VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

DISTRIBUTED P2P DATA BACKUP SYSTEM

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE Bc. ISTVÁN MÉSZÁROS
AUTHOR

BRNO 2013

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

DISTRIBUOVANÝ P2P SYSTÉM PRO ZÁLOHOVÁNÍ

DISTRIBUTED P2P DATA BACKUP SYSTEM

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE Bc. ISTVÁN MÉSZÁROS
AUTHOR

VEDOUCÍ PRÁCE Ing. IGOR SZŐKE, Ph.D.
SUPERVISOR

BRNO 2013

Abstrakt

Tato diplomová práce představuje model a prototyp kooperativního distributivního systému zálohování dat založeném na P2P komunikační síti. Návrh systému umožňuje uživatelům přispět svým lokálním volným místem na disku do systému výměnou za spolehlivé úložiště jejich dat u jiných uživatelů. Představené řešení se snaží splnit požadavky uživatelů na ukládání dat, zároveň však také řeší, jak se vypořádat s mírou nepředvídatelnosti uživatelů ohledně poskytování volného místa. To je prováděno dvěma způsoby - využitím Reed - Solomon kódů a zároveň také tím, že poskytuje možnost nastavení parametrů dostupnosti. Jedním z těchto parametrů je časový rozvrh, který značí, kdy uživatel může nabídnout předvídatelný přínos do systému. Druhý parametr se týká spolehlivosti konkrétního uživatele v rámci jeho slíbeného časového úseku. Systém je schopen najít synchronizaci ukládaných dat na základě těchto parametrů. Práce se zaměřuje rovněž na řešení zabezpečení systému proti širšímu spektru možných útoků. Hlavním cílem je publikovat koncept a prototyp. Jelikož se jedná o relativně nové řešení, je důležitá také zpětná vazba od široké veřejnosti, která může produkt používat. Právě jejich komentáře a připomínky jsou podnětem pro další vývoj systému.

Abstract

This master's thesis briefly presents a model and a prototype of a cooperative distributed data storage system based on P2P network communication. The system design lets the users to contribute their local free space in exchange of reliable remote storage, which is a virtually co-allocated space on other users' devices. The introduced solution tries to meet the users' requirements for data storing, meanwhile work around the issue of their unpredictability when it comes to contribution of free space. This is mainly done by harnessing the capabilities of Reed-Solomon codes, but by providing adjustable parameters for the contributions as well. One of these parameters is the time frame, which describes when can a user offer contribution to the system. The second parameter refers to the reliability of the user inside his promised time frame. The system is responsible for finding partner contributions for data storage based on these parameters. This thesis also focuses on solving the wider spectrum security issues of the system. The main goal of this work was to publish the prototype and the concept itself. As this is a relatively new solution and design, feedback is required from the public, which is also the main source for further designing and developing the system.

Klíčová slova

Online Zálohování, Kooperatíný Zálohovaní, Distribuovaný, Peer-to-peer, Ukládání dat, Spolehlivý, Adaptívný, Flexibilný, Java, AES, Reed-Solomon kódy, Systém založený na časových intervalech, MD5 hash funkce, Erasure kódy

Keywords

Online Backup, Cooperative Backup, Distributed, Peer-to-peer, Data storage, Secure, Reliable, Adaptive, Flexible, Java, AES, Reed-Solomon Codes, Time frame based system, MD5 hash function, Erasure codes

Citace

István Mészáros: Distributed P2P Data Backup System, diplomová práce, Brno, FIT VUT v Brně, 2013

Distributed P2P Data Backup System

Prohlášení

Prohlašuji, že jsem tento diplomový projekt vypracoval samostatně pod vedením Ing. Igor Szőke Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal

. . . .
István Mészáros
May 22, 2013

Poděkování

Chtěl bych poděkovat mému vedoucímu Ing. Igor Szőke, Ph.D. za jeho připomínky a cenné rady, poskytnutou pomoc a trpělivost.

Contents

Chapter 1

Introduction

Distributed peer-to-peer storage systems are relatively new solutions for remote data storing. They promise a cheap, secure and reliable data backup service. However there are only a few real working realizations, from which only one is commercially accessible by public users.

Implementing a system, based on distributing the client's data between contributors faces significantly more technical issues than implementing classic client-server architecture storage systems. The design has to deal with unreliability of the contributors and a different spectrum of possible attacks.

In this thesis a possible solution and a prototype of a distributed peer-to-peer data backup solution is presented. The main goal of this research is to uncover the advantages and disadvantages of the discussed systems. This is done by publishing the functional prototype and the concept for the wide public, mainly in order to receive feedback. The solution presented in this thesis is different from the other available distributed solutions (or which were available in the past). The main difference is that a wide range of possibilities are given to the users to indirectly select their partners for data storing. They are allowed to choose the parameters, which describe the time interval of their data availability and the reliability percentage. These options make the system more flexible and give ability to reach more contributors from the public than the currently available solutions.

