). In the original method, only the node changing its IP address is affected³. With my

³Actually, its predecessor and successor, too. But it is assumed that this is a problem of the Chord internals and does not influence the cost model.

piggybacking method, all nodes within the moving window need to be contacted. Again, if a node departs or a new node is introduced, completely new groups or moving windows are defined. Inserts or deletions of elements within a group are not updates.

Solving for $p(h) = 0$ results in the maximal allowable share of updates is $\frac{\lceil \log(s) \rceil - z + 1}{|h|}$, or more exactly, due to the parameter $\alpha_{Chord} = 0.5$ derived by the authors of Chord [44], $\frac{\lceil \alpha_{Chord} \log(s) \rceil - z + 1}{|h|}$.

Summarizing, an application of redundancy is only worthwhile if the maximally allowable share of updates within a history h is not overstepped.

If the maximal allowable share of updates is not overstepped, this thesis can guarantee that redundancy is of help regarding the number of contacted nodes. Consequently, it is possible to reduce the dependence of locality. This is valid for both the normal spatial space and the virtual space.

3.3 Reed-Solomon erasure coding

Reed-Solomon erasure coding (RSEC) has important properties that make an application to files or data structures worthwhile. Mainly, RSEC introduces the freedom to choose randomly a minimal subset out of a set. Altering the dependence on locality is only possible thanks to this favourable property of erasure coding, especially RSEC. It is important to explain the fundamentals in detail since there are interesting properties and limits inherent in RSEC. Any method applying RSEC has to cope with these properties.

Let there be blocks of the size k bytes or $k \cdot 8$ bits. A byte has a value ranging from 0 to 255 inclusive [52]. There are n data devices given, each of size k [31]. Furthermore, there are m checksum devices, also each of blocksize k [31]. Reed-Solomon coding calculates the value of the checksum devices in a way that if m of the data devices or checksum devices may be erased, the contents from the failed devices can be reconstructed from the non-failed devices [31]. The calculations are made over words of size w bits [31]. Unnecessary padding of words, that means extension to the size of words, can be avoided by choosing w as multiples of 8, as word boundaries subsequently fall directly on byte boundaries [31]. Each device is split into a sequence of words. For simplicity, assume that each device holds exactly one word. Actually, for I input data bytes and J output checksum bytes, the blocksize or the size of a device k can be chosen arbitrarily, as long as following two conditions hold [31]:

$$\left\lceil \frac{I}{k} \right\rceil + \left\lceil \frac{J}{k} \right\rceil < 2^w$$

$$n + m < 2^w$$

If $\lceil \frac{I}{k} \rceil \neq \frac{I}{k}$ or $\lceil \frac{J}{k} \rceil \neq \frac{J}{k}$, the input sequence of bytes does not align with the block size, and the space left needs to be filled with non-important padding bits, as depicted in figure 29. This is achieved by converting the binary input to base 2, and adding a binary 1 to the end of the payload, followed by zeroes until the block is filled up. In the special case of a perfect fit, it seems that no padding is required. But the recipient can not distinguish between data bits and padding bits in this case. It is impossible to interpret the end correctly. The

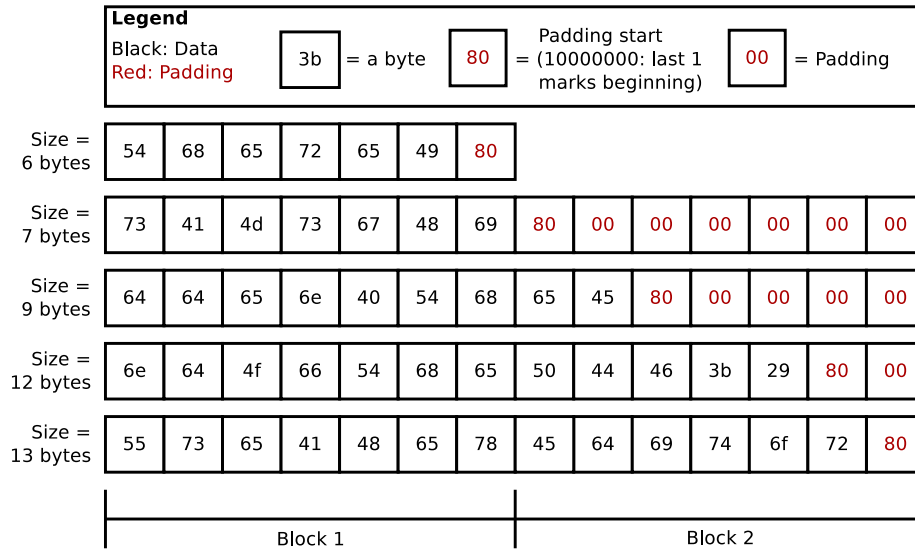


Figure 29: Padding in order to align with block borders.

producer has to append a whole block starting with the bit 1 and containing nothing else than zeroes after that starting bit. This worst case of having to add a whole block is impossible to circumvent since all available information space is used up for the binary object. Adding at least one bit reserved for information about padding at the end of the binary object is the only way to match block sizes without further external information. I decided that padding is normally required even in the case of $\lceil \frac{I}{k} \rceil = \frac{I}{k}$ or $\lceil \frac{J}{k} \rceil = \frac{J}{k}$, since external information would be needed at higher levels otherwise. Choosing k and w wisely reduces the impact of the worst case of a whole padding block, and such a padding block at the end of the last data device may be the only overhead. Otherwise, every data device would additionally have to transport at least one byte containing padding instructions.

3.3.1 Basic idea

Suppose following system of equations:

$$4V - 5W = 8 \tag{1}$$

$$-2V + 6W = 10 \tag{2}$$

There are two unknown variables, but with two equations, the system of equations is solvable. The solution to this system is $V = 7$, $W = 4$ and is calculated with Gaussian Elimination. It is possible to add another equation to the system of equations:

$$3V + 5W = 41 \tag{3}$$

Since $3 \cdot 7 + 5 \cdot 4 = 41$, this equation is a valid equation taking part in the presented solvable system of equations. Equation 3 over-defines the system of equations. Addition of an infinite number of distinct equations to this system

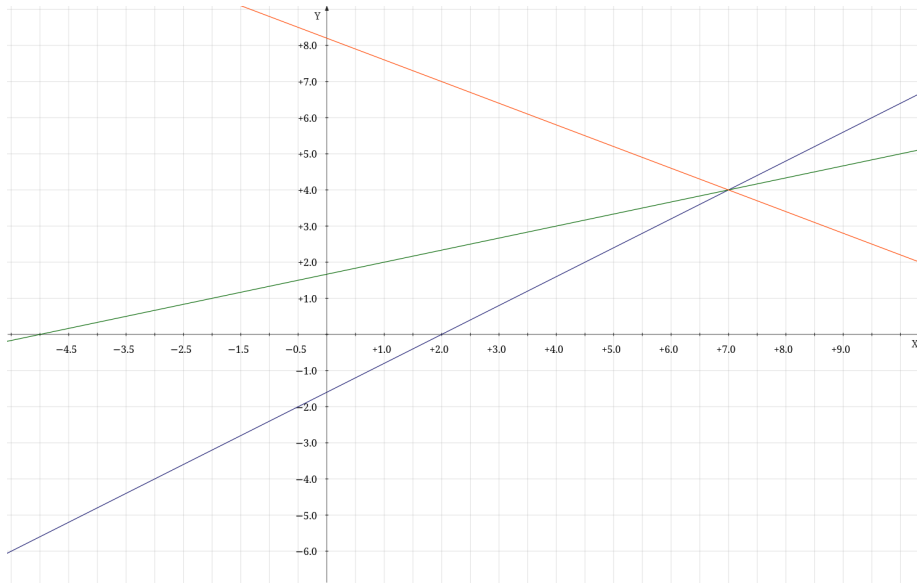


Figure 30: As long as $W = f(V = 7) = 4$ holds, *any* two lines are sufficient to derive the intersection point $(V, W) = (7, 4)$. Equation 3 (red) is optional unless one of the other lines is unknown. If V and W are already known, calculating the intersection is superfluous.

of equations is possible. As long as the added equations hold for $V = 7$ and $W = 4$, the system of equations remains uniquely solvable. Calculation of V and W is always possible as long as at least *any* two equations out of the solvable system of equations are known, as shown in figure 30.

The idea is to treat the two variables V and W as important words of data devices and specifying m more equations than needed. The coefficients of the two variables are generated deterministically depending on the number of input variables n and the number of equations $n + m$ (the information dispersal matrix). The right part of the equations represents the calculated checksum word.

Since the coefficients of V and W are generated deterministically, the position of the equation needs to be known in order to derive the coefficients successfully for a given position. At the core, Reed-Solomon Coding calculates checksum words by multiplying the coefficients with the data words, and summing up these products. During reconstruction of the original data words, such systems of equations are solved. This is done by Gaussian elimination on a square matrix (matrix inversion), as outlined in section 3.3.4.

3.3.2 Galois field algebra

The arithmetic over the words is done over finite fields [31]. A field $GF(2^w)$ is a closed set of 2^w elements or symbols on which addition, subtraction, multiplication and division is possible without leaving the set [31, 15]. The elements are actually binary coefficients of binary polynomials [31]. The elements are enumerated with an irreducible generator polynomial [15]. A polynomial of degree

Generated element	Power representation	Polynomial element	Binary element	Decimal element
0	0	0	0000	0
1	1	1	0001	1
α^1	X^1	x	0010	2
α^2	X^2	x^2	0100	4
α^3	X^3	x^3	1000	8
α^4	X^4	$x + 1$	0011	3
α^5	X^5	$x^2 + x$	0110	6
α^6	X^6	$x^3 + x^2$	1100	12
α^7	X^7	$x^3 + x + 1$	1011	11
α^8	X^8	$x^2 + 1$	0101	5
α^9	X^9	$x^3 + x$	1010	10
α^{10}	X^{10}	$x^2 + x + 1$	0111	7
α^{11}	X^{11}	$x^3 + x^2 + x$	1110	14
α^{12}	X^{12}	$x^3 + x^2 + x + 1$	1111	15
α^{13}	X^{13}	$x^3 + x^2 + 1$	1101	13
α^{14}	X^{14}	$x^3 + 1$	1001	9
α^{15}	$X^{15} = X^0 = 1$	1	0001	1

Table 4: Enumeration of the elements of $\text{GF}(2^4)$. Irreducible polynomial: $x^4 + x + 1$ [31, 15].

w can not be factored and is irreducible if it divides $x^{2^w-1} + 1$, but not $x^H + 1$ for any $0 < H < 2^w - 1$ [15]. Such a polynomial is comparable to a prime number which is also not divisible by anything smaller than itself; not counting 1 since $H > 0$. For $w = 4$, such a polynomial is $x^4 + x + 1$, for $w = 16$ it is $x^{16} + x^{12} + x^3 + x + 1$ [31, 15]. Each operation is calculated modulo the generator polynomial if the result has a degree $\geq w$ [31]. Enumeration of the field as in table 4 is therefore achieved by multiplying the previous element with x and calculating the result modulo the generator polynomial, with the polynomial elements 0, 1 and x as the first elements [31].

The addition of the two polynomials $x^3 + x + 1$ (1011) and $x^2 + x + 1$ (0111) modulo $x^4 + x + 1$ results in $x^3 + x^2$ (1100). The subtraction of those two elements has the same result since we are calculating with binary coefficients. Addition is actually the same as subtraction. Addition can be effectively made by calculating the bitwise exclusive-or on the binary elements [31, 52].

Multiplication or division of two polynomials is analogous if the method “polynomial multiplication or division” by Geisel [15] is applied. There are also other methods. For example, using the “exponent mod n multiplication method” for $w = 4$:

$$\alpha^5 \alpha^{14} = \alpha^{19} = \alpha^{19 \bmod 2^4-1} = \alpha^4$$

Division is analogous by multiplying with inverse symbols:

$$\alpha^5 / \alpha^{14} = \alpha^5 \alpha^{-14} = \alpha^{-9} = \alpha^{-9 \bmod 2^4-1} = \alpha^6$$

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
gflog [i]	-	0	1	4	2	8	5	10	3	14	9	7	6	13	11	12
gfilog [i]	1	2	4	8	3	6	12	11	5	10	7	14	15	13	9	-

Table 5: Mapping the binary element to the logarithm (**gflog**) and the power Q to the binary element (**gfilog**) in the Galois field for $w = 4$ [31]. The lower row is actually the enumeration of the decimal elements, and the middle row is derived by taking the lower row and swapping the index with the value.

Otherwise, a method is to keep dividing by the unity until a symbol within the finite field is obtained, defined by Geisel as “multiply or divide by unity method” [15]. Keep in mind that $\alpha^{2^4-1} = \alpha^{-2^4-1} = \alpha^0 = 1$ [15].

$$\alpha^5/\alpha^{14} = \alpha^5\alpha^{-14} = \alpha^{-9} = \alpha^{-9 \bmod 2^4-1} = \alpha^{-9}\alpha^{2^4-1} = \alpha^6$$

By precomputing the mapping between the binary element and the power Q (9 in the case of α^9) in both directions, multiplication or division becomes a matter of lookups in a multiplication table [35]. All what is left to do is the addition or subtraction of the powers. Symbol α^{11} has the power 11, which is the logarithm to its binary element $1110_2 = 14_{10}$. Conversely, the binary element is the inverse logarithm to the power Q . Within the context of the paper of Plank [31], these tables of length $2^w - 1$ are called **gflog** if the index is the decimal element, or **gfilog** if the index is the power Q (table 5).

3.3.3 Calculating and maintaining checksum words

The element $a_{i,j}$ at position i, j of a Vandermonde matrix is defined to have the value i^j [32, 28]:

$$\begin{bmatrix} 0^0 & 0^1 & 0^2 & \dots & 0^{n-1} \\ 1^0 & 1^1 & 1^2 & \dots & 1^{n-1} \\ 2^0 & 2^1 & 2^2 & \dots & 2^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ (n+m-1)^0 & (n+m-1)^1 & (n+m-1)^2 & \dots & (n+m-1)^{n-1} \end{bmatrix}$$

The matrix has a maximum number of rows since the arithmetic is over a closed field, motivated by operation over words of fixed size [28, 31]. Therefore, n and m must adhere to $n + m < 2^w$. The closed field is needed since Gaussian elimination described in section 3.3.4 of this master thesis is not solvable otherwise [31].

So, for $n = 3$ and $m = 3$ over $\text{GF}(2^4)$, the Vandermonde matrix of dimension $(n + m) \times n = (3 + 3) \times 3 = 6 \times 3$ is [32]:

$$A = \begin{bmatrix} 0^0 & 0^1 & 0^2 \\ 1^0 & 1^1 & 1^2 \\ 2^0 & 2^1 & 2^2 \\ 3^0 & 3^1 & 3^2 \\ 4^0 & 4^1 & 4^2 \\ 5^0 & 5^1 & 5^2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 2 & 4 \\ 1 & 3 & 5 \\ 1 & 4 & 3 \\ 1 & 5 & 2 \end{bmatrix}$$

The element at row 3, column 2 is actually the result of the multiplication of the decimal element 3 with the decimal element 3. With the help of `gflog`, we find that the power representation is X^4 (`gflog[3]=4`). Using the “exponent mod n multiplication method” by Geisel [15] we find $\alpha^4\alpha^4 = \alpha^{4+4} = \alpha^8$. With the help of `gflog`, we find that the decimal representation of the result is 5 (`gflog[8]=5`).

The information dispersal matrix B is derived from the Vandermonde matrix A by a finite sequence of elementary matrix transformations [32, 52]. The element at the i -th row and j -th column is denoted by $b_{i,j}$. The $n \times n$ matrix in the first n words of B has to be the identity matrix, and any submatrix derived from B by deleting m rows has to be invertible [32]. This is achieved by swapping columns and replacing values on the diagonal of the identity matrix by its multiplicative inverse, since $b_{i,j} \cdot b_{i,j}^{-1} = 1$, affecting the whole column that the element is part of. Afterwards, the columns that do not contain a 0 on the height of the row of the replaced element, get subtracted by itself multiplied by $b_{i,j}$. See Plank [32] for the formal description of the algorithm. The deletion of maximally m rows from A does not change the property that A' remains invertible [32]. Since elementary matrix operations do not change the rank of a matrix, the matrix B remains invertible, too [32].

$$B = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \\ 15 & 8 & 6 \\ 14 & 9 & 6 \end{bmatrix} = \begin{bmatrix} 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 1 \\ b_{n,0} & b_{n,1} & b_{n,n-1} \\ b_{n+1,0} & b_{n+1,1} & b_{n+1,n-1} \\ b_{n+m-1,0} & b_{n+m-1,1} & b_{n+m-1,n-1} \end{bmatrix}$$

The calculation of the vector C of checksum words out of the vector of data words $D = [d_0, d_1, \dots, d_{n-1}]$ is now done by following formula [52]:

$$B \cdot D = \begin{bmatrix} d_0 \\ \vdots \\ d_{n-1} \\ \text{---} \\ c_0 \\ \vdots \\ c_{m-1} \end{bmatrix} = \begin{bmatrix} D \\ \text{---} \\ C \end{bmatrix} = E \quad (4)$$

With reference to section 3.3.1 of this master thesis:

$$c_0 = b_{n,0} \cdot d_0 + b_{n,1} \cdot d_1 + \dots + b_{n,n-1} \cdot d_{n-1}$$

$$c_1 = b_{n+1,0} \cdot d_0 + b_{n+1,1} \cdot d_1 + \dots + b_{n+1,n-1} \cdot d_{n-1}$$

...

$$c_{m-1} = b_{n+m-1,0} \cdot d_0 + b_{n+m-1,1} \cdot d_1 + \dots + b_{n+m-1,n-1} \cdot d_{n-1}$$

Actual numbers, \oplus is the exclusive-or operation:

$$D = \begin{bmatrix} 3 \\ 13 \\ 9 \end{bmatrix}$$

$$c_0 = (1 \cdot 3) \oplus (1 \cdot 13) \oplus (1 \cdot 9) = 3 \oplus 13 \oplus 9 = 7$$

$$c_1 = (15 \cdot 3) \oplus (8 \cdot 13) \oplus (6 \cdot 9) = 2 \oplus 2 \oplus 3 = 3$$

$$c_2 = (14 \cdot 3) \oplus (9 \cdot 13) \oplus (6 \cdot 9) = 1 \oplus 15 \oplus 3 = 13$$

All words within the same device are multiplied by the same element in the matrix B , allowing $I \gg n + m$.