This thesis is divided into six chapters. In chapter 2 the available regular and cooperative data backup solutions are presented and compared. The basic issues about security, reliability and communication of the distributed systems are discussed in chapter 3. In chapter 4 an unconventional model of distributed peer-to-peer data storage system is presented. Besides that the necessary agreements between the clients and contributors, fair pricing model and basic operations such as data storing and retrieving are described as well. In chapter 5 the architecture of this irregular system is presented. Among the covered topics are the client and server side applications, communication protocols between them, the database architecture and the used technologies and frameworks. In chapter 6 the real time test results, objective measurements, subjective feedback from the test users are presented. In the last chapter conclusion is discussed based on the users' feedback and objective measurements. The possible improvements to the design of this system is described as well.

The main motivation for this study was to find an alternative solution for inefficient client-server online storage architecture, which often deals with uplink bandwidth problems, vulnerability against cyber attacks and limited small storage space. The idea was represented in Starcube startup accelerator in Brno (Spring 2013).

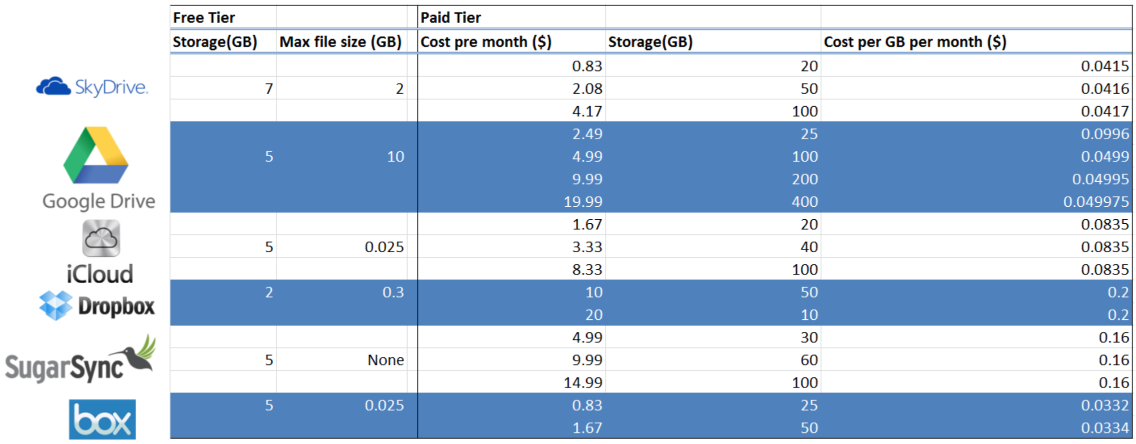
Chapter 2

Online Data Storage Systems

In this chapter the motivation for creating the time frame based cooperative data storage system is presented. The basic concepts of online backups and general backup techniques are introduced as well.

2.1 Client-Server Architecture Based Data Storage Systems

There are several online backup solutions worldwide, which are well known for the public. These solutions are mainly data center based, which means the providing company offers a portion of his storage complex to other companies, organizations, or individuals in exchange for money. These solutions are comfortable for the users, due to simple and user-friendly client applications and the always available servers. Most of these cloud based storage systems follow the Freemium business model, which means they offer also free but limited service for everybody, hoping that some of the users will pay for the extended service. The table below shows some of the well known online backup services and their prices.

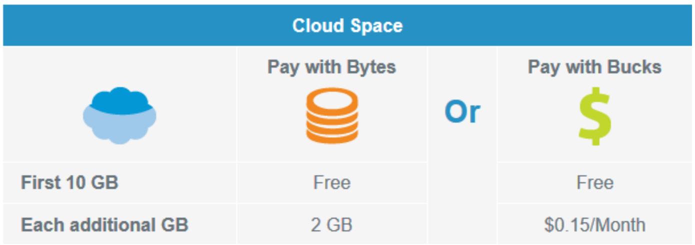


	Free Tier		Paid Tier		
	Storage(GB)	Max file size (GB)	Cost pre month ($)	Storage(GB)	Cost per GB per month ($)
SkyDrive	7	2	0.83	20	0.0415
			2.08	50	0.0416
			4.17	100	0.0417
Google Drive	5	10	2.49	25	0.0996
			4.99	100	0.0499
			9.99	200	0.04995
			19.99	400	0.049975
iCloud	5	0.025	1.67	20	0.0835
			3.33	40	0.0835
			8.33	100	0.0835
Dropbox	2	0.3	10	50	0.2
			20	10	0.2
SugarSync	5	None	4.99	30	0.16
			9.99	60	0.16
			14.99	100	0.16
box	5	0.025	0.83	25	0.0332
			1.67	50	0.0334

Table 2.1: Comparison of online backup service[1]

[1] http://lifehacker.com/5905702/dropbox-google-drive-skydrive-and-others-pricing-per-gb-and-more-compared- sin-convenient-charts

As table 2.1 shows each of these back up services offer limited free storage place for users (Freemium business model). All of them are based on centralized storage system. On the other hand if these services are compared with a solution from the company Symform, which is a cooperative data backup solution a significant difference can be found. The solution from the company Symform offers a unlimited reliable storage place.



Cloud Space			
	Pay with Bytes	Or	Pay with Bucks $
First 10 GB	Free		Free
Each additional GB	2 GB		$0.15/Month

Table 2.2: Symform free backup service[2]

The unlimited backup storage is achieved by making the users able to contribute their free unused space from their local computers to the system in rate of 1 for 2 GB. This means if they provide 2 GB free space from their local computers (or NAS, server) they get 1 GB reliable online storage space (which is actually space on the other contributors' devices). This is the basic idea behind a cooperative data storage systems.

2.2 Cooperative Data Storage Systems

There are only a few commercial and non-commercial distributed backup solutions currently available. The only available commercial solution is provided by the already mentioned company Symform (recently some startup companies are trying with similar ideas, like Spacemonkey [3]). In the past Wuala storage trading was well known also. This system was shut down in 2010. The reason Wuala shutdown its service, was due to the complexity of the maintenance of their completely distributed system (without tracker server), as well the significant drop in hard drive prices lead to adopt only data center based solution[8].

Symform's model is very simple. Each of their users can contribute free space from their immovable nonstop running device(NAS, server, etc.) to the network, in exchange of reliable online backup storage. This means that users are able to pay for online backup storage by giving up part of their own local hard drive. This has only one disadvantage, the users who contribute have to own a server, which has to be always available and be reachable from the internet. This is not an issue for companies, but for individuals it can be limiting. In chapter 4 a possibility is discussed, how to avoid the obligation of having a server running for contribution in a cooperative data backup solution.

There can be 2 basic types of distributed data storage system designs:

- **Hybrid or Server based** - this category of storage systems is based on a control server (or servers), which tracks the users, but the data is still distributed over the network via peer-to-peer protocols. BitTorrent's new backup service, which is coming

[2] http://www.symform.com/our-solutions/pricing/

[3] http://www.kickstarter.com/projects/clintgc/space-monkey-taking-the-cloud-out-of-the-datacente

out later this year[4] can be mentioned as an example.

- **Completely distributed** - in this category the tracking is not dependent on a server, instead it uses a second layer of routing at the application level at each peer, which finds the appropriate partners for the distribution in the peer-to-peer network (e.g. Chord is such a system under development and research at MIT,Cambridge)[2]

This thesis focuses on uncovering the opportunities, what a hybrid or server based cooperative data storage system can offer to the users.

2.3 Data Repository Models

Backup in general means to store data at least in 2 different places. In case if one of the storage devices crashes, it can be still recovered from the other. There are several backup techniques, which are widely used around the world. The most common schemes are the following:

- **System Images (Complete backup)** - This is the simplest way to protect a file system against disk failures and file corruption. It is done by copying the entire content of the file system to a backup device. The resulting archive is called a full backup. This has some disadvantages, that it is time consuming and each backed up image of a system requires the same amount of space for backup as the original data (if not compressed, which decreases the performance even more)[1].
- **Incremental** - Faster and smaller backups can be achieved using an incremental backup scheme, which copies only those files that have been created or modified since a previous backup (full or incremental). Incremental schemes reduce the size of backups, since only a small percentage of files change on a given day. A typical incremental scheme performs occasional full backups supplemented by frequent incremental backups[1].
- **Differential** - A differential backup is a type of data backup, that saves only the difference in the data since the last full backup[5].
- **Continuous Data Protection** - The system immediately logs every change on the system. This is generally done by saving byte or block-level differences rather than file-level differences as in incremental back up[17].

[4] http://www.nbcnews.com/technology/gadgetbox/bittorrent-takes-dropbox-sync-sharing-backup-service-1C8123522

[5] http://searchdatabackup.techtarget.com/definition/differential-backup

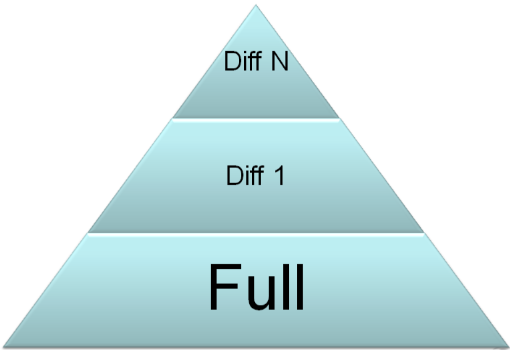


Figure 2.1: Incremental backup method scheme[1].

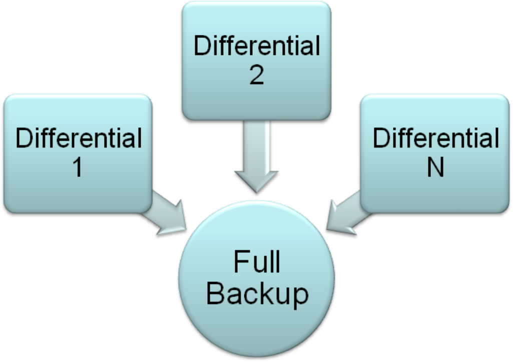


Figure 2.2: Differential backup method scheme[2].

The figures 2.1 and 2.2 show the difference between the incremental and differential backup process. As the scheme shows, the size of the incremental backup grows by each change made since the last (full or incremental) backup, while in the case of differential backup only the difference since the last full backup is stored.

[1] http://wiki.r1soft.com/display/TP/Backup+Method+-+Incremental+Backup

[2] http://wiki.r1soft.com/display/TP/Backup+Method+-+Differential

Chapter 3

Issues and Solutions For The Cooperative Data Backup Solutions

In this chapter the main issues and problems of a distributed peer-to-peer data storage system are discussed.

3.1 Reliability Issues

A key problem of a cooperative data storage system is how to create a reliable service for storing the data with a set of unreliable resources. Even if the contributors have obligation to run a server, they are unreliable with a certain percentage, which is due to device failures, power cuts, possibility to abandon the system, etc. Reliability of the data can be ensured by two basic ways.