Each word in the checksum device is therefore dependent on all data words. Suppose that one data word changes its value. The difference between the former data word d_i and the new data word d_i'' is exactly the same as the difference between the former checksum word c_i and the new checksum word c_i'' [31, 28]. Therefore:

$$c_i'' = c_i + b_{i,j} \cdot (d_j'' - d_j)$$

Consider the case where data devices are distributed amongst computers. An update in a data device requires each node holding a checksum device to update its checksum device. All what is needed is the data word containing the change, not all input words, since the checksum word remains dependent on all data words.

Appending or deleting the last data device is actually the same as an update in relation to an implicit empty data device [28]. In this case, all nodes holding checksum devices need to follow the described update procedure, but all nodes, even possibly unaffected nodes, that hold devices, need to update their value of n and m , too.

For the telephone book example, as it is listed in table 3 on page 6, the following is the procedure for calculating the checksum words: First, a matrix providing space for the whole input and output is selected. Then, the data devices are identified as variables (example: 6 variables) and additional equations are added (example: 18 equations). Finally, the information dispersal matrix is derived from the original matrix and the data devices are multiplied with the coefficients of the information dispersal matrix.

More concretely: The number of data devices and data words is 6 ($\{d_0, d_1, d_2, d_3, d_4, d_5\} = \{43, 33, 51, 90, 65, 34\}$) and the number of checksum devices is 18 ($\{c_0, c_1, \dots\} = \{15, 90, \dots\}$). The data words are treated as variables and are input variables for the 18 additional equations, which are represented as additional rows in the information dispersal matrix B . The Vandermonde matrix A and the information dispersal matrix B therefore have the dimension 24×6 . The input words and output words therefore have the necessary space in the matrix. The resulting vectors D and C are the result of the multiplication as it is performed in formula 4.

3.3.4 Recovering from failures

Suppose that q_d data devices and q_c checksum devices were erased. $q_d + q_c \leq m$ must hold. Since we assume that each device only contains a single word, this is equivalent to q_d missing words in D and q_c missing words in C . Otherwise, there are continuous sequences of $q_d \cdot \frac{8k}{w}$ words missing in D and continuous sequences of $q_c \cdot \frac{8k}{w}$ words missing in C . The erased elements are therefore removed from

$$E = \begin{bmatrix} D \\ \text{---} \\ C \end{bmatrix} \text{ (formula from [52]), resulting in:}$$

$$E' = \begin{bmatrix} D' \\ \text{---} \\ C' \end{bmatrix} = \begin{bmatrix} d'_0 \\ \vdots \\ d'_{q_d - q_c - 1} \\ \text{---} \\ c'_0 \\ \vdots \\ c'_{q_d - q_c - 1} \end{bmatrix} = \begin{bmatrix} e'_0 \\ \vdots \\ e'_{n - q_d + m - q_c - 1} \end{bmatrix}$$

with $n - q_d + m - q_c$ elements. Since the position of the erased elements are known, it is possible to delete the corresponding rows in B and to choose any remaining n rows, resulting in a $n \times n$ matrix B' [31]. Reconstruction is now done by solving for D in

$$B'D = E' \quad (5)$$

Since A is a Vandermonde matrix, any row is linearly independent to any other row [31, 35]. B' , derived from A through elementary matrix operations that do not change the rank of a matrix, is non-singular and invertible [31, 52], since the determinant of any square submatrix consisting of elements $u_j \in GF(2^w)$, $j = 0, \dots, \min(2^w - 1, |A|)$ is never 0: $\det A = \prod_{i < j} (u_j - u_i) \neq 0$ and $\det A = \prod_{i < j} (u_1 + u_j) \neq 0$ [28, 35]. The calculation of the inverse $F = B'^{-1}$ can be done from B' using Gaussian elimination [31, 52]. F consists of elements $f_{i,j}$, where $0 \leq i \leq n - 1$ and $0 \leq j \leq n - 1$ [52]. Afterwards, the data words - the other e'_x are useless as no new information is gathered - get restored [52]:

$$D = B'^{-1}E' = F \cdot \begin{bmatrix} D' \\ \text{---} \\ C' \end{bmatrix} = \begin{bmatrix} f_{0,0} & \cdots & f_{0,n-1} \\ \vdots & \ddots & \vdots \\ f_{n-1,0} & \cdots & f_{n-1,n-1} \end{bmatrix} \cdot \begin{bmatrix} e'_0 \\ \vdots \\ e'_{n-p+m-q-1} \end{bmatrix}$$

$$d_{n-1} = f_{n-1,0} \cdot e'_0 + f_{n-1,1} \cdot e'_1 + \cdots + f_{n-1,n-1} \cdot e'_{n-1}$$

Assume that, in the scope of the running example of $n = m = 3$, D and C , that d_1 , d_2 and c_0 were lost. The recovery calculation is only needed for d_1 and d_2 [31]:

$$B' = \begin{bmatrix} 1 & 0 & 0 \\ 15 & 8 & 6 \\ 14 & 9 & 6 \end{bmatrix} E' = \begin{bmatrix} 3 \\ 3 \\ 13 \end{bmatrix}$$

$$F = B'^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 6 & 10 & 13 \end{bmatrix}$$

$$d_1 = f_{1,0} \cdot e'_0 + f_{1,1} \cdot e'_1 + \cdots + f_{1,n-1} \cdot e'_{n-1} = (1 \cdot 3) \oplus (1 \cdot 3) \oplus (1 \cdot 13) = 3 \oplus 3 \oplus 13 = 13$$

$$d_2 = f_{2,0} \cdot e'_0 + f_{2,1} \cdot e'_1 + \cdots + f_{2,n-1} \cdot e'_{n-1} = (6 \cdot 3) \oplus (10 \cdot 3) \oplus (13 \cdot 13) = 10 \oplus 13 \oplus 14 = 9$$

Gathered	Decoded	Notes
{3, 141, 5, 92, 107, 230, 90}	{3, 141, 5, 92}	Perfect information case
{3, 141, ?, 92, 107, 230, 90}	{3, 141, 5, 92}	One erasure
{3, ?, ?, 92, 107, ?, 90}	{3, 141, 5, 92}	Three erasures (max.)
{3, ?, ?, ?, 107, ?, 90}	-	Too many erasures

Table 6: Recovery cases for $n = 4$, $m = 3$, $\{d_0, d_1, d_2, d_3\} = \{3, 141, 5, 92\}$.

If needed, c_0 can then be reconstructed according to section 3.3.3 of this master thesis:

$$c_0 = b_{n,0} \cdot d_0 + b_{n,1} \cdot d_1 + \dots + b_{n,n-1} \cdot d_{n-1}$$

$$c_0 = (1 \cdot 3) \oplus (1 \cdot 13) \oplus (1 \cdot 9) = 3 \oplus 13 \oplus 9 = 7$$

For the telephone book example, as it is listed in table 3 on page 6, the following is the procedure for restoring the original data devices: As soon as two telephone book pages are gathered, there are enough devices available for the reconstruction since eight words are known now. First, the information dispersal matrix B is calculated in the same way as it was for a calculation of checksum words. Then, rows are deleted from B , resulting in B' . These rows are corresponding to the missing words. Then, B' is inverted using Gaussian elimination, and F is obtained. Finally, F is multiplied with the vector of gathered words and the vector of original data words is obtained. Because we assume that one device holds exactly one word, the original devices are hereby restored.

These two pages may be page 1 containing the devices $\{d_0, d_1, c_0, c_1\} = \{43, 33, 15, 47\}$ and page 4 containing the devices $\{c_6, c_7, c_8, c_9\} = \{55, \text{ec}, \text{ec}, \text{bf}\}$.

3.3.5 Properties

Since the maximal number of devices are $n + m < 2^w$, it is impossible to choose 100% redundancy ($n = m$), if $n > \frac{2^w}{2}$. Choosing a larger w results in larger words, each of size 2^w bits. Recovery of all original data devices is only possible if at least n devices are gathered, whether these are data devices or checksum devices. Every device is dependent on every data device. Therefore, every device contributes to the recovery. See table 6.

n symbols are always needed, even if all participants share a common dictionary. Applying a dictionary only results in calculating the exclusive or (\oplus) of the elements of the dictionary and all results. Even occupying some data words with a dictionary word has no use since the number of devices that need to be gathered is exactly the number of unknown data devices.

A node can only make use of a device if the parameters n , m and the position of the device is known, 0 being the first position. This is seen in action in figure 35 of this master thesis. n and m are assumed to be known to the node. But each device is labeled with the required position which is offset by the position of the node in the moving window. Whether the device is a data device or a checksum device is instantly clear, since no checksum device has a position $< n$.

Data devices	Calculated data and checksum devices
$\{3, 141, 5, 92\}$	$\{3, 141, 5, 92, 77, 24, 71, 55, 192, 84, 130, 244, 106, 122\}$
$\{3, 141, 5, 93\}$	$\{3, 141, 5, 93, 149, 218, 216, 88, 7, 10, 221, 133, 247, 187\}$
$\{3, 141, 4, 92\}$	$\{3, 141, 4, 92, 12, 38, 157, 74, 160, 106, 74, 182, 179, 174\}$
$\{141, 3, 5, 92\}$	$\{141, 3, 5, 92, 138, 219, 80, 154, 202, 121, 72, 99, 7, 211\}$

Table 7: Example of the effects of changes in the input devices ($n = 4$, $m = 10$, $k = 1$, $w = 8$). Observe how the checksum devices change completely.

k is instantly known as soon as one device has been obtained. k is simply the number of bytes of the device.

An update to a data device affects the node holding the data device. Furthermore, all nodes holding checksum devices are affected and need to update their checksum device as described in section 3.3.3. The effect is shown in table 7. While this is efficient in terms of bandwidth, the analysis is not influenced since the number of nodes that have to be contacted stays the same.

Altering the redundancy allows to control the proportion of nodes that have to be contacted to the total number of nodes for successful recovery. Very low redundancy requires the agent gathering devices to contact nearly all nodes. Very high redundancy results in a very low number of nodes that have to be contacted.

Since there is the maximum of $n + m$ devices, problems occur for too small block sizes k if the dataset changes. There is the tradeoff between w , k and $n + m$. A big w results in big words, which may enlarge the devices too much if no addition is planned. But a big w allows the addition of a lot of devices without contacting the rest, especially at a later time. See section 3.3.3. Similarly, a big k allows for the reduction of devices, but each new device may allocate too much wasted space. For example, adding one byte may result in a new data device of one mebibyte. Allowing the possibility of an exorbitant high number of checksum devices results in larger data words.

Regarding the limit of $n + m$, there are implementations that break up the message into chunks and calculate different RSEC-encoded sets of devices. With such a setting, it is not possible to gather *any* n devices anymore as the data words do not contribute to all checksum devices anymore. No use of chunking is made throughout this thesis.

If there is external information available, it is possible to extract information without having acquired n devices, called brute-forcing. This is achieved by testing through all possible values of an unknown data device, observing the calculated values for other unknown data devices and checking the plausibility with the help of external information. Again, no such use of an extension is made throughout this thesis.

3.4 Applying redundancy

Up to now, it has become more ascertainable thanks to the previous parts, how the main idea of this master thesis can utilize locality as locational advantage. As a matter of fact, the idea of high availability aiming for a reduction of

the dependence on locality is supported by the both the direct application of redundancy to data elements and by the application of redundancy to data structures. A location is no longer the contrary to EVERYWHERE, but a location gains a locational advantage.

Resulting from the preceding explanations, figure 1 and 2 dealing with the application of redundancy can be specified as follows:

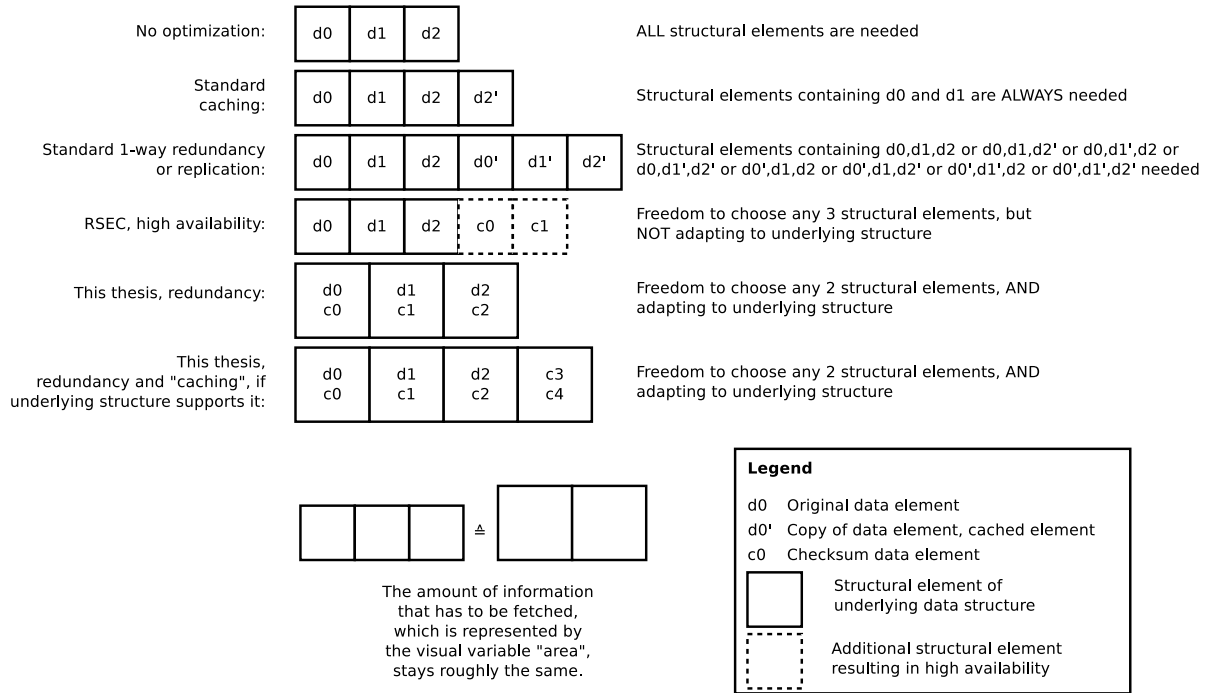


Figure 31: The method of this thesis has the ability to adapt to underlying structures in order to integrate seamlessly. Bringing freedom to choose randomly from a set, thereby lowering the need to discover all elements that are accessed together normally. Supporting random accesses even if there is less than full redundancy. Technically, the method of this thesis identifies the data elements and replaces the data elements with structural elements that contain a mixture of original data and checksum data.

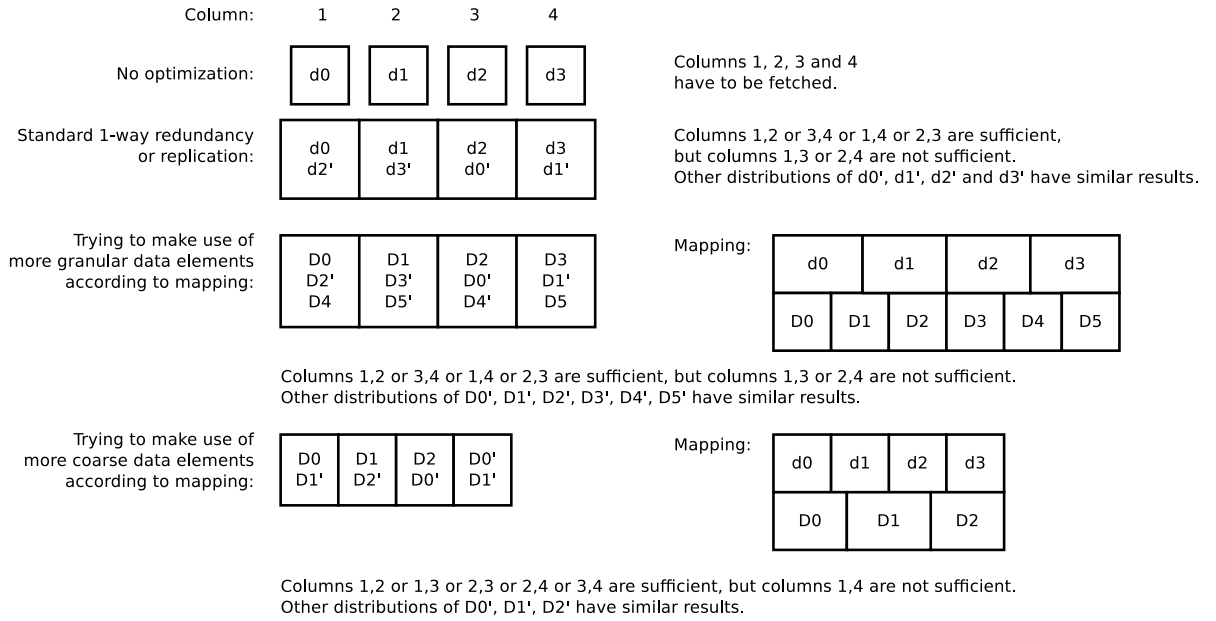


Figure 32: As long it is possible to encounter already gathered elements or copies of already gathered elements, there is no freedom to choose a random sequence of structural elements. No distribution of cached elements can guarantee truly random accesses, even if adaption to underlying structures is performed.

In the following sections, redundancy is applied to the DHT in the Connectivity layer (section 3.4.2) and to the Quadtree in the Core services layer (section 3.4.3). In the preceding section 3.4.1, the redundancy, as it is applied afterwards, is formulated and written as a formula. The formula plays a key role in the cost model, which is applied after the application of the redundancy.

3.4.1 Adapting the redundancy

This section describes how the redundancy is being adapted to the history h consisting of the last operations over the set of devices. There are two classes of operations: The first class consists of read queries that make use of the set of devices. The second class consists of updating queries.

h is the array containing the last $|h|$ operations on the set. The only two possible operations are R and U . $|h|$ is limited to maximally 50 throughout this thesis. h is initially empty and gets filled each time an operation is made by applying $\text{add}(op \in \{R, U\}, h)$. For example, after four reads of the set and one update, $h = [R, R, R, R, U]$. The oldest value gets deleted as soon as the array is full:

$$\begin{aligned}
 h &= [R, R, R, R, U], |h| = 5 \\
 h &= \text{add}(R, h_{old}) = [R, R, R, R, U, R], |h| = 6 \\
 &\dots 44 \text{ operations } \dots \\
 h &= [R, R, R, R, U, R, \dots, R, R, U], |h| = 50 \\
 h &= \text{add}(R, h_{old}) = [R, R, R, U, R, \dots, R, R, U, R], |h| = 50
 \end{aligned}$$

0% redundancy is available if no RSEC is performed. For I input data bytes there are $J = 0$ checksum bytes available. 100% redundancy results in $J = I + \varepsilon$ checksum bytes, ε being the number of padding bytes. 200% redundancy results in $J = 2(I + \varepsilon)$ checksum bytes. In 2-way redundancy, there are $m = 2n$ checksum devices. Generally, for v -way redundancy, there are $J = v \cdot (I + \varepsilon)$ checksum bytes available. Full redundancy is achieved if every participating node holds 1 distinct data device and $n - 1$ distinct checksum devices, thereby each participating node becomes capable of restoring the I input bytes, or n data devices. Full redundancy is therefore $p - 1$ -way redundancy, p being the number of participants. The equation $m = (p - 1) \cdot n$ holds for full redundancy. Full redundancy has the same effect as copying I input data bytes to all participants.