3.2 Replication Model

The first way is to replicate the data among multiple devices (contributions). This is efficient if the redundancy (replication ratio or the count of replications) of the data is high, but by increasing the replications the transfer bandwidth is increasing as well. The model of reaching the desired reliability based on replicating the data is the following:

Let's assume the data is distributed between n contributors. Let contributor C_i has the probability $P(C_i, t)$ of being online in time t, where $i \in \{1..n\}$, then the probability of our data being online at time t is[12]:

$$P_{online}(t) = 1 - \prod_{i=0}^{n}(1 - P(C_i, t)) \tag{3.1}$$

Let's assume $P_{desired}$ is the desired reliability for the data, in ideal case it is 1.0, but due the issues already mentioned (un-installation, power cuts, etc.) this is practically unreachable. In all case $P_{desired}$ has to be less than the calculated P_{online}. This can be reached only by asking more and more contributors for the backup.

In conclusion with this method high reliability can be reached, but in practice it is unusable, due to the ineffective bandwidth and storage space usage. This is because the original data has to be transferred and stored within peers n times, so the redundancy rate in percentage is $100n\%$.

A practical example:

Let $P_{desired} = 0.99(99\%)$, and $\forall i \in \{1..n\} : P(C_i, t) = 0.8$ Therefore:

$$P_{online}(t) = 1 - (0.2)^n$$

$$1 - (0.2)^n \geq 0.99$$

$$n \geq 3$$

So n has to be at least 3, to reach the desired reliability. This means each MB of data has to be transferred 3 times.

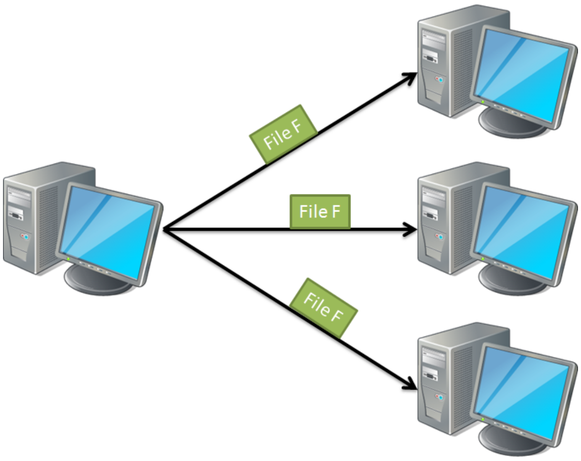


Figure 3.1: Scheme of the replication model with redundancy of 300%.

3.3 Erasure Coding Model

The second way to reach the desired reliability can be done by using erasure codes. These codes are part of the family of FEC (forward error correcting codes), but in comparison with error correction codes they have better parameters with **known** erased data, which can be reflected to the distributed data storage case, when some of the data fragments are missing. The erasure codes can be divided into two main categories:

Optimal erasure codes - with a $(k+m, k)$ block, where k is the size of the original data block and m is the size of the redundancy block. It is possible reconstruct **any** missing m blocks from the $k+m$ blocks. Reed-Solomon codes can be mentioned as part of this category[10].

Near optimal erasure codes - these codes need $(1+\epsilon)k$ to restore k data blocks. Where $\epsilon > 0$, which can be often reduced with the cost of CPU performance and memory. The modern algorithms require relatively small ϵ to reach high reliability, but still these algorithm are used mainly in telecommunication, not in data storing. The latest designed Tornado codes has at $\epsilon = 0.01$ reliability of 99.9% [13]

Using optimal erasure codes the data is not replicated among contributors, but some additional redundant data sequences are created from the original data.

In other words m redundant bytes are added to the original k byte data, so any m bytes of the final $k+m$ encoded data can be missing, and yet the data will be still recoverable. In practice $k+m$ has to be a relatively small number between 1 and 256 (most common), so the data which is longer then k bytes, has to be divide to fragments, where each fragment will be maximally k bytes long. Then to each of this k bytes m redundant bytes are calculated. After that each of the $k+m$ encoded bytes a distributed equally among $k+m$ contributors, so any m of the contributors can be offline, and with the remaining k bytes the original $k+m$ bytes can be reconstructed (containing the original data). Figure 3.2 shows this interleaving technique.

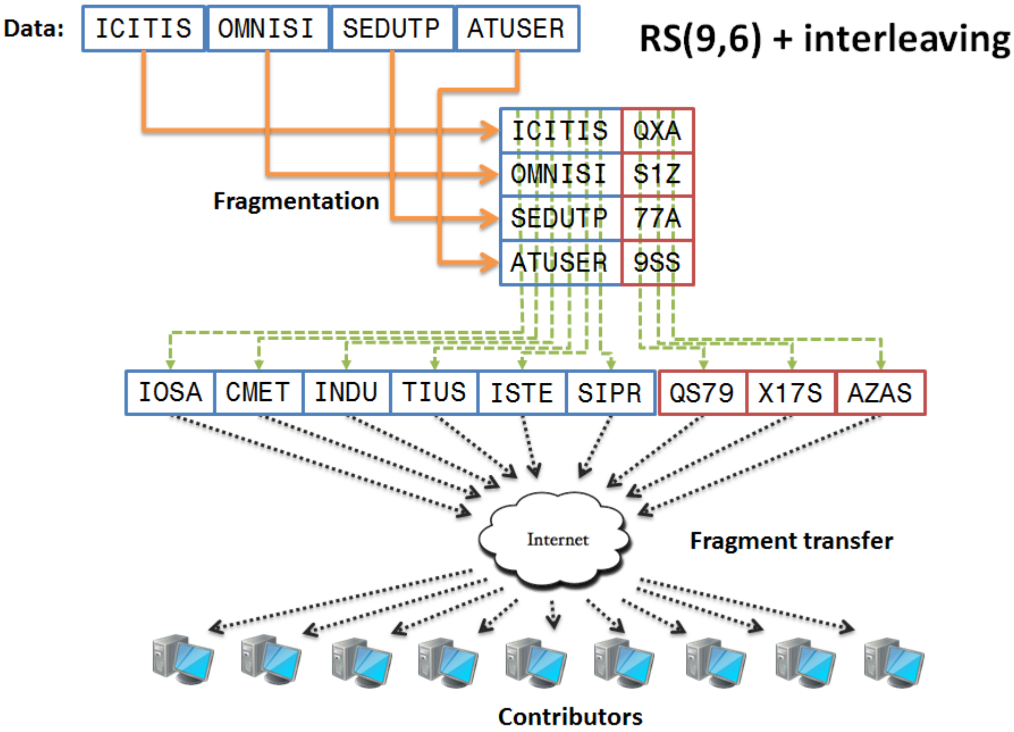


Figure 3.2: Byte interleaving and fragmentation scheme.

This method is much more efficient in usage of bandwidth as well as space. If the contributors availability is known, in order to reach the required reliability the redundancy rate of these methods can be adjusted.

To calculate the probability of the data being available P_{online} is more complex than the situation for replication model. The calculation has to be done with the (Bernoulli) binomial formula [12]:

$$n = k + m$$

$$P_{online}(t) = \sum_{z=k}^{n} \binom{n}{z} P(t)^z (1 - P(t))^{n-z} \tag{3.2}$$

This formula is valid only if every contributor's probability is equal. In reality every contributors probability for being available is different. For this case Poisson binomial distribution can be used. This distribution at its simplest form has high computational complexity, therefore an approximation of this method has to be implemented in practice, this approximation will be discussed in chapter 5.

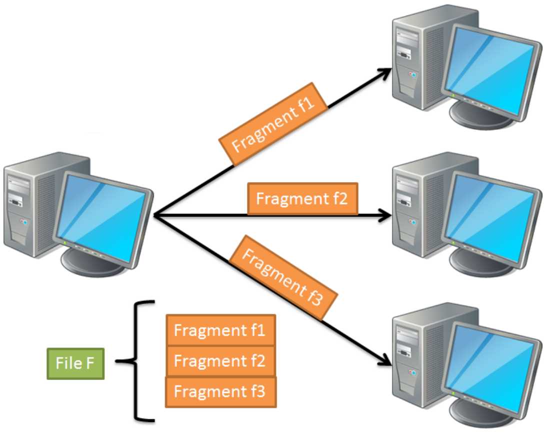


Figure 3.3: Scheme of the erasure code model with redundancy.

3.4 Reed-Solomon Codes

Reed-Solomon codes are widely used in telecommunication, raid systems, QR codes, optical storage devices like CDs DVDs and many other fields to fix errors in data (mainly burst error). The first time this algorithm was published in 1960 by Irving Reed and Gustav Solomon.

The core of the Reed-Solomon codes is a Galois field, which is a finite field generated by a generator polynomial. Galois fields hold only none repeating number sequence, with the length of p^n where p has to be a prime. Most often it is 2, than the $GF(2^n)$ can hold up to numbers with n bits, which makes it practical for usage in software engineering. Most often $n = 8$, because this way the number in the Galois field can be coded into one byte. The complete explanation of the Galois fields and Reed-Solomon coding is out of the scope of this thesis, but the some properties has to be described.

Encoding with Reed-Solomon means to add m parity bytes to the k data bytes, so by dividing the $k + m$ code polynomial (represented by the $k + m$ bytes) with the generator polynomial the reminder is 0. During decoding this reminder is verified; if it is not zero the code contains error.

Decoding is a much more complex multistage process. First the syndromes (coefficients of the reminder after the division with the generator polynomial) have to be found. Than these syndromes are used to determinate, which bytes were corrupted (Chien search). After the bytes were found, the corruption rate has to be determined (Forney algorithm). As the corruption rate is found, these data has to be XOR-ed with the original code sequence, which is containing errors. The Reed-Solomon codes can fix up to $\frac{m}{2}$ unknown errors and m known errors. The decoding algorithm usually needs at least 10 times more resources (computational, memory) as the encoding [14].

Fortunately in the case of known errors, the syndrome searching and corruption rate determining part can be skipped. This will result in a much faster algorithm for known erasure errors, which is the case of the distributed data backup systems. The data from contributors, who are offline is considered as erased code, therefore it has to be calculated.