Each time an update to the set is made, the opportunity to adjust the redundancy is taken. Redundancy r is defined as follows:

$$r = \begin{cases} 1 & \text{if } updateCost \leq |R| \text{ or } readProfit \leq |U| \\ \frac{|R|}{|h|} & \text{otherwise} \end{cases} \quad (6)$$

$|R|$ is the number of elements R in h , analogous for $|U|$. $updateCost$ and $readProfit$ depend on the piggybacking method and are determined with the cost model introduced in section 4.1⁴. During a read operation, the node giving out the checksum device increments a read counter. During an update, a deterministically elected node participating in the set⁵ finds and contacts all participating nodes. The number of read queries is gathered from all nodes, which reset their counter afterwards. Then, the elected node determines the redundancy r with the help of equation 6. If current k and r do not result in $n + m < 2^w$ anymore, then the elected node has to determine a suitable and padding-reducing blocksize k . k can be held fixed if the piggybacking methods requires it. Afterwards, the elected node calculates checksum device after checksum device and sends the checksum devices to the appropriate nodes. This is actually the Fan-in algorithm [31]. Of course, if the participant can not derive the data device itself, then the data devices are distributed, too.

If $n + m < 2^w$ still holds, only the changed data words need to be communicated to the affected nodes holding data devices and need to be communicated to all nodes holding checksum devices. Blocksize k can not be changed in this case, since the node responsibility of checksum words shift in a cascading fashion. But, additions of blocks or deletions of blocks are still possible, as it is explained at the end of section 3.3.3 of this master thesis.

3.4.2 Altering Chord in the Connectivity layer

This section gives an idea how RSEC-encoded sets can be adapted to existing data structures, as depicted in figure 31. Furthermore, scope expansion is explained.

The example used in the following section is Chord with its finger tables, as it is used by Tanin, Harwood, Samet et al. [21, 49, 48, 47]. But the methodology presented in this section remains generally applicable to any data structure that

⁴ $updateCost$ and $readProfit$ actually depend on r themselves. This loop is handled by both the modifications and analysis.

⁵For example, the peer with the lowest IP address. Depends on the piggybacking method actually.

24	2f	31	3a	50	5d	64	69	6e											
	2f	31	3a	50	5d	64	69	6e	7b										
		31	3a	50	5d	64	69	6e	7b	c4									
			3a	50	5d	64	69	6e	7b	c4	e0								
				50	5d	64	69	6e	7b	c4	e0	e4							

Figure 33: Some moving windows over the DHT. The rightmost moving window for the node at address 6e has $L = R_{win} = 4$ neighbours on each side.

contains a linked list and optionally random skip pointers, similar to the method of Aspnes et al. [4], which is also generally applicable. For example, the skip graph used by Toda et al. [50] or Huq and Ghosh [24], the 1D skip graph used in MURK-Ran [14], the n-dimensional skip graph used in MURK-SF [14] or the rainbow skip graph [17] all can make use of the presented methodology on the list on level 0.

Tanin, Harwood, Samet et al. [21, 49, 48, 47] exploit parallelism to speed up spatial queries. Parallelism defies the use of redundancy since the query objects that operate in parallel do not communicate with each other. Still, applying redundancy to a strategic place helps to improve query latency. Tanin, Harwood, Samet et al. [21, 49, 48, 47] remain dependent on the Connectivity layer. Actually, the Chord lookup method is invoked a large number of times. Not only are the nodes residing at level $l = f_{min}$ looked up this way, but also the children of each control point. Each node returning a spatial object needs to find at least the issuing node or the node holding extra information, which itself needs to find the node which issued the query. As a matter of fact, even if parallelism is taken into account, at least $O(\log N)$ hops to the nodes at level f_{min} are needed, and 0 to maximally $O(\log N + f_{max} - f_{min})$ message hops during the traversal of the tree [47]. Assuming a network of $N = 1000000$ nodes, this are at least $\log(1000000) \approx 20$ hops on average. Taking the parameter $\alpha_{Chord} \approx 0.5$ from [44] into account - shifting in a zero bit from the distance does not necessitate to follow a finger -, this are still ≈ 10 hops on average within a tight bound. This is at least one second of latency if 100ms latency to establish a connection with a peer is assumed. Caching the addresses of the children of a control point alleviates the need to optimize the lookup, but cache misses make use of the Chord lookup method again. Therefore, it makes sense to optimize the lookup in terms of number of contacted nodes.

In order to describe the modification, a model is presented. Then, parameters of the model are fitted to represent the case where only the redundancy is increased, i.e. the case where the modification adapts to the underlying structure. Afterwards, scope expansion is performed.

The model consists of a moving window containing the addresses of the L preceding neighbours, the own address, and R_{win} succeeding neighbours, depicted in figure 33. Every node has L preceding pointers, even the nodes

at the beginning of the identifier space. Because of modular arithmetic, the same holds for the nodes at the very end of the identifier space with their R_{win} successors.

The $L + 1 + R_{win} = s$ addresses concatenated together form the I input bytes to a RSEC-encoded set of devices. Each node receives some checksum devices. Redundancy is determined with the help of the adaptive redundancy algorithm described in section 3.3.3, and will be discussed in more detail later. The number of participants, p , is $s = L + 1 + R_{win}$ in this case. n is p . k is held fixed as the size of an address of a node, no padding is applied. w and m are defined by section 3.3.3. For example, for $p = 9$, there are $n = 9$ input data devices. Full redundancy results in $n - 1 = 8$ checksum devices at each node, totalling to $m = 8p = 72$. No redundancy results in 0 checksum devices, and the moving windows are unusable: the fact that each node is having its data device, the address, does not help. Apart from 0, $r_{min} = \frac{1}{p-1} = \frac{1}{8}$ is the lowest redundancy that is supported. In this case, each node receives one checksum device. The effective redundancy has to be rounded to the nearest multiple of the minimum redundancy $r_{min} = \frac{1}{8}$. Figure 34 shows the distribution in the case of $r = \frac{2}{8}$.

The number of nodes that have to be contacted for successful reconstruction of data devices is

$$z = \left\lceil \frac{p}{\left(r + \frac{1}{p-1}\right)(p-1)} \right\rceil \quad (7)$$

Note that for $r = \frac{2}{8}$, $z = 3 = \left\lceil \frac{p}{\left(r + \frac{1}{p-1}\right)(p-1)} \right\rceil$, not $z = 5 = \left\lceil \frac{p}{r(p-1)} \right\rceil$. This can be checked by counting nine boxes in figure 34. Both data devices and checksum devices contribute towards the reconstruction of the data devices.

The crucial redundancy (equation 6) is not only adapting to the history h , but is also affected by the analysis of this piggybacking method being described. The analysis in section 4.2.3 derives $updateCost = p - 1$ and $readProfit = \lceil \alpha_{Chord} \log p \rceil - z + 1$ for the worst case. r is therefore defined as:

$$r = \begin{cases} 1 & \text{if } p - 1 \leq |R| \text{ or } \lceil \alpha_{Chord} \log p \rceil - z + 1 \leq |U| \\ \frac{|R|}{|h|} & \text{otherwise} \end{cases} \quad (8)$$

z is the number of nodes that have to be contacted for a given redundancy and was defined in equation 7. In order to deal with a recursive definition reasonably, the r in equation 7 has the value $\frac{|R|}{|h|}$. α_{Chord} is 0.5 [44].

Node lookup

Each node knows the IDs of the moving windows crossing its address, as well as the own position in the moving windows through metadata. The issuing node constructs a lookup message containing the target key. The issuing node creates an array of empty buckets representing the moving windows of which the issuing node is part of. Then, the issuing node fills the buckets with all available devices in the hope that subsequent nodes can make use of this information. A visualization is provided in figure 35. For example, the bucket with the label 702d receives the data device and some checksum devices, depending on the

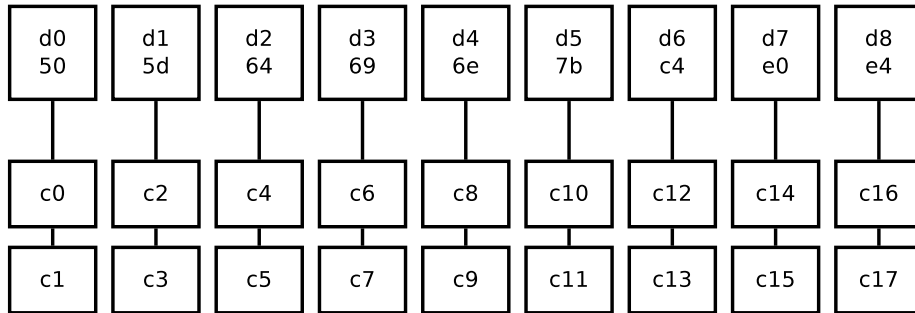


Figure 34: How the data devices and checksum devices are distributed over a moving window for redundancy $r = \frac{2}{8}$. The number of actually available checksum devices can be read from the formula $r = \frac{2}{8}$, and is therefore 2. $8 = p - 1$ is the maximal number of checksum devices that a node can hold. The devices d_4 , c_8 and c_9 are situated at the node with the IP address $6e$.

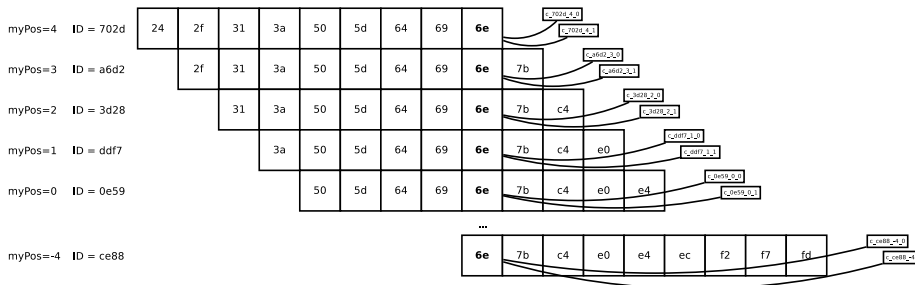


Figure 35: Some of the checksum devices node $6e$ is holding ($r = \frac{2}{8}$). The data device, that is the own address, and all checksum devices are attached to the message looking up a node.

redundancy. Then the lookup message is sent to the next node according to the finger table [44].

The intermediate node adds its devices and checks whether the information is sufficient to reconstruct the target node. The intermediate node proceeds as follows:

1. The node inspects the lookup message for buckets labeled with a moving window ID which the node itself is part of.
2. If so, the node adds its data device and all checksum devices belonging to such a bucket.
3. The node creates additional buckets if there are moving window IDs from which the node is part of, but that are not in the lookup message. For each created bucket, the node fills in the data device and every checksum device.
4. The node checks each bucket. If there are enough devices for a given bucket, the data devices can be reconstructed, as described in section 3.3.4. The node remembers the reconstructed moving window and deletes the bucket from the lookup message.
5. The node checks whether each reconstructed moving window contains the target address. If so, the lookup can be sent to the destination instantly. If not, the node introduces the data devices to all other buckets if the other buckets representing moving windows are overlapping. Figure 37 depicts this operation as red arrows.
6. The node checks again and continues until it is not possible anymore to reconstruct a moving window. The node subsequently sends the lookup message to the next node according to the finger table.

The node increments the reading counter of a moving window whenever the node reads a device from it - see section 3.4.1. The node increments such a counter only once per lookup message and moving window.

Node update, node insertion and node deletion

In Chord there are three types of events affecting moving windows: addresses of nodes change, there are nodes being introduced to the DHT and there are nodes that depart. Afterwards, Chord heals the ring [44]. Each time a node detects changes, the overlay needs to be updated afterwards, as shown by figure 36. All nodes that were possibly part of all moving windows that the missing peer was part of need to be contacted and found. Then, the nodes elect the node with the lowest hash in the identifier space with the task to update the moving window situation to a valid state. The node with the lowest hash in the identifier space is selected because the algorithm can traverse the successor list.

A moving window is constructed the same way as already described. A moving window is updated by sending the new address to every participant in the moving window, who integrate the update to the checksum devices as described in section 3.3.3.

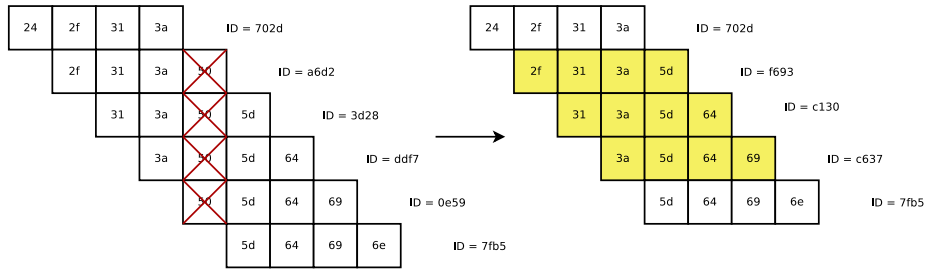


Figure 36: The moving windows containing the address of a peer gone missing need to be invalidated. New moving windows with a new ID need to be constructed. The inverse case of a node insertion also causes invalidation. In such a case the original situation is on the right and the invalidated moving windows are on the left, each marked with a node containing a red X.

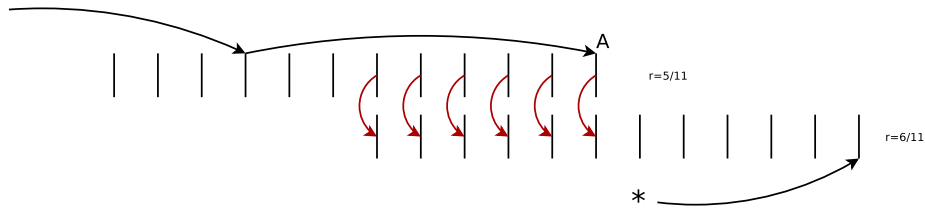


Figure 37: The search aiming for the rightmost peer gathers enough devices in order to jump to the target. Node A alone does not communicate enough devices for the lower moving window ($6 < 12$). But the data devices derived from the reconstruction of the upper moving window give the necessary information ($6+6$ red arrows = 12). This case shows that the last node A is also important.

Overlaying the moving windows and scope expansion

In order to fit the model to Chord, the parameters need to be set in order to adapt to the underlying structure. The model fits perfectly to the existing moving windows consisting of the predecessor, the current node and the successor by setting $L \leftarrow 1$, $R_{win} \leftarrow 1$ and $s \leftarrow L + 1 + R_{win} = 3$.

Scope expansion alters the locality properties and the consistency maintenance properties of the underlying method. Scope expansion is achieved by allowing the moving window to be larger than $s = 3$. The working number of this thesis is $s = 9$ if it is not made explicit otherwise. As a consequence, there is an advantage for lookups and additional costs in terms of number of nodes that have to be looked up in the case of updates. For skip graphs, $L = R_{win}$ is kept. For Chord and scope expansion, R_{win} is set to 0 in all instances since all searches use finger tables; such a search is illustrated in figure 25. The last node, the target node, also stores checksum symbols since the checksum devices held by the last node may be used for calculating the values of the moving window which contributes to other moving windows as soon as the data words become known; see figure 37.

This modification altering the Connectivity layer can be explained with the telephone book illustration as follows: Originally, every person has a private

telephone contact list of other members. If the telephone number of a person located in Parkes, Australia had to be looked up, then the telephone number of someone who has most to do with Australia was picked. This modification has resulted in following change: the private telephone contact list of everybody was enlarged by a constant number of entries. These entries are parts of the telephone numbers of the neighbours of a person. Whenever the search for a telephone number arrives in the area close to the destination, the last steps can be skipped as soon as enough parts of telephone numbers are gathered from the large contact lists. Because the contact lists are always enlarged by entries pertaining to neighbours, it can be shown that there are less steps on average. As a result, the required granularity of the search is lowered. It is enough to resolve a neighbourhood, as a single house does not have to be resolved anymore. The same procedure can operate on a more generalized or more coarse level on a map. Dependence on locality is therefore minimized.

3.4.3 Altering the Core services layer

This section deals with the application of redundancy to data elements itself. Consequently, the original data elements are replaced by a mix of fragments of original data elements and checksums. The application of redundancy is done in the Core services layer of the method of Tanin, Harwood, Samet et al. [21, 49, 48, 47].

The spatial object S is assigned to a set of responsible control points [47]. The list G_{min} , the list of responsible control points at level f_{min} , is determined by invoking `Subdivide(S, root, Gmin)` [48]. Afterwards, the spatial object is forced down to lower levels if the spatial object only covers one subdivided area. Departing from the original method by Tanin, Harwood, Samet et al. [21, 49, 48, 47], the list of responsible control points at all levels, $P = G_{eff}$, is calculated. G_{eff} therefore contains the complete list of responsible control points for a spatial object. The list contains all control points that are effectively responsible for the spatial object. G_{eff} is calculated by using following method called `DetermineG_eff`:

```
//Variables and methods are kept consistent with the original method
global G_eff
DetermineG_eff(object X, control point u) {
    if(X is not within exactly one R(C(u,i)) or L(u)=f_max) {
        add u to G_eff
    } else {
        for i:=1 to 4 do sequentially {
            if(Ints(X,R(C(u,i))) is not empty) {
                DetermineG_eff(X,C(u,i))
            }
        }
    }
}
```

Each node responsible for the control point listed in G_{eff} stores the complete replica of the spatial object [21, 49, 48, 47]. The general method introduced in this section is the application of the second last idea enlisted in figure 31. The modification to the original approach works as follows:

The serialized spatial object S has I bytes. A file containing the spatial object is also a binary serialization of the spatial object. These I bytes are the input bytes to a RSEC-encoded set of devices. n is the number of responsible

nodes, or $n = |G_{eff}| = p$, p being the number of participants. Thereby, $n = p$. The redundancy r is 0 if there are no checksum devices. In the case of no redundancy, each node i at control point G_{eff_i} only holds a data device of spatial object S . Full redundancy is defined as $r = 1$ and results in $p - 1$ checksum devices at each node, m is $m = (p - 1) \cdot n$ in this case. For $n = 9$, $m = 8 \cdot 9 = 72$. k , w and m are defined according to the constraints defined in section 3.3 of this master thesis. Storing 1 additional checksum word at a node results in redundancy $r = \frac{1}{p-1} = \frac{1}{8}$. The situation is analogous to the situation in figure 34.

Updates are handled according to section 3.4.1 of this master thesis. The redundancy can be freely chosen, such as

$$r = \frac{|R|}{|h|} \quad (9)$$

The overlay is now made by distributing one distinct data device to each participant. The checksum devices are distributed evenly amongst the participants.

No scope expansion is performed. Neither neighbours to the area controlled by the control point nor parents hold additional checksum devices.

If the spatial object is simply a point, then the modification brings no benefits. If the underlying method maximizes locality properties, then definitions of groups of spatial objects that are likely accessed together can be identified. Tanin, Harwood, Samet et al. [21, 49, 48, 47] do not define such groups⁶. As a consequence, the piggybacking method can not help more than it does already. Still, it is possible to define such a grouping. For example, points are commonly stored on the same spatial layer and have attributes. Even if the query statistics are unknown, points can be grouped opportunistically. For example, all points with the same value of an attribute can be grouped together. Grouping the points does not change the discoverability. The only change results in possibly more data that needs to be downloaded to the node issuing the query since the minimum bounding rectangle (MBR) or the three-dimensional variant minimum bounding box is enlarged, as shown in figure 38. Another candidate for grouping consists of the pixels of a raster image. Because the effects are not changing the impact of erasure coding on locality, no such optimizations are made.

The serialization may destroy the correspondence between the part of the area of the polygon with the area of the control point. By treating the vertices of a polygon as data words (n =number of vertices), and by making intersection calculus on the vertices instead on the MBR, the correspondence can be kept, as shown in figure 39. A node may then perform intersection calculations on the part of the polygon. An analogous approach can be made with the pixels of a raster image, if the spatial objects are pixels, not the raster image itself.

Again, it is not the task of the overlaying piggybacking method to group data. The analysis remains on the level of spatial objects, the finest resolution of grouping mentioned by Tanin, Harwood, Samet et al. [21, 49, 48, 47]. The nodes can only determine whether an object intersects by comparing the MBR in the metadata with the search rectangle.

For the telephone book illustration see page 39 f.

⁶Except all spatial objects under a control point, which is useless since the objects reside at the same node.

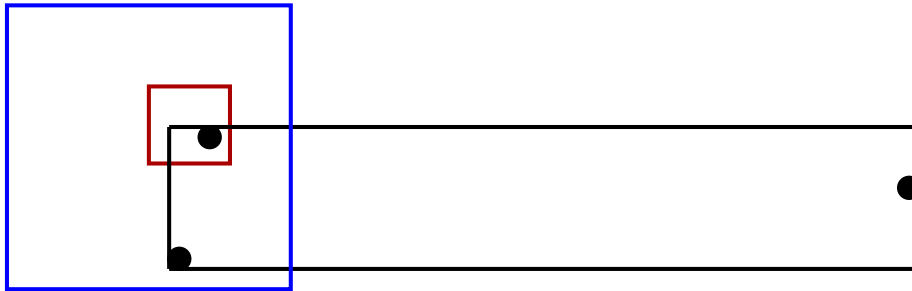


Figure 38: Effect of grouping: The blue area controlled by a control point may respond with three points instead of one even though the query window in red only selects one point finally. In isolation, the MBR of a point would have no area.

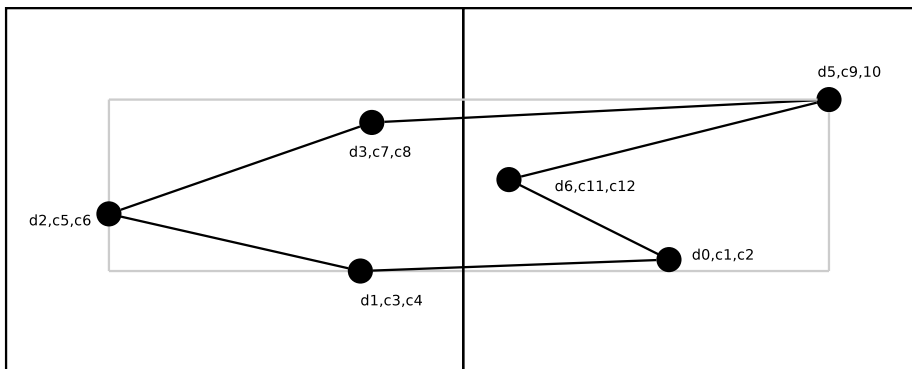


Figure 39: A redundant polygon distributed over two areas controlled by different control points for $r = \frac{2}{1}$. The MBR is in grey and would normally be the only information available for intersection calculations.

3.4.4 A typical insertion and search procedure

The user interacts with a Graphical user interface (GUI). It is possible to start a range search, insert an object or select an object and delete it.

P2P database by Tanin, Harwood, Samet et al.

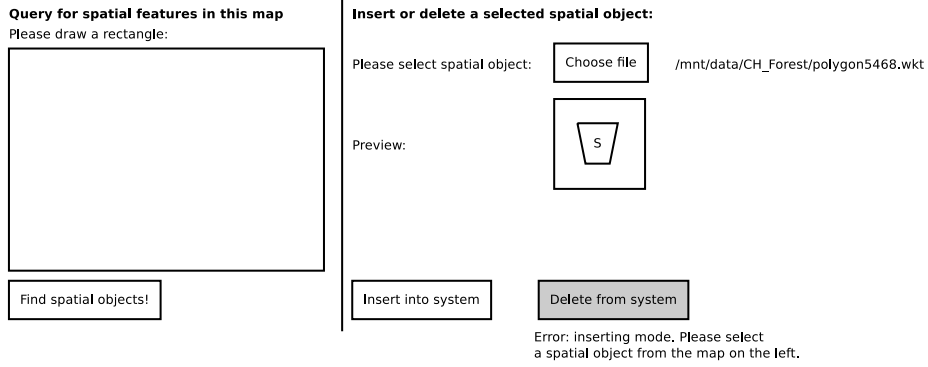


Figure 40: The Graphical user interface.

After the user presses the button labeled “Insert into system”, the application layer makes use of the service “object insertion” of the interface between the Application layer and the Core services layer. The application layer delegates the task of inserting the object into the system. The Core services layer adds redundancy and calls `InsertObject(Object S)`, which determines the list of control points at level f_{min} that intersect with the spatial object, called G_{min} .

The list G_{min} is determined with the procedure `Subdivide()`. Namely, the spatial object is subdivided into parts until the fundamental minimum is reached. Assume $f_{min} = 2$. The red polygon in figure 41 is subdivided into a lower part and an upper part for the first iteration of the recursive function `Subdivide()` [47]. Furthermore, the `Subdivide()` is recursively called again for the areas where the spatial object intersects, unless the control point is at the level f_{min} . For the spatial object in figure 41, $G_{min} = \{AA, AB, AC, AD, CA, CB, CC, CD\}$. Additionally, departing from the original method, G_{eff} is calculated by setting temporarily $f_{min} = f_{max}$ and using the modified version of `Subdivide` with the replaced `if`-statement. For the spatial object in figure 41, $G_{eff} = \{AC, AD, CA, CB, AAD, ABC, CCB, CDA\}$. $n = |G_{eff}| = 8$. Assume that $r = r_{min} = \frac{1}{|G_{eff}|-1} = \frac{1}{7}$. Consequently, $m = |G_{eff}| = 8$.

In order to add redundancy, the spatial object first has to be serialized. The file already contains a serialization of the spatial object. In this case, the file `/mnt/data/CH_Forest/polygon5468.wkt` has the content `POLYGON((4 7,10 7,9 2,5 2,4 7))`. After adding the padding byte, the input size is $I = 31 + 1 = 32$. Therefore, $k = 4$.

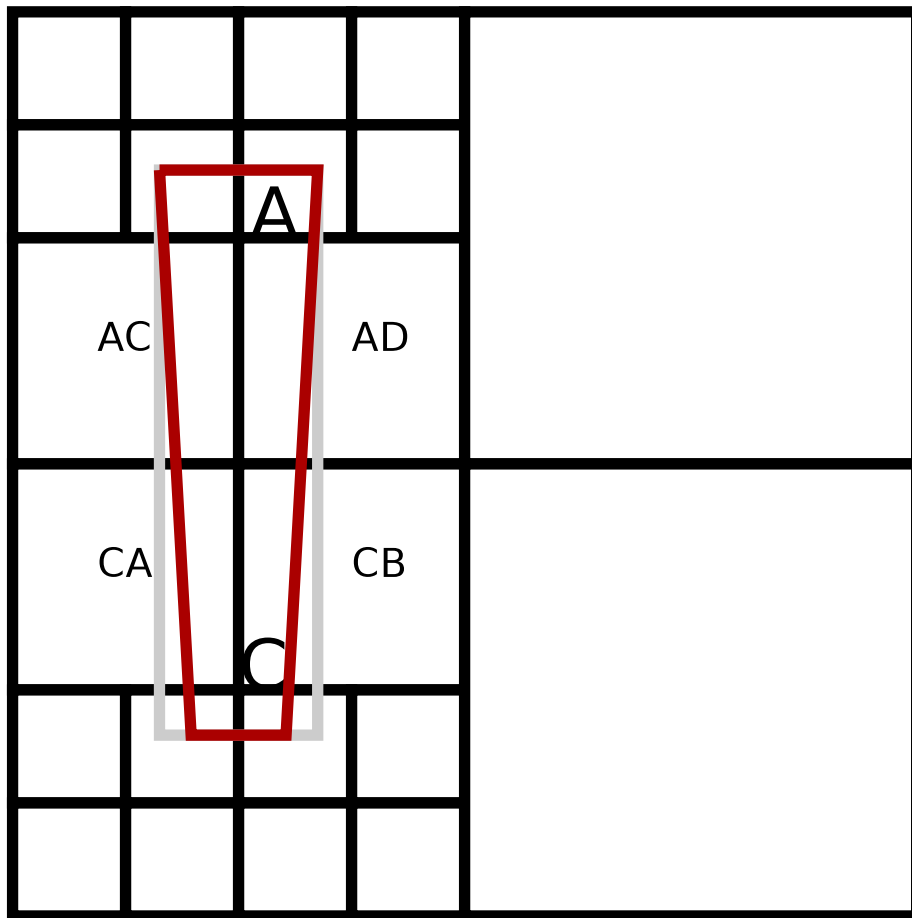


Figure 41: The spatial object (red) in the tree. The MBR is grey. Adapted from Tanin, Harwood and Samet [47].

ASCII	P	O	L	Y	G	O	N	(...	7))	
Base 10	80	79	76	89	71	79	78	40	...	55	41	41	128
Base 16	50	4f	4c	59	47	4f	4e	28	...	37	29	29	80
Device	$d_0 = 504f4c59$				$d_1 = 474f4e28$...	$d_7 = 37292980$			

Table 8: The serialized spatial object S and the resulting data devices. $k = 4$

All needed parameters for the calculation of the checksums are known. The resulting devices are assigned to the nodes in G_{eff} according to figure 34 as follows:

Node AC	$d_0 = 504f4c59, c_0 = a1f8189b$
Node AD	$d_1 = 474f4e28, c_1 = b0a4b385$
Node CA	$d_2 = 28342037, c_2 = ed89183f$
Node CB	$d_3 = 2c313020, c_3 = 9c0a9177$
Node AAD	$d_4 = 372c3920, c_4 = ac0cdb4d$
Node ABC	$d_5 = 322c3520, c_5 = 82d3ead8$
Node CCB	$d_6 = 322c3420, c_6 = e1799e9f$
Node CDA	$d_7 = 37292980, c_7 = 43919b98$

Table 9: The resulting devices and assignment to the nodes.

As part of the `InitiateInsertion` procedure, the insertion is delegated in parallel to the nodes listed in G_{min} . The nodes responsible for the control points AA, AB, AC, AD, CA, CB, CC and CD residing at the addresses `Hash(AA)`, `Hash(AB)`, `Hash(AC)`, `Hash(AD)`, `Hash(CA)`, `Hash(CB)`, `Hash(CC)` and `Hash(CD)`, respectively, are contacted with the help of the Connectivity layer, which is explained next.

On the Connectivity layer, a control message containing the necessary devices, the Minimum Bounding Rectangle of the spatial object S , the target node t_C and the procedure information `DoInsert` is routed towards target node t_C . The devices assigned to the same prefix as the node at level f_{min} are attached to the message. For example, node AA receives a control message containing the devices d_4 and c_4 , since the assignment AAD of the devices shares the same prefix with AA. In this example, only two devices are sent with each control message. This must not hold for the general case.

Assume that the target node is

$$t_C = \text{Hash(AA)} = 801c34269f74ed383fc97de33604b8a905adb635.$$

The node V at address `238efd6b43cf340d153d55b9cb039f490637f7d5` issues the query. A subset of the distributed hash table is as follows:

238efd6b43cf340d153d55b9cb039f490637f7d5	Node V
2f4466b21d0e7be7dbcaef8442633a7438793889	Node responsible for control point CAB
...	
36d9a6df06b9f610f7db8e036896ed03662d168f	A participating node
391dfd55226bd544eba386bba448e869cd9e9f74	A participating node
4c994f835d5b565f9f4a2656d73e40c91998a7a1	A participating node
4fffeef313824dd1205430053d405828d5a491ff	A participating node, p_0
57c513654c56a969364760fe15d7343c51750594	A participating node, p_1
58251f478ae442b85c62861e078022242aa84a46	A participating node, p_2
6094b6877e6d5bbb523c1688d3163ab9054a30be	A participating node, p_3
6acbc1e3bc738effad1f88f6d3d87693e17fa5a9	A participating node, p_4
71cbcf163cf812ab6f6fb5934a1ebc27ec3f9e3f	A participating node, p_5
7c89bbd1a0fc355a99696bdabd54653471204e76	A participating node, p_6
801c34269f74ed383fc97de33604b8a905adb635	Node responsible for control point AA, p_7
...	
b1fb3bec6fdb22e19a94fe4c6c4481ccba2ee9f0	Node responsible for control point AC
f3af8f9ca13efbcf7ca51aff3995b160e590f80e	Node responsible for control point AAD

Table 10: A subset of the distributed hash table. If applicable, p_i is the name of a node in the moving window of size $s = 8$ defined by node AA.

Node V has the node responsible for control point CAB as its successor. The routing towards key 801c34269f74ed383fc97de33604b8a905adb635 is achieved by following a finger pointer in node V. The finger pointer, which is smaller than the key 801..., but which has the lowest difference to 801... compared to other preceding finger pointers in the list of finger pointers at node V, points to the Internet Protocol (IP) address of the node p_0 . p_0 is participating in a moving window defined by AA, amongst others. The moving window is enriched with redundancy. Since $r = \frac{3}{7}$, every node holds three additional checksum devices:

Node p_0	IP $_{p_0} = d_0 = \text{fdfd11...7a}$, $c_0 = \text{eb7dda...18}$, $c_1 = \text{6c9605...7f}$, $c_2 = \text{b77d6d...29}$
Node p_1	IP $_{p_1} = d_1 = \text{fdfdfa...68}$, $c_3 = \text{27c348...61}$, $c_4 = \text{ebcb62...b5}$, $c_5 = \text{e607dc...55}$
Node p_2	IP $_{p_2} = d_2 = \text{fdfd99...4f}$, $c_6 = \text{cd6ad5...04}$, $c_7 = \text{5708e7...18}$, $c_8 = \text{62e866...1a}$
Node p_3	IP $_{p_3} = d_3 = \text{fdfd14...16}$, $c_9 = \text{e5b3e8...b0}$, $c_{10} = \text{64fc1b...b5}$, $c_{11} = \text{0adbd9...3c}$
Node p_4	IP $_{p_4} = d_4 = \text{fdfdf0...ec}$, $c_{12} = \text{c92dc2...e7}$, $c_{13} = \text{85fee5...64}$, $c_{14} = \text{911bc6...04}$
Node p_5	IP $_{p_5} = d_5 = \text{fdfd77...98}$, $c_{15} = \text{b0e79b...68}$, $c_{16} = \text{c6beb7...11}$, $c_{17} = \text{69b9cf...ca}$
Node p_6	IP $_{p_6} = d_6 = \text{fdfd4d...34}$, $c_{18} = \text{babb0a...73}$, $c_{19} = \text{3bc230...c1}$, $c_{20} = \text{1ef42c...d2}$
Node p_7	IP $_{p_7} = d_7 = \text{fdfde0...85}$, $c_{21} = \text{b537ce...76}$, $c_{22} = \text{b5202c...03}$, $c_{23} = \text{8fd67a...fc}$

Table 11: The devices of the moving window rooted at node AA.

For simplicity, data sharing between moving windows is not thematized. p_0 receives the control message, creates the bucket called MW_{AA} and adds its devices d_0 , c_0 , c_1 and c_2 to the bucket MW_{AA} . p_0 then sends this altered control message to p_4 . p_0 registers this reading operation on the moving window by incrementing its read counter by 1. p_4 adds d_4 , c_{12} , c_{13} and c_{14} . p_4 also adds its devices to the bucket MW_{AA} . Since p_4 can restore the devices d_0 , d_1 , d_2 , d_3 , d_4 , d_5 , d_6 and d_7

out of the eight devices $d_0, d_4, c_0, c_1, c_2, c_{12}, c_{13}$ and c_{14} , the control message is instantly sent to p_7 , the target node AA of the Chord node lookup. p_4 also increments its read counter. p_6 would have been contacted if redundancy were not used during the lookup. p_7 can now continue with executing `DoInsert` in the Core services layer.

The messages sent to the other nodes that are responsible for control points residing at level f_{min} are processed analogously. The other nodes execute `DoInsert` and store the devices assigned to the control points the nodes are responsible for if the spatial object intersects more than one area under the area controlled by the other nodes [47]. Otherwise, the spatial object insertion is delegated to one of the four or eight subareas and a counter is increased. Observe that each delegation invokes a full Chord lookup procedure, during which lower number of contacts are observed on average if RSEC is applied. For the example spatial object in figure 41, the spatial object is stored at $G_{eff} = \{AC, AD, CA, CB, AAD, ABC, CCB, CDA\}$ at the end. If there is no control point available, the control point is implicitly allocated with default parameters [48].