3.5 Security Issues

Distributed technologies generally deal with more security risks than conventional client-server architecture applications. There are many possibilities how the attackers can abuse or disrupt the system. The most common attacks can be [6]:

Free rider attack – a user wants to back up his data, but he refuses to contribute in order to get free service. A solution for that attack is in a formal contract between the client and the contributor. This contract has to be checked periodically, whether the contributor is keeping his promise. In case the contract is broken, the contributor has to be punished, so the cost of free rider attack is higher than using the service regularly. A contract can be done concerning the ratio of contribution and using the reliable storage. The only problem with this solution is that, it is hard to distinguish free rider attack from a failure contributor device. For this reason "grace" period has to be allowed in the system, which is a time interval when the user is being unpunished for breaking the contract.

Bandwidth attack – since peer-to-peer data transfer is used, the users can easily abuse the bandwidth of their partners by continuously uploading and downloading their data. It is possible to defend against this attack, by limiting the amount of upload and download of the client. Other option is reducing the bandwidth for suspicious clients on the side of the contributor. Also a bandwidth attack can be done by the contributors, by reducing the transfer speed of the files, which are already stored on their devices. Against this attack contract has to be made for minimal upload speed as well. Breaking the contract again should lead to punishment.

Grace period attack – is an attack where a client uses the grace period time to back up his data for free, without contribution. This can be done only for the time while grace period is valid, but by recreating his account the client can reach a free service with a relatively little effort. Possible solution for this attack is prepayment and post-payment. Prepayment requires the client to contribute or pay with money or some service before using the service of others. The best way to do this, is the define the price of the prepayment as high that for the attacker is not benefiting to use this attack. Post-payment is a mechanism of marking the clients (e.g. by IP address), who already attacked the system, and not allowing them to use the service anymore until they pay for their misused grace period. This has a disadvantage that the user can use the grace period once.

Third party data access – as contributor based model is realized over the internet, unauthorized users can read or change the unprotected data of the users. It is possible to prevent this attack by encrypting and interleave the data. Each encrypted fragment has to be sent to a different contributor. This issue is solved if erasure coding model is used, like it was mentioned in the section 3.3. The possible encryption algorithms are described in the section 3.6.

Data corruption attack - since the contributors hold other users' data, they can easily corrupt it (change the data). Even a change of one single bit can lead to a data sequences, what is impossible to decode for the erasure coding algorithms. Against this attack digital signatures can be a possible defense (with at least 2nd pre-image resistant hash functions). This solution will detect the change in data. In case of any change the hash value would be different, indicating the attack. Those contributors who corrupt the data could be punished. The possible hash functions are described in the section 3.8

3.6 Encryption Algorithms

In order to keep the data unreadable by third party users of the system, it has to be encrypted with a symmetric encryption algorithm. The key must be available only for the clients who own the data.

There are two mayor types of symmetric encryption algorithms [5]:

Block encryption algorithm – Uses a block of bits to encrypt data. It is commonly used to encrypt large amount of data.

- **AES** is a 128,192 or 256 bit key globally used and well trusted algorithm.
- **DES** is one of oldest digital encryption algorithms, it has 64 bit block size with 56 bit size key. It is based Feistel's cipher scheme with 16 rounds. Historically it has encrypted the most bytes of all cipher algorithms [9]. DES is considered as none secure algorithm, since it is easy to crack by brute-force cracking algorithm. It's main disadvantage is it's small key size.
- **3DES** is the modern version of the historical DES algorithm, which is considered breakable today. The basic DES algorithm uses 56 bit keys. The newer version 3DES has 168 bit length key. Compared to the AES algorithm 3DES requires more computational resources.
- **BlowFish** is a various 32 up to 448 bit length key symmetric algorithm. It is considered still secure since its birth (1993), but AES has more attention. Blowfish algorithm is based on improved Feistel's cipher scheme.

Stream encryption algorithm – uses two separate streams to encrypt the data. One is the data stream and the second is the key stream, which contains generated pseudo-random values to encrypt data stream. Most often the encryption is done by XORing these two streams.

Block encryption algorithms are generally more secure than stream ciphers, and since the cooperative storage system is not based on stream of data, but static, local and fixed size data, it is better to use block cipher algorithms.

If the reliability model of the cooperative data storage system is based on erasure coding, the distributed data is even more secure. As it was described in the section 3.3, each $k+m$ byte of the erasure code blocks is distributed equally among the $k+m$ contributors with byte interleaving method (each fragment gets exactly 1 byte from each $k+m$ long code sequence), therefore there is no continuous data sequence at any contributor. This makes practically impossible to retrieve information from the data fragments, under condition if none of the contributors has knowledge about the others, who store fragments corresponding to their stored fragment.

3.7 Performance Comparison of Symmetric Encryption Algorithms

In section 3.6, the listed symmetric algorithms are currently considered as secure (except DES), but they have different computational performance. Performance of these algorithms is an important part in the design of a cooperative data backups system. It is expected that the users want to store large amount of data, therefore the usage of computational power for encryption has to be minimized. The table 3.1 shows the performance of the mentioned algorithms:

Algorithm	**Megabytes(2^{20} bytes) Processed**	**Time Taken**	**MB/Sec**
Blowfish	256	3.976	64.386
AES (128-bit key)	256	4.196	61.010
AES (192-bit key)	256	4.817	53.145
AES (256-bit key)	256	5.308	48.229
DES	128	5.998	21.340
(3DES)DES-XEX3	128	6.159	20.783
(3DES)DES-EDE3	64	6.499	9.848

Table 3.1: Comparison of the encryption algorithms.