A range query is very similar to the insertion procedure. In fact, the method `InitiateRangeQuery` is semantically equivalent to the procedure `InitiateInsertion`. Similarly, the procedure `DoRangeQuery` is very similar to the procedure `DoInsert` in terms of delegation. For a range query, the issuing node determines the nodes responsible for the spatial object at level f_{min} , which delegate the query to their children if the spatial objects crosses divisions. Different to the insertion case, the ultimately responsible nodes - those responsible for areas covered by the range query - intersect the range query window with the MBRs of all locally stored spatial objects. Then, the responsible nodes send all devices pertaining to the intersecting spatial objects back to the issuing node.

The fastest nodes of G_{eff} are sufficient for the complete download of a spatial object, even though the redundancy is $r < 1$. If the speed of light is a root cause in the response delay, then the application of a piggybacking method resulted in the fact that the spatial objects were gathered from the most local nodes in respect to the issuer of the query. Additionally, since moving windows are used during node lookup, the number of contacted nodes during lookup is smaller if there is enough redundancy. The average performance and scalability of the network is increased.

Even if speed of light were not the root cause in the query delay, but any other unknown factor, the piggybacking method in the Core services layer with any $r \geq r_{min}$ or the original method with $r = 1$ are capable of taking advantage of the most favourable route that data takes, oblivious to the myriads of possible causes of delays. The piggybacking method in the Core services layer is capable of maintaining parallelism during object acquisition, different to the original method were it is not guaranteed that every byte received is new knowledge.

At the end of a range search, the spatial objects collected at the issuing node are handed over from the Core services layer to the Application layer. The application, as depicted in figure 40, draws the spatial object on the map. Additionally, the application layer can prune unneeded parts of the spatial object, since the MBR of the spatial object may intersect with the search range, but not the spatial object itself. See figure 42.

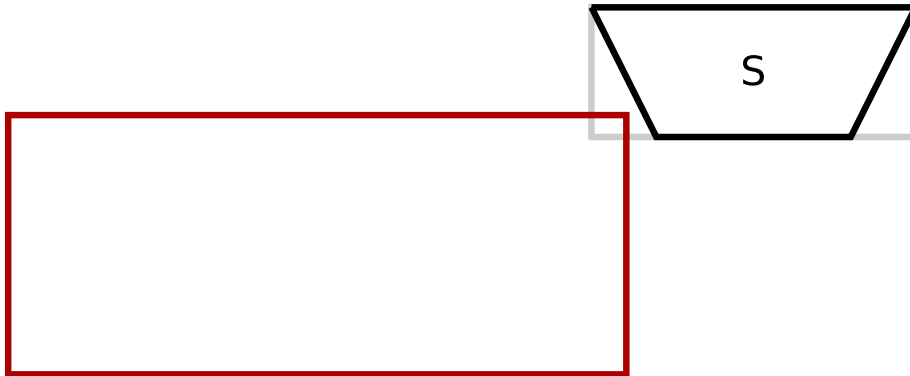


Figure 42: The spatial object S does not take part in the range query (red rectangle), even though the MBR intersects.

3.5 Excursus

3.5.1 Changing the spatial objects in the Core services layer

This section shows the difference between piggybacking a method and increasing the locality of an underlying method, like it is potentially done by scope expansion.

Changes resulting in maximized locality properties are not part of a piggybacking method if the changes do not increase redundancy with the help of erasure coding or if more locality is achieved by replication ($r = 1$).

Assume following change to the method of Tanin, Harwood, Samet et al. [21, 49, 48, 47]: all spatial objects are buffered before being inserted into the index, as shown in figure 43. As a result, G_{eff} will have more control points on average. If the buffer size is at least half the width or height or depth of the area controlled by the control point at the highest level within G_{eff} ($+\varepsilon$), the minimum bounding rectangle or its three-dimensional variant minimum bounding box will always cross a partition boundary. Therefore, the search rectangle can exclude at least one superfluous area controlled by a control point on each of the four sides or eight sides, respectively. For example, the range query in figure 43 can remove the intersection with the left blue rectangle before starting. The disadvantage is that the client has to prune some objects because the tree does not do this job for the client on the level of original MBRs anymore. Ignoring updates, the number of nodes that have to be contacted is lower during a query.

The change does increase redundancy since more replicas are stored in the whole system. But no use of erasure coding is made. Thus, the analysis of the influence of erasure coding on locality properties is not affected.

3.5.2 Synchronizing the f_{min} level in the Core services layer

This section illustrates how the core idea manifests itself in other parts of the data structure. This is done in order to explain better the idea of the thesis. This example shows that RSEC is always applicable since there is always at

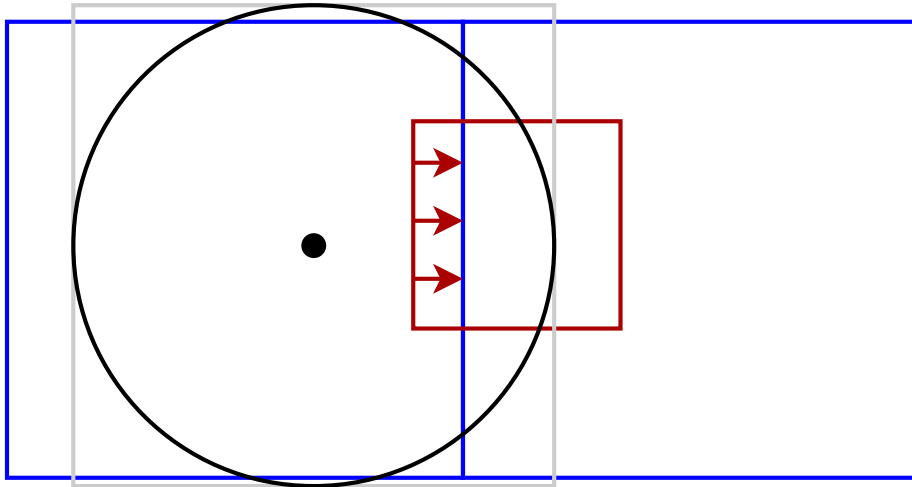


Figure 43: By calculating buffers (circle) with globally known buffer sizes around all spatial objects (point), a range query (red) has to visit less control points on average (areas controlled by control points in blue).

least one group of elements.

The addresses of the peers at level $l = f_{min}$ are synchronized globally and made known to every node in the full redundancy case. For any lower redundancy value, continue as follows: While routing the query to the nodes at level f_{min} , the nodes participating in the Chord lookup method for the particular query attach their device(s) containing information about the globally synchronized state to the query. Each node analyzes whether the query has enough devices and computes the target address from it if possible. If so, the query is redirected directly to the node residing at level f_{min} , underbidding the average Chord routing cost in the average case.

Any update of a node that resides at level f_{min} affects all data devices holding its address, as well as all checksum devices. If there are more control points than peers, the most efficient strategy in terms of contacted nodes would be to send all devices around the ring of Chord. Otherwise, the most efficient strategy regarding parallelism can be applied: updates would be trickled downwards the tree.

Regarding the update through the ring of Chord, the devices can be sent sequentially, which has the effect that updates of the global state are constant background traffic, and are not blocking other operations as burst traffic.

Since the nodes that are responsible for the control point on level $l < f_{min}$ are idling on the Core services level, a combination of the ring-update and tree-update makes sense. The global state is synchronized amongst the nodes responsible for level $l < f_{min}$ by sending all devices at once and trickling down the tree. Then, each node injects the update into the ring of Chord by sending device after device to its successor. Assuming random distribution of the nodes at level $l = f_{min} - 1$ on the ring of Chord, the procedure is accelerated from contacting $O(N)$ nodes sequentially to contacting $O(N)$ nodes in parallel by the factor $4^{f_{min}-1}$ in the case of a Quadtree and $8^{f_{min}-1}$ in the case of an Octree. Every node participating in the distribution knows whether it is responsible

for a control point at level $l = f_{min} - 1$ or not and can stop the distribution accordingly.

3.6 Interim conclusion

The whole section 3 has enabled us to answer the first main question of this master thesis:

Is it possible to reasonably apply redundancy (II) onto the principle of hierachization ($I\alpha$) or onto the principle of distributed hash tables ($I\beta$)?

The answer to this question is “yes” because I successfully applied redundancy in two cases. Redundancy has been applied to a method organizing spatial objects along a hierarchy ($I\alpha$), namely to the Quadtree in the Core services layer. Additionally, redundancy has been applied to a method organizing data along a line ($I\beta$), namely to Chord in the Connectivity layer.

The relevant parameters for the cost model introduced in section 3.1 are now specified in detail in sections 3.4.1, 3.4.2 and 3.4.3, and, as a matter of fact, can be used by the cost model for the calculation of the profitability of my piggybacking method in relation to the original method in question.

4 Analysis

4.1 The cost model

	No optimization:	This thesis:
History h:	R R R R U	R R R R U
Contacted nodes:		
Profit:		1 1 1 1 -2
Overall profit p(h):		2

Figure 44: The cost model for $|R| = 4$, $|U| = 1$, $a = 3$, $b = 1$, $a' = 2$, $b' = 3$, $h = \{R, R, R, R, U\}$. $p(h) = 4 \cdot (3 - 2) + 1 \cdot (1 - 3) = 4 \cdot 1 + 1 \cdot -2 = 4 - 2 = 2$

In order to calculate whether the application of redundancy is worthwhile in comparison to the original method, a cost model that deals with history h was conceptualized in section 3.1, and following formula was derived:

$$p(h) = |R| \cdot (a - a') + |U| \cdot (b - b') \quad (10)$$

The context of the cost model is always in relation to the group, that is the set of elements P .

a' is dependent on the amount of redundancy r . In the course of the method, the parameters of this formula were elaborated: r and z .

r was modified according to the specific application of redundancy. z is the number of contacted nodes for a given amount of redundancy. A new r , which is derived from the profit, is dependent on z itself. I deal with this recursion reasonably by setting the default value $\frac{|R|}{|h|}$.

In relation to the application of redundancy, following three formulas for r ensued: for the general application of redundancy (section 3.4.1), for the application of redundancy to Chord (section 3.4.2) and for the application of redundancy to the Quadtree (section 3.4.3).

- For the general case (formula 6 on page 55):

$$r = \begin{cases} 1 & \text{if } updateCost \leq |R| \text{ or } readProfit \leq |U| \\ \frac{|R|}{|h|} & \text{otherwise} \end{cases}$$

- For the application of redundancy to Chord (formula 8 on page 57):

$$r = \begin{cases} 1 & \text{if } p - 1 \leq |R| \text{ or } \lceil \alpha_{Chord} \log p \rceil - z + 1 \leq |U| \\ \frac{|R|}{|h|} & \text{otherwise} \end{cases}$$

In this section, z was introduced (formula 7 on page 57):

$$z = \left\lceil \frac{p}{\left(r + \frac{1}{p-1}\right)(p-1)} \right\rceil$$

- For the application of redundancy to the Quadtree (formula 9 on page 62):

$$r = \frac{|R|}{|h|}$$

In the following analysis, calculations are made with these specified listed formulas.

Regarding a , b , a' and b' , observe that the average number of hops required to reach the target node t_C from any node on the identifier space is $d(\{t_C\}) = \alpha_{Chord} \log N$ [44]. The average number of hops required to reach the $|G_{min}|$ nodes at level f_{min} is $d(G_{min}) = |G_{min}| \cdot \alpha_{Chord} \log N$ hops on average. α_{Chord} is 0.5 and is derived from the fact that the distance only reduces the maximal possible distance with probability 0.5 [44].

Finding p participants in a moving window on the Chord DHT always costs $d(\{p_0, p_1, \dots\}) = p - 1 + \alpha_{Chord} \log N$ hops, since the predecessor pointers and successor pointers can be used. Consistency maintenance of the neighbouring pointers is assumed to be a problem of the lower layer and does not affect the cost model.

4.2 Analyzing the profitabilities of the redundancy applications

4.2.1 Relation to caching

Reed-Solomon erasure coding (RSEC) exhibits similarities to caching. Caching is done by storing copies of acquired data. For example, Globe [51] makes use of cache pointers, or Tanin, Harwood, Samet et al. [49, 48, 47] cache the pointers to the children. Tanin, Harwood, Samet et al. [21, 49, 48, 47] cache the full spatial object on every responsible node.

Caching only certain data devices on other nodes results in the following problem similar to the Coupon collector's problem: An agent such as a lookup query needs to gather information. Each time a random node is visited, data devices are gathered. The probability of obtaining a new data device belonging to the same group is getting smaller and smaller with more and more data devices already in memory. For example, assume that d_0 and d'_2 from figure 31 on page 53 are gathered. Obtaining d_2 afterwards is a waste of resources, because d'_2 is a copy of d_2 with the same content.

RSEC, on the other hand, always contributes to the local knowledge. This results in the property that the number of nodes that have to be contacted is always minimal and constant in relation to redundancy r . Using only caching results in a constantly higher number of nodes that have to be looked up on average, and this number distributes around the mean since device gathering is subject to chance. Still, it is possible to achieve to contact a sequence of nodes so that only distinct data devices are gathered. Such a special case is less and less probable if the number of possible nodes to contact is increased.

As a matter of fact, RSEC allows to effectively reduce the dependence on locality while caching does not. Whatever the spatial location of the elements, choosing any most local locations containing n elements out of $n + m$ satisfies the request. Caching can not guarantee that, unless $r = 1$. Of course, in the

case of caching the agent may choose freely between d_0 and d'_0 . But still, all objects such as d_0 , d_1 , and d_2 need to be resolved. For an illustration see figure 32 on page 54. The improvement of RSEC over caching is the freedom to choose any $r_{min} \leq r \leq 1$ without affecting the freedom to choose any subset out of a set, leading to the answer to the first research question:

Research Question 1: Does the introduction of redundancy result in a lower number of contacted nodes during read operations than without? Is copying elements of the set to multiple participating nodes more efficient in terms of contacted nodes than Reed-Solomon erasure coding?

Answer to Research Question 1: Redundancy, as applied by RSEC, results in a lower number of contacted nodes during read operations as soon as the redundancy is $> 0\%$. The worst case of RSEC regarding the number of contacted nodes is better than the worst case of caching if the redundancy is $> 0\%$ and equal if full replication is employed. In the best case, both are equal. In the average case, RSEC performs better.

The difficulty lies in finding candidate sets. For example, Tanin, Harwood, Samet et al. [21, 49, 48, 47] provide two sets: the set of spatial objects at a node responsible for a control point and the parts of a spatial object. The devices provide the flexibility to download from the fastest nodes in conjunction with RSEC in the context of the second modification (section 3.4.3). But a range query still has to contact all nodes covered by the range query. Reducing the locality regarding the search itself can be achieved with the help of the modification to the spatial objects in the Quadree or Octree, respectively (section 3.5.1). But this has nothing to do with the effect of erasure coding on locality.

The method altering Chord (section 3.4.2) has to define a custom set P in order to make an adaption of RSEC possible.

4.2.2 Altering the Core services layer by introducing redundancy

This modification is the example case for analyzing the effects of erasure coding on locality without any external effects. The analysis differs from the adaption to Chord therein that the number of contacts during updates are the same for both cases. Furthermore, the analysis is not affected by scope expansion. This analysis is therefore solely affected by erasure coding.

Lemma 1. *In an environment without updates ($\frac{|R|}{|h|} = 1$), the application of this modification with $r = 1$ results in the same number of nodes that have to be contacted as without this modification.*

Proof. Tanin, Harwood, Samet et al. [21, 49, 48, 47] implicitly set $r = 1$. For their approach and the approach of this modification, only one node needs to be looked up in order to obtain the full information case regarding a spatial object, since $r = 1$ is full replication.

Lemma 2. *In an environment without updates ($\frac{|R|}{|h|} = 1$), the application of this modification results in gradually more freedom to choose from the nodes responsible for $|P| = |G_{eff}|$ with increasing redundancy.*

Proof. Any node can be contacted because any n rows can be selected from the Vandermonde matrix. Regarding the redundancy: the minimal number of nodes that have to be contacted is $z = \left\lceil \frac{|P|}{(r + \frac{1}{|P|-1})(|P|-1)} \right\rceil$. $z \approx |P|$ for $r = 0 + \varepsilon$ and $z \approx 1$ for $r = 1 - \varepsilon$. ε in this case is $\varepsilon > 0$, and $\frac{1}{\varepsilon} \gg 1$.

Lemma 3. *In an environment without read operations ($\frac{|R|}{|h|} = 0$) and $r > 0$, the application of this modification results in the same number of nodes that have to be contacted as without this modification.*

Proof. In the case of the method of Tanin, Harwood, Samet et al. [21, 49, 48, 47], an update affects all replicas. In the case of this modification, an update must be represented in all checksum devices, since the checksum devices are dependent on all data words. The checksum devices are distributed evenly amongst the participants. Therefore, all nodes need to be contacted in both cases.

In the case of $r = 0$, only nodes responsible for affected data devices would need to be contacted. $r = 0$ in the original method of Tanin, Harwood, Samet et al. [21, 49, 48, 47] would result in the fact that no hit is superfluous. But the original method only supports $r = 1$.

Lemma 4. *Applying this modification never results in a loss according to the cost model in relation to the method of Tanin, Harwood, Samet et al [21, 49, 48, 47].*

Proof. Consider $p(h) = |R| \cdot (a - a') + |U| \cdot (b - b')$. Updates contact the same number of nodes in both cases ($b - b' = 0$). The right part of the equation is therefore always zero. Assuming the worst case of $r = 0$, $a - a'$ becomes zero, too. Therefore, it is impossible to reach a negative value.

Corollary 1. *The value of r only influences the amount of profit in relation to the method of Tanin, Harwood, Samet et al [21, 49, 48, 47], if the same redundancy is used for comparison. $p(h) \geq 0$. Maximal profit is achieved by setting $r = 1$.*

Proof. It is assumed that $r < 1$ is achieved by copying parts of the spatial object to participating nodes in the original method (called caching). $r = 1$ results in a minimal b' and a maximal $b - b'$. Since the right part of the equation 10 on page 72 is not affected by r , maximal profit is achieved by setting $r = 1$. Since additional storage costs are not represented in the cost model, Tanin, Harwood, Samet et al. [21, 49, 48, 47] with $r = 1$ are optimal in this regard. Otherwise, it is not guaranteed that newly gathered knowledge is new knowledge, since caching as alternative is assumed in the original method if $r < 1$.

Lemma 5. *This modification may never reach the maximal theoretical profit.*

Proof. If the last elements in h are read operations, this modification can not adapt to a new redundancy. This modification is always lagging behind and reactive, never proactive. Consider the worst case of $|h|$ update operations, followed by $|h|$ read operations, alternating eternally. The profit always stays zero, which is suboptimal. This pathological case is, however, unlikely.

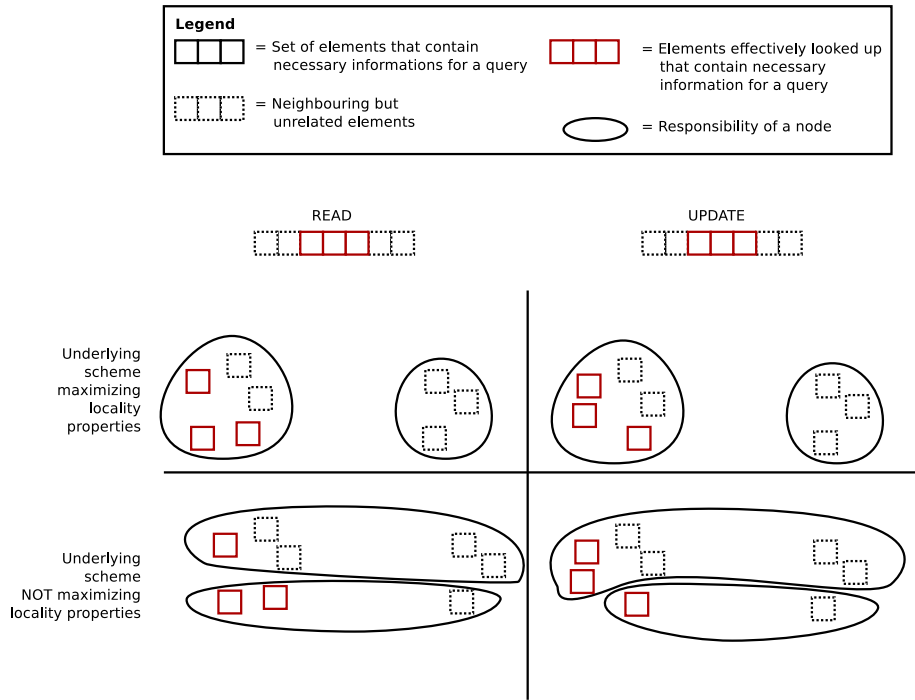


Figure 45: Not introducing RSEC to the set of elements commonly accessed together ($r = 0$).

Lemma 6. *The profit from read operations is as global as the task of maintaining consistency.*

Proof. The number of hops is at least $d(P) = |P| \cdot \alpha_{Chord} \log N$ in both cases.

Corollary 2. *Even though a minimal amount of nodes need to be contacted for the download of a spatial object if RSEC is profitable, the number of initial contacts to nodes responsible for control points stays the same.*

Proof. Due to the way the method of Tanin, Harwood, Samet et al [21, 49, 48, 47] works, namely due to the use of parallelism with non-communicating search instances, it is not possible to effectively reduce the number of contacted nodes for a given range query. It is impossible to reduce the number of contacted nodes for a given range query since every node may be responsible for a spatial object that is only stored on this single node. Still, downloads of the spatial objects are fast since the fastest nodes can be used due to the properties of RSEC. Additionally, only new knowledge is downloaded.

The rest of this section summarizes some insights because all lemmas are stated.

Consider figure 45. All nodes need to be contacted in order to obtain all data devices. Now consider figure 46. Introducing RSEC has alleviated the need to contact all nodes responsible for a spatial object during download. As a consequence, stragglers as defined by Halbawi et al. [20] are avoided. Final

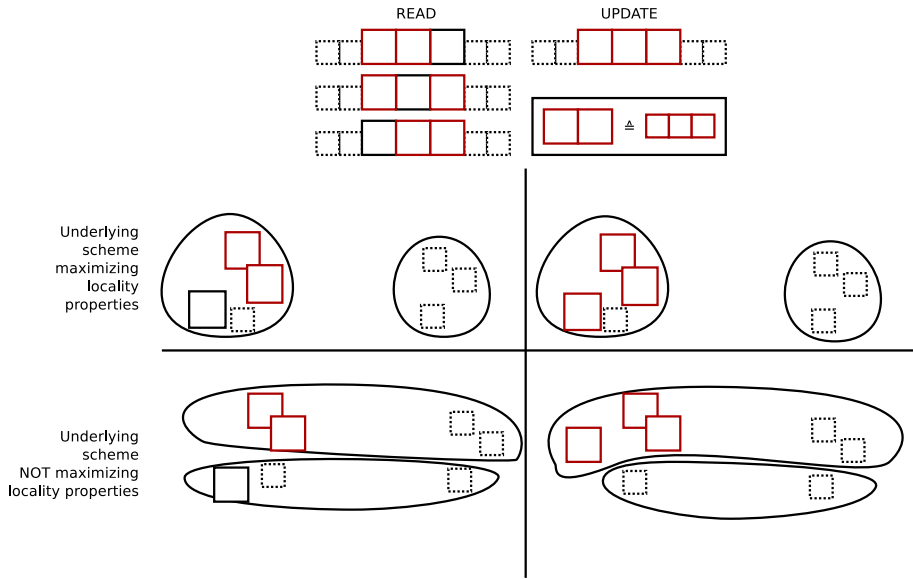


Figure 46: Effects of increasing the redundancy.

query delay is effectively less dependent on bandwidth-starving nodes. During updates, the whole set has to be contacted. In the worst case of $r = 0$, the number of nodes that have to be contacted stays the same. The profit of the cost model is never negative. Answering research question 2:

Research Question 2: Assuming the method of Tanin, Harwood, Samet et al [21, 49, 48, 47] as an underlying method with nodes in a set P , each holding one copy of a spatial object: Does the introduction of RSEC-calculated devices over the nodes in P result in less contacts for a given history h of read operations and update operations than without?

Answer to Research Question 2: The number of contacted nodes stays the same for both the underlying method and the piggybacking method due to parallelism. Regarding the number of contacted nodes during the download of a spatial object: There is no profit and no loss in an update-only environment in relation to the method of Tanin, Harwood, Samet [21, 49, 48, 47], as discussed in lemma 3. In all other cases, applying RSEC with redundancy $> 0\%$ always results in profits in relation to the method of Tanin, Harwood, Samet [21, 49, 48, 47] according to the cost model and if the same level of redundancy is used for the comparison, as it is shown by corollary 1. Applying RSEC in relation to the method of Tanin, Harwood, Samet [21, 49, 48, 47] never results in a loss according to the cost model (lemma 4).

4.2.3 Altering Chord in the Connectivity layer

As the modification to the Chord lookup method is on a lower layer than the Core services layer, the changes are affecting all queries equally. The intention of the modification was not only to show how overlaying redundancy and how

scope expansion works, but also in order to analyze whether addition of erasure coding to a data structure dictated by hashes results in tangible improvements regarding the number of nodes that need to be contacted. Hashes have the original intention to destroy any similarity between the input and output, called avalanche effect. The sets of commonly accessed elements is random, since use of finger tables is made and random distances occur. Therefore, the analysis analyzes what the gains or losses from the embedding of RSEC to a data structure are. The advantages from RSEC are only practically usable if RSEC deals with real-world constraints that stem from the adaption of the data structure to support RSEC.

Lemma 7. *Fitting the modification to Chord, i.e. $L = R_{win} = 1$, does not result in less contacted nodes.*

Proof. The lookup will never read the third element. If the middle element is in a lookup table, the lookup has succeeded. For the last element, the leftmost one, assume the best case with full redundancy $r = 1$. Even with the knowledge of the whole moving window, the last step to the target node has to be made. Therefore, the number of contacted nodes is the same. Actually, the node at the leftmost element already has a successor pointer to the target.

Corollary 3. *The R_{win} successor nodes of the owner of the moving window are not contacted during Chord lookup of the owner node.*

Proof. Lookups are dictated by the finger tables at each node [44]. The search algorithm reduces the distance to the target by redirecting the lookup message to the nearest node in the finger table that is preceding the target node. On a number line from left to right, the search never goes past the target node.⁷

Lemma 8. *Assume scope expansion and a constant s during the lookup of a node ($s \geq 4$, $R_{win} = 0$). The number of nodes that have to be contacted on average are lower than without this modification.*

Proof. This is achieved on average by setting the redundancy to $r = 1$, i.e. by employing full redundancy. Assume that the elements d_0 , d_1 and d_2 in figure 31 are the last three information pieces that had to be acquired in order to successfully find the responsible node for a given key. By applying this modification, knowing one element of the moving window is enough in order to jump to the target. This is the case if d_0 , d_1 and d_2 are parts of a moving window with $s \geq 4$. For the case of $s = 4$, $R_{win} = 0$, if the search hits the leftmost element, the lookup of the predecessor to the target is spared. All other elements have finger pointers to the target. This explains why $s < 4$ does not result in an improvement: the finger tables of all participants are already pointing to the target. The pathological case for any $s \geq 4$ always hits nodes with direct finger pointers to the target, and does not result in a lower number of contacted nodes on average. But the pathological case does not happen with high probability, especially not on the average case.

Corollary 4. *The number of nodes that have to be contacted on average are lower than without this modification if there is at least one element in the moving window that does not contain a finger table entry pointing to the target. Any moving window holds an entry having this property with size $s \geq 4$.*

⁷This is why $R_{win} = 0$ is chosen.

Proof. See proof from previous lemma 8.

Corollary 5. *In a environment without updates ($\frac{|R|}{|h|} = 1$) and a fixed $s = 4$, the application of this modification with $r = 1$ results in a lower number of nodes that have to be contacted than without this modification.*

Proof. This is the full redundancy case mentioned by Plaxton [33], if only for each and every moving window. Setting $r = 1$ in such a situation is the best choice in relation to the available information encoded in h .

Corollary 6. *In a environment without updates ($\frac{|R|}{|h|} = 1$) and a fixed $s = 4$, the application of this modification with $r = r_{min} = \frac{1}{p-1} = \frac{1}{3}$ does not result on average in a lower number of nodes that have to be contacted than without this modification.*

Proof. It is needed to prove that all possibilities result in the same number of nodes that have to be contacted. If the only case resulting in a lower number of contacted peers from corollary 5 is analyzed, it is shown that the number of contacted peers is the same. The lookup message visiting the leftmost node in the moving window gathers two devices: d_0 and c_0 . Then, the routing reduces the distance by 2, 4 would overshoot the target. Additional devices d_2 and c_2 are gathered. It is not needed that the moving window is recovered, since the successor to the node in question is the target. All other nodes have finger pointers to the target.

Corollary 7. *In an environment without updates ($\frac{|R|}{|h|} = 1$) and a fixed $s \geq 4$, the application of this modification with $r = r_{min} = \frac{1}{p-1}$ never results in a lower number of nodes that have to be contacted than without this modification.*

Proof. Assume the worst case of a lookup always decreasing the maximal distance to the target. The minimum redundancy required for jumping to the target is always $r > r_{min}$, as listed exhaustively for $s = 17$ in figure 47. More generally, it is impossible to gather s devices if there are only $\log s$ visits in the moving window if they add only two devices each. $\frac{\lceil \frac{n - \log n}{\log n} \rceil}{n-1} > \frac{1}{n-1}$.

Lemma 9. *Assume an environment with equal number of reads and updates involving a certain moving window ($\frac{|R|}{|h|} = 0.5$), $s = 9$, $b = 1$ and $b' = 9$. With $r = 0$, the application of this modification results in a larger number of lookups on average.*

Proof. With equal $|R|$ and $|U|$, $p(h) = a - a' + b - b'$. $a = \lceil \alpha_{Chord} \log s \rceil = \lceil 0.5 \cdot \log s \rceil$ in the average case: finger tables are effective in reducing the distance and do not contact many nodes near the target, as shown in figure 47 for a certain case ($s = 17$) [44]. Assuming $r = 0$, $a' = a$. Since $b < b'$, this modification does not improve the number of nodes looked up.

Lemma 10. *Assume $b = 1$ and $b' = 9$. With $r = 1$, the application of this modification does not result in a larger number of lookups on average as soon as $|h|$ consists of $(8) \cdot |R|$ read operations and $(\lceil 0.5 \log s \rceil - 1) \cdot |U|$ operations.*

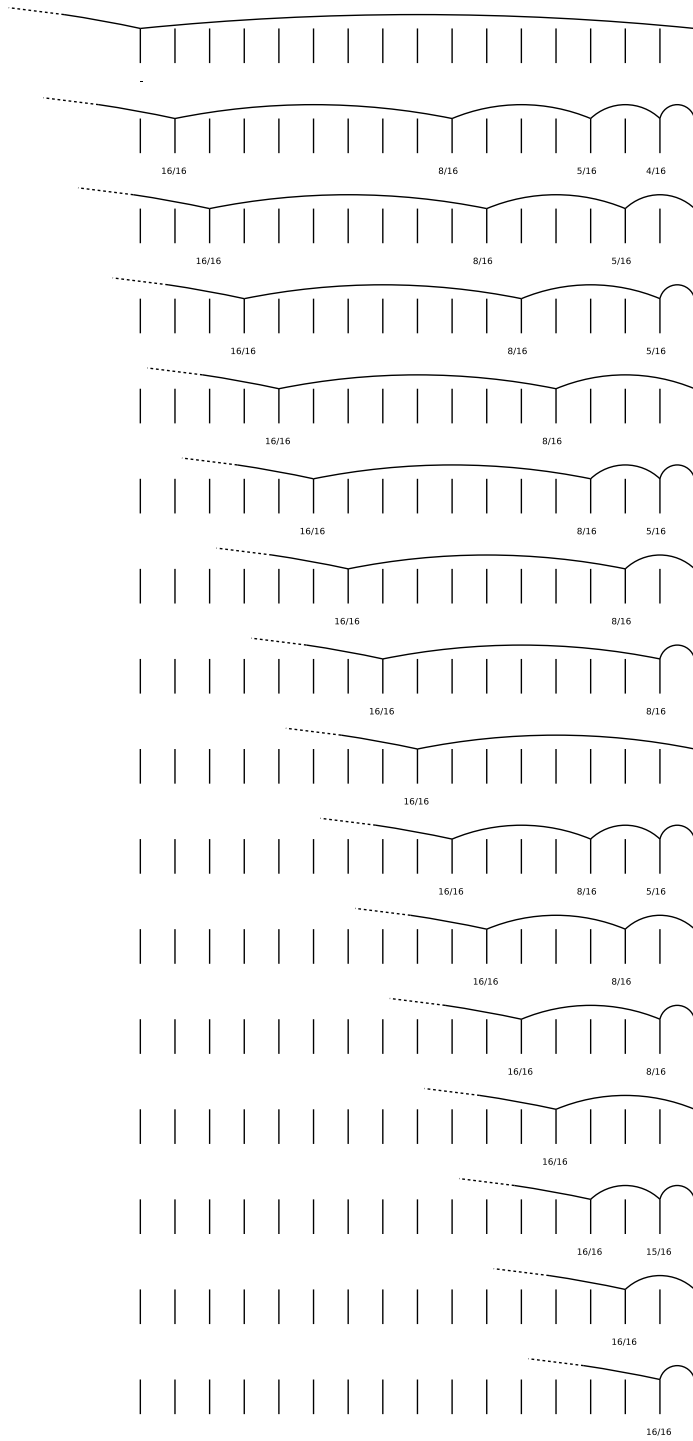


Figure 47: The required redundancy r in the moving window, if skipping the rest has to be supported, is shown. In this case, $n = 17$ devices need to be gathered. δ visits gather $|D| + |C| = 1 + \lceil \frac{n-\delta}{\delta} \rceil$ devices at each node, resulting in $\frac{|C|}{n-1}$ required checksum devices for a given position in the figure.

Proof. With equal $|R|$ and $|U|$, $p'(h) = a - a' + b - b'$. From above, $a = \lceil \alpha_{Chord} \log s \rceil = \lceil 0.5 \cdot \log s \rceil$ in the average case. Assuming $r = 1$, $a' = 1$. One read operation results in $a - a' = \lceil 0.5 \log s \rceil - 1$ nodes that do not have to be contacted. One update operation results in $b' - b = 8$ nodes that have to be contacted additionally, the profit is $b - b' = -8$. The number of contacted nodes over history h is equivalent if $b' - b = a - a'$, or $updateCost = readProfit$. For general $\frac{|R|}{|h|}$, the number of contacted nodes is equivalent if

$$updateCost \cdot readProfit = readProfit \cdot updateCost$$

$$updateCost \cdot |R| = readProfit \cdot |U|$$

$$(b' - b)(a - a') = (a - a')(b' - b)$$

$$8 \cdot (\lceil 0.5 \log s \rceil - 1) = (\lceil 0.5 \log s \rceil - 1) \cdot 8$$

The last equation holds.

Corollary 8. *Selecting a lower redundancy than $r = 1$ requires a higher number of reads in h for $p(h) \geq 0$ than stated in lemma 10 or lemma 11, respectively.*

Proof. a' with $r = 1$ is $a' = 1$. a' with any r is

$$a' = z - 1 = \left\lceil \frac{s}{(r + \frac{1}{s-1})(s-1)} \right\rceil - 1$$

$$a - a' = \lceil \alpha_{Chord} \log s \rceil - \left\lceil \frac{s}{(r + \frac{1}{s-1})(s-1)} \right\rceil + 1 = \lceil \alpha_{Chord} \log s \rceil - z + 1$$

$a - a'$ affects $p(h)$, and $a - a'$ is lower if r is lower. A $r < 1$ requires $|R|_{new} > |R|_{old}$ in the history in order to continue to reduce the total number of contacted nodes. In order to reach a positive $p(h)$, a high redundancy is optimal if there are only slightly more read operations than needed. Additionally, if there are not enough $|R|$, or if r is chosen too low, lemma 9 is repeated: the total number of node contacts is not reduced.

Lemma 11. *If h has at least the needed share of operations (lemma 10), $p(h) \geq 0$. This modification reduces the number of contacted nodes during Chord lookup on average if the history has at least $(s - 1)$ read operations for every $\lceil \alpha_{Chord} \log(s) \rceil - z + 1$ update operations. The share of updates in the history should not exceed $\frac{\lceil \alpha_{Chord} \log s \rceil - z + 1}{|h|}$.*

Proof. See lemma 10 and corollary 8. The redundancy r is adapted to this identified worst case, see equation 6. This modification sets $r = 1$ proactively, that is, even though a smaller share would have profited.