Table 3.1 contains the speed comparison for some of the most commonly used symmetric cryptographic algorithms: "all were coded in C++, compiled with Microsoft Visual C++ .NET 2003 (whole program optimization, optimize for speed, P4 code generation), and ran on a Pentium 4 2.1 GHz processor under Windows XP SP 1" [16].

In conclusion based on these data it is clear, that Blowfish and AES algorithms has overall better performance compared to 3DES and DES, therefore it is recommended to design the cooperative data storage system using these encryption algorithms.

3.8 Digital Signature and Data Integrity

As mentioned in section 3.5, with the data corruption attack contributors can cause damage to other users of the system. Even the smallest change in the stored fragments at the contributors side could cause the data not being possible to be reconstructed by the erasure coding algorithms. This is the most dangerous attack in the system, because it can be done with minimal effort and can cause loss of the clients data.

In order to prevent the data corruption attacks hash value (checksum) has to be computed for each distributed fragment. The hash value then has to be stored in a database what is reachable by the client. After downloading back the fragments the data integrity must be checked with the hash value from the database. If the checksum of the downloaded file doesn't correspond to the stored one, the fragment has to be dropped. As far as the data transfer runs under reliable TCP protocols, the only reason for the data integrity fail can be the contributor (except man in the middle attacks) itself, therefore he has to be considered as potential attacker.

The hash value is a constant length sequence of bytes, which has the following five main characteristics [9]:

1. Can be computed with low computational complexity.

2. Applicable for various length inputs.
3. **First Pre-image Resistance** - For the known y is computationally "hard" to find any x, where $F(x) = y$, F is the hash function (irreversibility).
4. **Second Pre-image Resistance** - For the known x is computationally "hard" to find any x', where $F(x) = F(x'), x \neq x'$.
5. **Collision Resistance** - For any x is computationally "hard" to find any x', where $F(x) = F(x'), x \neq x'$.

There are many type of implementations of hash functions, currently the two most know algorithms are the following:

MD5 - The MD5 (Message-Digest) Algorithm is a commonly known and popular hash function. It generates 128-bit hash value. Recently some cases of collision resistance attacks have been shown, but this fact doesn't made it deprecated. This discovery made MD5 unusable for SSL certificates and digital signature, which rely on collision resistance, but the pre-image resistance properties are still valid[11][15].

SHA-1 - "Secure Hash Algorithm„ was designed based on MD5 algorithm, it produces 160 bit hash value from the input. It has been standardised by the standards agency NIST (National Institute of Standards and Technology) in 1993. Generally it is considered safer as MD5, because of recent collision resistance failure for MD5 algorithm was proved with SHA-1, even that SHA-1 is very similar to MD5[3]

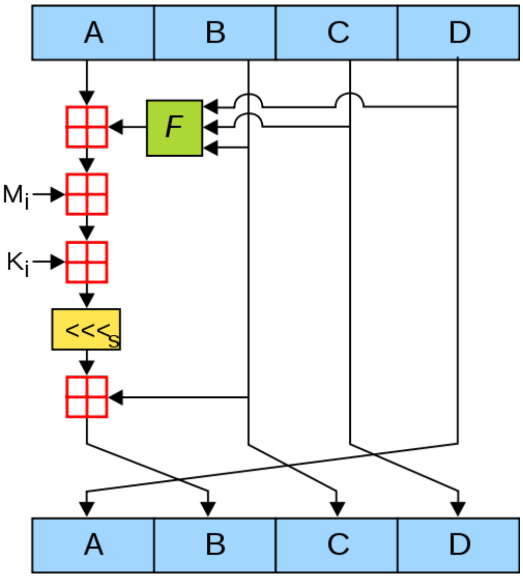


Figure 3.4: MD5 hash function[1].

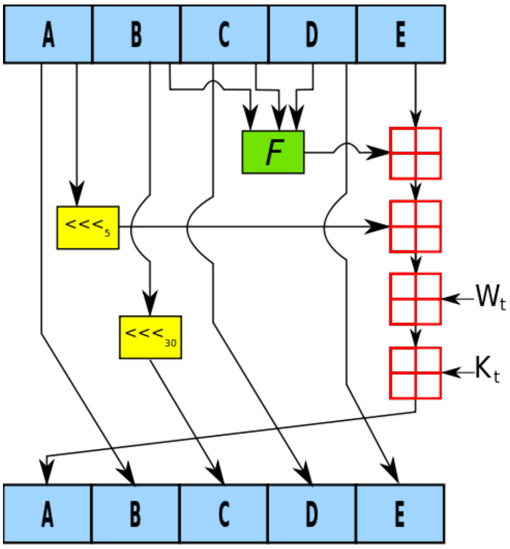


Figure 3.5: SHA-1 hash function[2].

In practice a cooperative data storage system requires a hash function, which meets the first four characteristics (up to 2nd pre-image resistance). Collision resistance doesn't have

[1] http://en.wikipedia.org/wiki/MD5
[2] http://en.wikipedia.org/wiki/SHA-1

to be guaranteed, because in reality the hash value is never send over the network (like in SSL), it has to be stored on the clients side or in a database in a controller server (hash value injection attack can't be performed). Other issue can appear if the data transfer between peers is unsecured. This way it is vulnerable against man in the middle attacks. This attack can be done by corrupting the fragments after leaving the user's but before reaching the contributor's device (or at downloading, reverse process), therefore SSL tunneling is necessary for data transfers between the client and contributor.