Lemma 12. *This modification provides advantages within stated limits (see lemma 11) if the scope is global, and exhibits disadvantages if the scope is local. Maintaining consistency is a local task and only affects neighbours.*

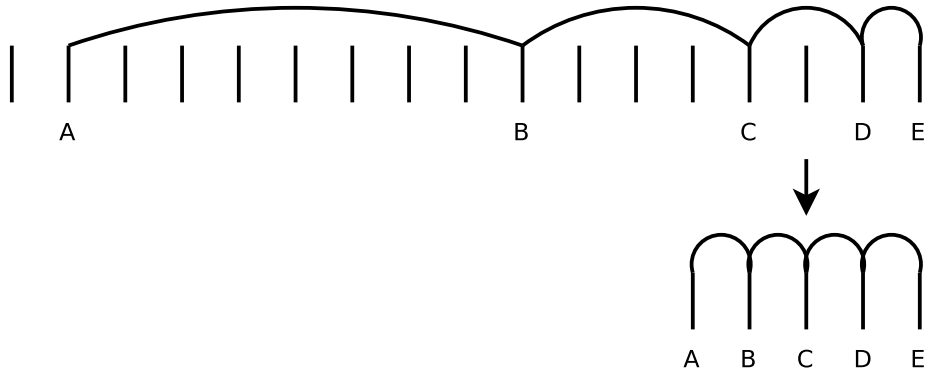


Figure 48: If the elements of the moving windows were increasing exponentially in distance and the search distance only contained binary ones, such as 11111_2 , it would have been known what is accessed together. If all queries arriving at node E used the same hops, only redundancy would have been applied. But then, the set of nodes that can make use of RSEC is limited to A, B, C and D. Even worse, in the case of same number of reads and updates, more elements need to be found during updates if A never looks up E using the Chord method.

Proof. The cost of contacting the nodes in a moving window during an update is not $d(P) = |P| \cdot \alpha_{Chord} \cdot \log N$ hops on average, but $d(P) = |P| - 1 + \alpha_{Chord} \cdot \log N$ hops.

The rest of this section summarizes some insights because all lemmas are stated.

In relation to the method of Tanin, Harwood, Samet et al. [21, 49, 48, 47], applying RSEC proves to never have loss in relation to replication if the set of elements accessed together is stable (lemma 4). In a randomized lookup protocol, it is difficult to identify such sets of elements. This modification has to deal with the situation that the elements that are accessed together during a lookup are nearly unknown. It is only known that there will be small jumps just in front of the target. This situation differs from the modification affecting redundant spatial objects (section 3.4.3), where the nodes that will be contacted for the download of a spatial object are known and listed in P . If the distance from the initiator to the target were always consisting of ones in the binary representation (11111_2), it would have been possible to adapt to such lookups to a certain length, as shown in figure 48. As this is not the case, the modification can only capture the lookup at the end of the search.

The entries in finger tables are elements that are accessed together, but applying RSEC to this set makes no sense since the set only covers one node.

Different to the redundant spatial object case, the number of contacted nodes is unbalanced in the case of same number of reads and updates, see lemma 10. This property of the underlying method affects the properties of the piggybacking method. Consequently, applying RSEC only makes sense if history h has a certain share of reads and updates. It is possible to mitigate the damage by increasing the redundancy, which lowers the number of contacted nodes during reads, resulting in a larger profit which is competitive to the number of ad-

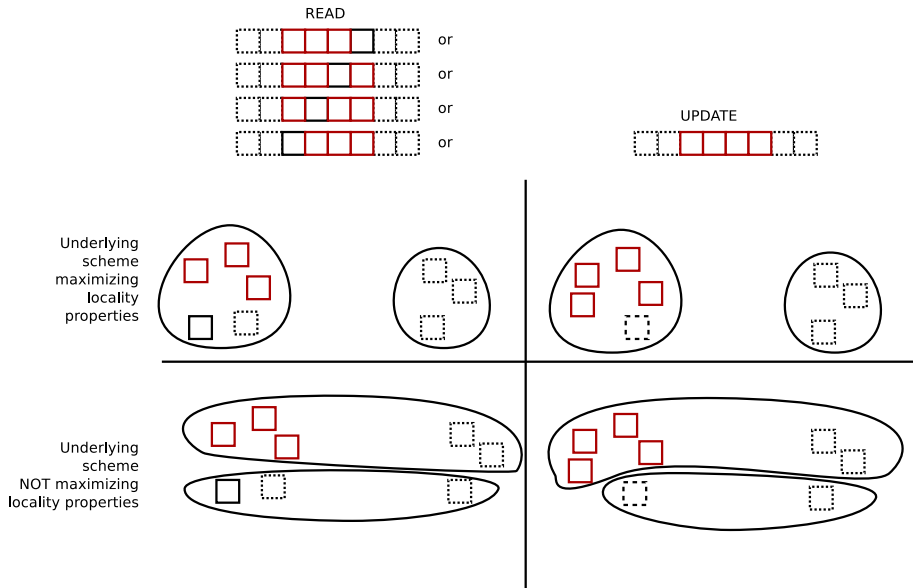


Figure 49: Effects of scope expansion. p was originally 3. This modification to Chord selects neighbours as participants in a set. Consequently, updates stay local in the identifier space.

ditional contacted nodes during updates. Otherwise, the modification is only profitable according to lemma 11.

If no scope expansion is performed, the worst case of an environment, where only updates exist, does not alter the dependence on locality ($b = b'$). If scope expansion has to be performed, this must not hold, but since maintaining consistency can be done locally, the costs are not as high as one may expect.

Applying RSEC can contribute to a lower number of contacted nodes during Chord lookup. At least β nodes do not have to be contacted, β being dependent on r and, subsequently, z . Therefore, Chord lookup efficiency is $O((1 - \frac{\beta}{\log N}) \cdot \log N)$ if RSEC is profitable.

Research question 3 can now be answered:

Research Question 3: How is the exact maximal share of update operations of a given history h determined so that the introduction of RSEC is still worthwhile, even if the identified set has been expanded and is distributed over more nodes than before?

Answer to Research Question 3: This is done by analysing the update cost and read profit for the operations of the piggybacking method according to the cost model. Afterwards, this is done by requiring at least as much reads as the value of the update cost and allowing maximally as much updates as the value of the read profit (lemma 10). The shares are then normalized by $|h|$ (lemma 11). If the observed share of update operations in h grows beyond the allowed share, the application of RSEC with scope expansion is not worthwhile. Regarding the exact value: Chord lookup is a randomized protocol. Therefore,

the maximal share is based on the average case. Still, the redundancy is adapted to lemma 10, which analyses the costs. This allows to guarantee that this modification is not worthwhile as soon as the share is overstepped in h .

4.3 Analyzing excursus

4.3.1 Synchronizing the f_{min} level in the Core services layer

Largely, the addition of redundancy as proposed in the section 3.5.2 makes only sense for read queries, as the costs for updates are prohibitive except for special cases.

The parameter f_{min} is paramount in determining whether this modification is an improvement. If f_{min} is small, the cost of updates is reduced since less updates can be expected. Increasing f_{min} trades available bandwidth rapidly if this modification is used. The delay time of sending the next device is affecting the bandwidth used per time interval. Economic use of the bandwidth is made by delaying the synchronization of the global state until an update has crossed the whole ring. Still, in the majority of cases, the global synchronization makes no sense since the required redundancy has to be at least $\frac{dim \cdot f_{min}}{O(\log p)} \cdot 100\%$ (p =number of participants, dim =dimension 4 or 8). f_{min} being close to zero is the special case where this modification reduces the difference of costs between read and updates enough to be paying off.

4.4 General reflections

4.4.1 History h

The size of the history must adhere to two requirements. First, the history must represent enough events in order to estimate the proportion between read operations and update operations. Secondly, it must not be too long, as it must cope with recent developments. The choice of $|h|$ does not affect much since equation 6 tries to optimize for storage that is not included in the cost model. Still, there are some thoughts that need to be communicated:

It is impossible to define any optimal length in relation to the future as the history is an online statistic. The future is not ours to see.

The number of accesses is balanced over history h if $p(h) = 0$. If the proportion between optimal number of reads and the optimal number of updates can not be represented, then $|h|$ is too small. For example, using the values from lemma 11, $s - 1$ reads are opposed to $\lceil \alpha_{Chord} \log(s) \rceil - z + 1$ updates. Reflecting the share is therefore possible if $|h|$ is at least $s - 1 + \lceil \alpha_{Chord} \log(s) \rceil - z + 1$. For $s = 9$, the working number $|h| = 50$ of this thesis is a conservative approach.

Too long histories may not adapt to a new situation. The impact could be mitigated by limiting the length of the history to a maximal age. But by introducing this requirement, all participants need to have their clocks synchronized. While this proves to be an avenue for further research, the scope of the thesis remains on determining whether the method makes sense in terms of number of contacted nodes or not. More accurate prediction only broadens the restrictions on the share of reads and updates, and does not alter the conclusion that applying the method can reduce the number of contacted nodes, even in the light of updates.

If the set of elements listed in h is expanded, more accurate adaption is possible. For example, JUMP could be introduced. JUMP would state that the node could jump to the full information case because enough devices were available for successful reconstruction. Again, better adaption only broadens the restrictions and the conclusion of this thesis is not affected.

4.4.2 Parallelism

Parallelism defies the use of redundancy. If a collected information piece does not increase the information available to a common pool, but to an isolated one, the advantages of RSEC are restricted to each isolated pool.

Still, especially in the case of the method of Tanin, Harwood, Samet et al. [21, 49, 48, 47], using RSEC brings advantages. The nodes that would cause the highest number of Chord lookups are likely the last to respond to the issuing node and are not needed as soon as enough devices are collected.

4.4.3 Effects of data sharing between moving windows on address discovery

Data sharing between moving windows is possible since each node attaches devices of all moving windows, of which the node is participant of, to messages. As a matter of fact, situations such as in figure 37 allow for an improvement over the worst case that was analyzed. The worst case is the case where only information from a single moving window is available. The net effect is that contacting one node results in the knowledge of a moving window of size $s' = s + r \cdot L + r \cdot R_{win}$.

Even though the change would alter the needed ratio of read operations to update operations for $p(h) \geq 0$, this more accurate prediction would not alter the conclusion that applying the method of this master thesis can reduce the number of contacted nodes, even in the light of updates.

4.4.4 Effects of RSEC on overall performance

Overall performance is dependent on a large number of variables that are a candidate for being optimized. Query delay, data transfer costs, connection establishment costs, per-query network traffic (M in [9]) and even message distribution are only a few. My cost model only captures the number of contacted nodes since it is assumed that the connection establishment delay is significant in relation to subsequent data transfer time costs. As a consequence, the aim is for a lower number of affected nodes during a query. According to Blanas et al. [9], average per-query network traffic is reduced most if the query affects as few nodes as possible. As a consequence, a smaller number of contacted nodes is preferable over more even message distribution as the network scales [9].

Regarding total amount of data: RSEC introduces redundancy in order to reduce the number of contacted nodes if possible. This is a tradeoff. The network usage and storage usage finally dictate whether RSEC should be favoured or not. If network congestion occurs, applying RSEC can be favourable since less nodes are contacted. If storage space is the limiting factor, storing additional redundancy makes no sense.

If the case of $r = 1$ were prohibitive, then the method of Tanin, Harwood, Samet et al. [21, 49, 48, 47] would be prohibitive, too. Applying RSEC to

Chord is not prohibitive regarding the amount of data since the data devices occupy only a few bytes.

Summarizing, whether to apply RSEC or not depends on the limiting factors of the applied system. As these factors are considered to be unknown, no general recommendation regarding the usage of RSEC can be given, even though advantages in exactly defined situations are derived analytically in this thesis.

4.5 Conclusion

Regarding the download of spatial objects, applying RSEC does not result in additional costs in relation to the method of Tanin, Harwood, Samet et al. [21, 49, 48, 47]. If scope expansion is done in order to piggyback RSEC to an underlying method, the number of nodes contacted is always lower if there are only reads. If there are additionally updates, the share of update operations must not be larger than the share of read operations over a history h . For the introduced modification the share of updates is $\frac{[\alpha_{Chord} \log(s)] - z + 1}{|h|}$ and the share of reads is $\frac{s-1}{|h|}$ (s being the window size). Even though RSEC increases the number of contacted nodes during updates, scope expansion continues to be affordable if the restrictions on h are met.

The introduced modification affecting the download of the spatial objects shows that there is never a loss in terms of number of contacted nodes in relation to the method of Tanin, Harwood, Samet et al. [21, 49, 48, 47], even if the task of maintaining consistency is included [33]. This applies only if the same redundancy is used in both the underlying method and the piggybacking method. Actually, the application of RSEC results in the freedom to use more granular redundancies.

The number of links per node stays the same for the modification introduced in section 3.4.3. The modification affecting Chord has different properties. The number of links per node is not the original $O(\log N)$ [44], but is increased by $2 \cdot s - 1$ to $O(\log N + 2 \cdot s - 1)$, because the node is additionally part in moving windows to the left and to the right. This is if the predecessor and the successor are ignored since they belong to the lower layer. Otherwise, the number is increased by $2 \cdot s - 3$.

The analysis focussed on worst cases if possible. I expect that the advantages are even greater than this analysis has documented.

With the contribution of this analysis (section 4), it has become possible to solve the second main problem of this master thesis, this is how the benefit of an application of redundancy (II) is calculated and assessed. ⁸

⁸In order to answer this main question, section 4 has to be read. In order to avoid a recursion, it is recommended to stop reading section 4 *now*.

5 Outlook

During analysis, worst case situations were assumed. In the light of research literature, applying the general and optimal forward error correcting system RSEC as it was done in this thesis becomes a bad choice. For example, Tornado codes prove to be an interesting alternative to RSEC because of their speed [10]. Tornado codes share many properties with RSEC, such as recovering devices. But Tornado codes can only reconstruct the set if $n + \varepsilon$ devices are gathered. Byers et al. [10] assessed during 10000 trials that the inefficiency of Tornado codes was 1.0536 on average, with a maximum of 1.1 and a standard deviation of 0.0073 [10]. Including Tornado codes would affect z and, subsequently, $p(h)$.

RSEC has to contact all checksum devices since every checksum device is dependent on all data devices. Shahabinejad et al. [40] and Sathiamoorthy et al. [38] reduce the high repair cost. The effects of an inclusion of this approach would be manifold and would affect both r and the required share of updates in h .

Improving the adaption of r would be an interesting challenge, too. The optimal definition is recursive and needs to include more information in h . The communication across moving windows motivates the need to introduce JUMP. r is affecting and affected by the statistics of neighbouring moving windows.

The piggybacking of erasure coding with its locality-reducing properties on underlying methods is an interesting task. Since there are many settings in a distributed spatial database where commonly accessed elements occur, and since scope expansion can be performed, the methodology presented in this thesis can be used to reduce the dependence on locality for more than the two proposed modifications.

6 Summary

GIS-2 systems have requirements that are defined by the concerns of the public good [45]. The requirements motivate the need for a self-organizing, decentralized spatial database. Even the optimization of the most basic queries such as range queries is difficult since the optimal allocation of fragments of the spatial database is a difficult problem.

The research has focussed on striving for more locality properties or on caching. The effects of introducing redundancy on locality, however, are little-known. This thesis introduces an overlay enriched with redundancy. The application of such an overlay results in less dependence on locality. It is shown that such an overlay results in a smaller number of contacted nodes per query if only read requests on objects are made. There is a maximal ratio of updates that may be tolerated in the history of queries if the overlay should stay profitable in terms of number of contacted nodes, especially if the overlay has to alter underlying structures in order to be able to apply redundancy. This thesis derives how this ratio of updates is determined exactly.

The results of the application of redundancy are less contacted nodes during a Chord node lookup. Additionally, I provide the possibility of fast parallel object acquisition in the context of the method of Tanin, Harwood, Samet et al. [21, 49, 48, 47], because every information piece is new knowledge. With my method, it is not required to perform a complete download at a single node anymore.

Furthermore, the effect of redundancy as applied by the overlay method of this thesis that employs RSEC is analyzed thoroughly.

The resulting system supports basic queries of a GIS-2 more efficiently. The automatic organization of spatial data is an automatic process since data is sorted according to a tree. Furthermore, the Connectivity layer uses Chord, a decentralized lookup protocol. The Core services layer enables resource discovery and data insertion through a decentral Quadtree or Octree.

Such a system is beneficial for a public that wishes to participate in a global spatial database. It becomes possible to get an impression of what is going on, and it becomes possible to participate by contributing to the spatial database [45]. As a matter of fact, the necessary feedback loop is closed [25].

The advantages stemming from the self-organization of Chord [44] make the operation of the spatial database a simple matter, and thereby ease the entry. Additionally, the system is resilient and supports network partitioning, an important advantage in areas with low access speeds to the global internet [16, 27, 46]. My performance improvements resulting from more locality of the system further lower the barrier to adoption. Building a system that promotes participation and informed citizens and which is usable for the general public nourishes and promotes the possibility for a sustainable, functional democracy [45].

The goals of a GIS-2 such as democratic societies or community-defined goals have become more possible by the contribution of this thesis [45].

7 List of figures

1	Redundancy applied to files	4
2	Redundancy applied to data structures	5
3	Context of this thesis	13
4	Context of this master thesis	16
5	Problem the author is trying to solve	17
6	The layers of the OPeN architecture	18
7	Skip graph and Skip list	21
8	Records and indexes	24
9	Applying redundancy to records	24
10	How updates are handled by the original method and my redundancy method	25
11	Advantages and disadvantages of an extension of a dataset	26
12	History h through the time	27
13	The share of updates	28
14	A MX-CIF Quadtree	30
15	Some example objects in the Quadtree	31
16	Chord: DHT	32
17	Chord: Example	32
18	Chord: Neighbours	33
19	A range query: All spatial objects intersecting the query rectangle are returned	34
20	Query rectangle and its objects	35
21	Chord lookup without optimization	36
22	MX-CIF Quadtree: f_{min} and f_{max}	37
23	An example of a divided spatial object in the Quadtree	38
24	Chord: Finger table explained	38
25	Search using finger tables in Chord	39
26	Enriching a spatial object in the Quadtree	41
27	Chord: Comparison between a search and a finger table	42
28	Chord: Introducing a moving window	42
29	Padding in order to align with block borders	44
30	Determining the intersection point $(7, 4)$	45
31	The different approach of this thesis	53
32	Caching does not support random accesses	54
33	A moving window	56
34	The devices of a moving window	58
35	The checksum devices of a node	58
36	Affected moving windows after a node departure	60
37	Why the last node also stores checksum devices	60
38	Effect of grouping	63
39	A redundant polygon	63
40	Example: Application Layer	64
41	Example: determining G_{min} and G_{eff}	65
42	Example: MBR pruning	69
43	Buffering spatial objects	70
44	Cost model: introduction	72
45	Not applying the method	76
46	Effects of increasing the redundancy	77

47	Analyzing the required redundancy	80
48	No scope expansion for the Chord lookup modification	82
49	Effects of scope expansion	83

8 List of tables

1	A telephone book with an appendix of checksums.	5
2	A telephone book with directly appended checksums.	5
3	A telephone book with telephone numbers with directly appended checksums and an appendix of checksums.	6
4	Enumeration of the elements of $GF(2^4)$	46
5	Logarithm table for $GF(2^4)$	47
6	Recovery cases	51
7	Effects of changes in the input devices	52
8	The serialized spatial object S and the resulting data devices . .	65
9	The resulting devices and assignment to the nodes.	66
10	A subset of the distributed hash table	67
11	The devices of the moving window rooted at node AA	67

9 List of terms

- $I\alpha$: Research concentrated around the idea of making use of hierarchies.
- $I\beta$: Research concentrated around the idea of organizing a set of peers along a distributed hash table.
- II : Research concentrated around the idea of redundancy.
- A : Vandermonde matrix with elements $a_{i,j}$
- A : The names of objects from figures are in typewriter mode.
- a : The number of nodes that have to be contacted for a read operation in the underlying method
- a' : The number of nodes that have to be contacted for a read operation in the piggybacking method
- α^i : Element i of a Galois field
- α_{Chord} : From Chord [44], altering the path length by $\alpha_{Chord} = 0.5$
- B : Information dispersal matrix with elements $b_{i,j}$
- B' : Information dispersal matrix with n selected rows
- b : The number of nodes that have to be contacted for an update operation in the underlying method
- b' : The number of nodes that have to be contacted for an update operation in the piggybacking method
- β : The number of nodes that do not have to be contacted during a Chord lookup with the piggybacking method. Without the piggybacking method, $\beta=0$
- C : Vector of checksum words with elements c_i . c'_i is a copy of c_i . c''_i is an updated element of vector C .
- D : Vector of data words with elements d_i . d'_i is a copy of d_i . d''_i is an updated element of vector D .
- D^3 : The D^3 structure of Sioutas et al. [42] aims for locality properties
- $d(t_C)$: Average number of hops required to reach target node t_C
- δ : Helper variable used in figure 47
- ε : A tiny number larger than 0
- F : The inverse information dispersal matrix with elements $f_{i,j}$
- f_{min} : Fundamental minimum level of the tree of Tanin, Harwood, Samet et al. [21, 49, 48, 47]
- f_{max} : Maximal level of the tree of Tanin, Harwood, Samet et al. [21, 49, 48, 47]

- G_{eff} : List of control points for a spatial object that are effectively responsible for storing its content
- G_{min} : List of control points residing at level $l = f_{min}$ for a spatial object S or a search rectangle
- $GF(2^w)$: Galois field over 2^w elements
- H : Variable used in section 3.3.2
- h : History of size $|h|$
- I : Number of input/data bytes
- i : Used to denote the position of an element in a list or step in an algorithm
- J : Number of checksum bytes
- j : Used to denote a position, for example for an element in a matrix
- K : A key in the identifier space of Chord
- k : Blocksize of a device in bytes
- κ : A node in the DHT
- L : Number of elements left to the node defining a moving window.
- l : A level of the Quadtree or Octree, 0=root of the tree
- λ : A node in the DHT
- MBR : Minimum bounding rectangle
- m : number of checksum devices
- μ : A node in the DHT
- n : number of data devices
- N : Total number of participants in the Chord DHT
- $O(\log N)$: Chord lookup contacts $\log N$ nodes on average
- o : Doubly linked lists reside at level o of the Skip graph of Goodrich et al. [17]
- P : List of p nodes participating in a set encoded with erasure coding with nodes labeled p_0, p_1 , et cetera.
- p : Number of participants in an RSEC-encoded set, $p = |P|$. For the modification of section 3.4.2, $p = |P| = s$. For the modification of section 3.4.3, $p = |P|$. For the modification of section 3.5.2, $p = |P| = N$. In figure 32, $p = 4$.
- $p(h)$: The profit according to the cost model
- Q : Power of an element in a Galois field

- q_d : Number of missing data devices in the context of section 3.3.4
- q_c : Number of missing checksum devices in the context of section 3.3.4
- R : Read operation affecting a set of devices, possible element of h . $|R|$ is the number of read operations listed in h .
- R_{win} : Number of elements right of node defining a moving window.
- RSEC : Reed-Solomon erasure coding [35]; the RAID-like algorithm calculating checksum devices and restoring data devices [31, 32]
- r : Redundancy. $r = 1$ is equivalent to full replication over all p participants
- r in percent : 100% needs twice as much storage than the set of data devices needs
- r_{min} : Each participant holds one data device and one checksum device
- S : Spatial object in the context of the method of Tanin, Harwood, Samet et al. [21, 49, 48, 47]
- s : Size of the moving window
- t : Parameter defining the size of the Chord identifier space
- t_C : Target node of a Chord node lookup
- t_G : Node on which a spatial object or a part thereof is stored on.
- t_M : The first contacted node in a moving window.
- T : Maximal number of objects in a bucket (from [37])
- U : Update operation affecting a set of devices, possible element of h . $|U|$ is the number of update operations listed in h .
- u_j : Element j of matrix A .
- V : A variable in a system of equations (actually a data word)
- v : v -way redundancy is $v \cdot 100\%$ redundancy
- W : A variable in a system of equations (actually a data word)
- w : Word size of a word in bits
- X : Coordinate on first dimension
- X^i : Power representation of an element of a Galois field
- Y : Coordinate on second dimension
- Z : Coordinate on third dimension
- z : Number of nodes that have to be contacted for the successful reconstruction of the data devices (equation 7 on page 57)

10 Bibliography

References

- [1] Abu-Libdeh, H., Princehouse, L., and Weatherspoon, H. (2010). RACS: a case for cloud storage diversity. In *Proceedings of the 1st ACM Symposium on Cloud Computing, SoCC 2010, Indianapolis, Indiana, USA, June 10-11, 2010*, pages 229–240.
- [2] Alaei, S., Ghodsi, M., and Toossi, M. (2010). Skiptree: A new scalable distributed data structure on multidimensional data supporting range-queries. *Computer Communications*, 33(1):73–82.
- [3] Arup Foresight (2018). Drivers of change — Arup Foresight. <http://www.driversofchange.com/>. [Online; accessed 9-May-2018].
- [4] Aspnes, J., Kirsch, J., and Krishnamurthy, A. (2004). Load balancing and locality in range-queriable data structures. In *Proceedings of the Twenty-Third Annual ACM Symposium on Principles of Distributed Computing, PODC 2004, St. John's, Newfoundland, Canada, July 25-28, 2004*, pages 115–124.
- [5] Barabas, C., Narula, N., and Zuckerman, E. (2017). Defending internet freedom through decentralization: back to the future? *The center for civic media & the digital currency initiative*.
- [6] Benouaret, K., Valliyur-Ramalingam, R., and Charoy, F. (2013). Crowdsc: Building smart cities with large-scale citizen participation. *IEEE Internet Computing*, 17(6):57–63.
- [7] Bertino, E., Thuraisingham, B. M., Gertz, M., and Damiani, M. L. (2008). Security and privacy for geospatial data: concepts and research directions. In *Proceedings of the SIGSPATIAL ACM GIS 2008 International Workshop on Security and Privacy in GIS and LBS, SPRINGL 2008, November 4, 2008, Irvine, California, USA, Proceedings*, pages 6–19.
- [8] Bisadi, P., Mirikharaji, Z., and Nickerson, B. G. (2017). A fault tolerant peer-to-peer spatial data structure. *Peer-to-Peer Networking and Applications*, 10(4):874–886.
- [9] Blanas, S. and Samoladas, V. (2009). Contention-based performance evaluation of multidimensional range search in peer-to-peer networks. *Future Generation Comp. Syst.*, 25(1):100–108.
- [10] Byers, J. W., Luby, M., and Mitzenmacher, M. (1999). Accessing multiple mirror sites in parallel: Using tornado codes to speed up downloads. In *Proceedings IEEE INFOCOM '99, The Conference on Computer Communications, Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies, The Future Is Now, New York, NY, USA, March 21-25, 1999*, pages 275–283.
- [11] Chen, H. C., Hu, Y., Lee, P. P., and Tang, Y. (2014). NCCloud: A network-coding-based storage system in a cloud-of-clouds. *IEEE Transactions on computers*, 63(1):31–44.

- [12] Faculty assembly (2010). UZH — Faculty of Science — Mission Statement. <https://www.mnf.uzh.ch/en/fakultaet/leitbild.html>. [Online; accessed 9-May-2018].
- [13] Ferrucci, L., Ricci, L., Albano, M., Baraglia, R., and Mordacchini, M. (2016). Multidimensional range queries on hierarchical Voronoi overlays. *J. Comput. Syst. Sci.*, 82(7):1161–1179.
- [14] Ganesan, P., Yang, B., and Garcia-Molina, H. (2004). One torus to rule them all: Multidimensional queries in P2P systems. In *Proceedings of the Seventh International Workshop on the Web and Databases, WebDB 2004, June 17-18, 2004, Maison de la Chimie, Paris, France, Colocated with ACM SIGMOD/PODS 2004*, pages 19–24.
- [15] Geisel, W. A. (1990). Tutorial on reed-solomon error correction coding. *NASA Lyndon B. Johnson Space Center, Houston, TX, USA*.
- [16] Goddard, S., Deogun, J. S., Harms, S. K., Hayes, M. J., Hubbard, K. G., Reichenbach, S. E., Revesz, P. Z., Waltman, W. J., and Wilhite, D. A. (2004). A geospatial decision support system for drought risk management. In *Proceedings of the 2004 Annual National Conference on Digital Government Research, DG.O 2004, 2004*.
- [17] Goodrich, M. T., Nelson, M. J., and Sun, J. Z. (2009). The rainbow skip graph: A fault-tolerant constant-degree P2P relay structure. *CoRR*, abs/0905.2214.
- [18] Gu, Y., Boukerche, A., and De Grande, R. E. (2016). Supporting multi-dimensional range queries in hierarchically distributed tree. *Concurrency and Computation: Practice and Experience*, 28(6):1848–1869.
- [19] Gupta, A., Agrawal, D., and El Abbadi, A. (2003). Approximate range selection queries in peer-to-peer systems. In *CIDR 2003, First Biennial Conference on Innovative Data Systems Research, Asilomar, CA, USA, January 5-8, 2003, Online Proceedings*.
- [20] Halbawi, W., Azizan-Ruhi, N., Salehi, F., and Hassibi, B. (2017). Improving Distributed Gradient Descent Using Reed-Solomon Codes. *CoRR*, abs/1706.05436v1.
- [21] Harwood, A. and Tanin, E. (2003). Hashing spatial content over Peer-to-Peer networks.
- [22] Hu, L., Ku, W., Bakiras, S., and Shahabi, C. (2010). Verifying spatial queries using voronoi neighbors. In *18th ACM SIGSPATIAL International Symposium on Advances in Geographic Information Systems, ACM-GIS 2010, November 3-5, 2010, San Jose, CA, USA, Proceedings*, pages 350–359.
- [23] Hu, L., Ku, W.-S., Bakiras, S., and Shahabi, C. (2013). Spatial Query Integrity with Voronoi Neighbors. *IEEE Transactions on Knowledge and Data Engineering*, 25(4):863–876.

- [24] Huq, S. and Ghosh, S. (2017). Locally self-adjusting skip graphs. *CoRR*, abs/1704.00830.
- [25] Janssen, M., Charalabidis, Y., and Zuiderwijk, A. (2012). Benefits, adoption barriers and myths of open data and open government. *IS Management*, 29(4):258–268.
- [26] Karapiperis, D., Gkoulalas-Divanis, A., and Verykios, V. S. (2016). LSHDB: a parallel and distributed engine for record linkage and similarity search. In *IEEE International Conference on Data Mining Workshops, ICDM Workshops 2016, December 12-15, 2016, Barcelona, Spain*.
- [27] Kodrich, K. and Laituri, M. (2008). On line disaster response community & People as sensors of high magnitude disasters using internet GIS. *Sensors*, 8(5):3037–3055.
- [28] Litwin, W. and Schwarz, T. J. E. (2000). Lh*_{rs}: A high-availability scalable distributed data structure using reed solomon codes. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, May 16-18, 2000, Dallas, Texas, USA.*, pages 237–248.
- [29] Özsu, M. T. and Valduriez, P. (2011). *Principles of Distributed Database Systems, Third Edition*. Springer Science & Business Media, New York, Dordrecht, Heidelberg, London.
- [30] Pickles, J. (1999). Arguments, debates, and dialogues: the GIS-social theory debate and the concern for alternatives. *Geographical Information Systems: Principles and Technical Issues*, 1:49–60.
- [31] Plank, J. S. (1997). A tutorial on Reed-Solomon coding for fault-tolerance in RAID-like systems. *Software – Practice & Experience*, 27(9):995–1012.
- [32] Plank, J. S. and Ding, Y. (2005). Note: Correction to the 1997 tutorial on Reed-Solomon coding. *Software – Practice & Experience*, 35(2):189–194.
- [33] Plaxton, C. G., Rajaraman, R., and Richa, A. W. (1999). Accessing nearby copies of replicated objects in a distributed environment. *Theory of computing systems*, 32(3):241–280.
- [34] Ratnasamy, S., Francis, P., Handley, M., Karp, R. M., and Shenker, S. (2001). A scalable content-addressable network. In *SIGCOMM*, pages 161–172.
- [35] Reed, I. S. and Solomon, G. (1960). Polynomial codes over certain finite fields. *Journal of the society for industrial and applied mathematics*, 8(2):300–304.
- [36] Resolution (2015). General assembly resolution RES/70/1. Transforming our world: the 2030 agenda for sustainable development. *Seventieth United Nations General Assembly, New York*, 25.
- [37] Samet, H. (2006). *Foundations of multidimensional and metric data structures*. Morgan Kaufmann, San Francisco, CA, USA.

- [38] Sathiamoorthy, M., Asteris, M., Papailiopoulos, D., Dimakis, A. G., Vadali, R., Chen, S., and Borthakur, D. (2013). Xoring elephants: Novel erasure codes for big data. *CoRR*, abs/1301.3791.
- [39] Schmaltz, C. and Aziz, I. (2012). The connected company—bridging data silos. *it-Information Technology Methoden und innovative Anwendungen der Informatik und Informationstechnik*, 54(5):235–242.
- [40] Shahabinejad, M., Ardakani, M., and Khabbazi, M. (2017). An erasure code with reduced average locality for distributed storage systems. In *2017 International Conference on Computing, Networking and Communications, ICNC 2017, Silicon Valley, CA, USA, January 26-29, 2017*, pages 427–431.
- [41] Shamir, A. (1979). How to share a secret. *Communications of the ACM*, 22(11):612–613.
- [42] Sioutas, S., Sourla, E., Tsihlias, K., and Zaroliagis, C. D. (2015). ART⁺: A fault-tolerant decentralized tree structure with ultimate sub-logarithmic efficiency. In *Algorithmic Aspects of Cloud Computing - First International Workshop, ALGO CLOUD 2015, Patras, Greece, September 14-15, 2015. Revised Selected Papers*, pages 126–137.
- [43] Speiser, S. and Harth, A. (2010). Taking the LIDS off data silos. In *Proceedings the 6th International Conference on Semantic Systems, ISEMANTICS 2010, Graz, Austria, September 1-3, 2010*.
- [44] Stoica, I., Morris, R. T., Liben-Nowell, D., Karger, D. R., Kaashoek, M. F., Dabek, F., and Balakrishnan, H. (2003). Chord: a scalable peer-to-peer lookup protocol for internet applications. *IEEE/ACM Trans. Netw.*, 11(1):17–32.
- [45] Sui, D. and Goodchild, M. (2011). The convergence of GIS and social media: challenges for GIScience. *International Journal of Geographical Information Science*, 25(11):1737–1748.
- [46] Tanin, E., Brabec, F., and Samet, H. (2002). Remote access to large spatial databases. In *ACM-GIS 2002, Proceedings of the Tenth ACM International Symposium on Advances in Geographic Information Systems, McLean, VA (near Washington, DC), USA, USA, November 8-9, 2002*, pages 5–10.
- [47] Tanin, E., Harwood, A., and Samet, H. (2007). Using a distributed quadtree index in peer-to-peer networks. *The VLDB Journal—The International Journal on Very Large Data Bases*, 16(2):165–178.
- [48] Tanin, E., Harwood, A., Samet, H., Nayar, D., and Nutanong, S. (2006). Building and querying a p2p virtual world. *Geoinformatica*, 10(1):91–116.
- [49] Tanin, E., Harwood, A., Samet, H., Nutanong, S., and Truong, M. T. (2004). A serverless 3d world. In *12th ACM International Workshop on Geographic Information Systems, ACM-GIS 2004, November 12-13, 2004, Washington, DC, USA, Proceedings*, pages 157–165.

- [50] Toda, T., Tanigawa, Y., and Tode, H. (2017). Autonomous and distributed construction of locality aware skip graph. In *14th IEEE Annual Consumer Communications & Networking Conference, CCNC 2017, Las Vegas, NV, USA, January 8-11, 2017*, pages 33–36.
- [51] Van Steen, M., Hauck, F., Ballintijn, G., and Tanenbaum, A. (1998). Algorithmic design of the Globe wide-area location service. *The Computer journal*, 41(5):297–310.
- [52] Xu, H. and Bhalerao, D. (2015). Reliable and Secure Distributed Cloud Data Storage Using Reed-Solomon Codes. *International Journal of Software Engineering and Knowledge Engineering*, 25(9-10):1611–1632.
- [53] Zave, P. (2016). Reasoning about identifier spaces: How to make Chord correct. *CoRR*, abs/1610.01140.
- [54] Zhang, Y., Katz, J., and Papamanthou, C. (2015). Integridb: Verifiable SQL for outsourced databases. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12-16, 2015*, pages 1480–1491.

11 Personal declaration

I hereby declare that the submitted thesis is the result of my own, independent work. All external sources are explicitly acknowledged in the thesis.

Zürich, 20.05.2018

.....

Benedikt Steger