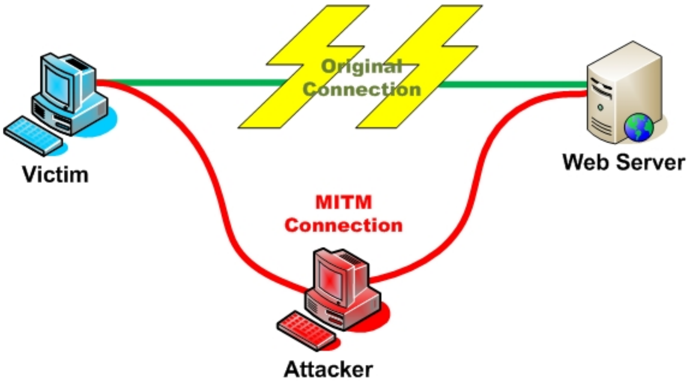


Figure 3.6: Man in the middle attack scheme[3].

3.9 Performance Comparison of Digital Signature Algorithms

If the collision resistance condition for hash functions is ignored both SHA-1 and MD5 are considered secure. The computational performance comparison of these algorithms has to be done, again due to predictable huge amount of data transfer in the system. The following table shows results of a speed test for these algorithms [20]:

Algorithm	CPI	Path Length Instruction	MB/Sec
MD5	0.72	12	197.86
SHA-1	0.52	24	135.30

Table 3.2: Comparison of the MD5 and SHA-1 hash functions.

The table 3.2 shows that MD5 algorithm has better performance over SHA-1, and since it is enough to ensure 2nd pre-image resistance it is the better option.

3.10 Data Transfer Issues and Possible Solutions

The peer-to-peer communication is the biggest issue for a cooperative data storage system concerning data transfer over the IP network. The reason for this is simple, there are not enough public IPv4 addresses available. The trend for the ordinary users, who are the potential clients and contributors of a cooperative data storage system is to use NAT (Network Address Translator). The NAT allows them to easily send outgoing requests, but makes them unreachable for the users outside their local network. This issue is around for a long time and will be around even if IPv6 addresses will be used globally.

[3] http://www.owasp.org/index.php/Man-in-the-middle_attack

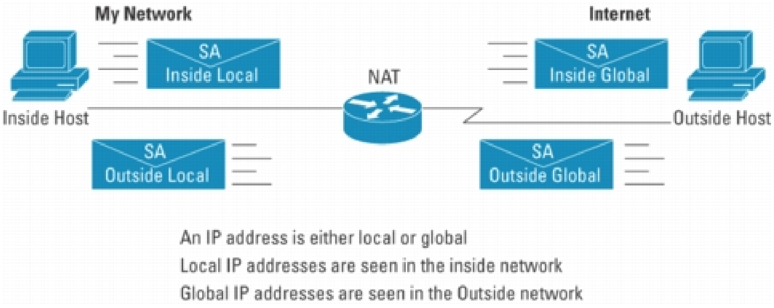

My Network
Internet
SA
Inside Local
NAT
SA
Inside Global
Inside Host
Outside Host
SA
Outside Local
SA
Outside Global
An IP address is either local or global
Local IP addresses are seen in the inside network
Global IP addresses are seen in the Outside network

Figure 3.7: Typical NAT based architecture[1].

This problem is often referred as "NAT traversal problem". Fortunately there are several protocols and techniques for NAT traversal. The following list shows some of the most used methods:

Session Traversal Utilities for NAT (STUN) - The STUN protocol allows the discovery of a NAT device in the network, the public IP address or NAT address and port number, what was allocated for the communication. The protocol requires a third party STUN server, which is addressed by the client application (behind NAT) to determine the presence of the NAT device[7].

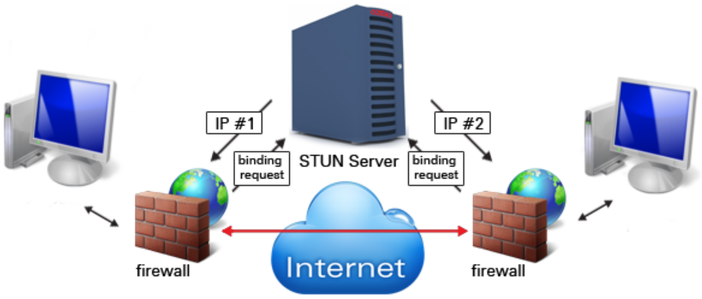


Figure 3.8: STUN based NAT traversal architecture scheme.

Traversal Using Relay NAT (TURN) - The relay based NAT traversal is the most reliable but the least effective peer to peer communication method in a P2P network communication. This technique is based on a middle device, which is on a server side of the network (accessible by everybody), all data transfer is realized through this relay device, by one of the clients uploading the data, and the other downloading from it simultaneously. This method requires the resource of the server, which makes it ineffective[18].

[1] http://www.cisco.com/en/US/technologies/tk648/tk361/tk438/technologies_white_paper09186a0080091cb9.html

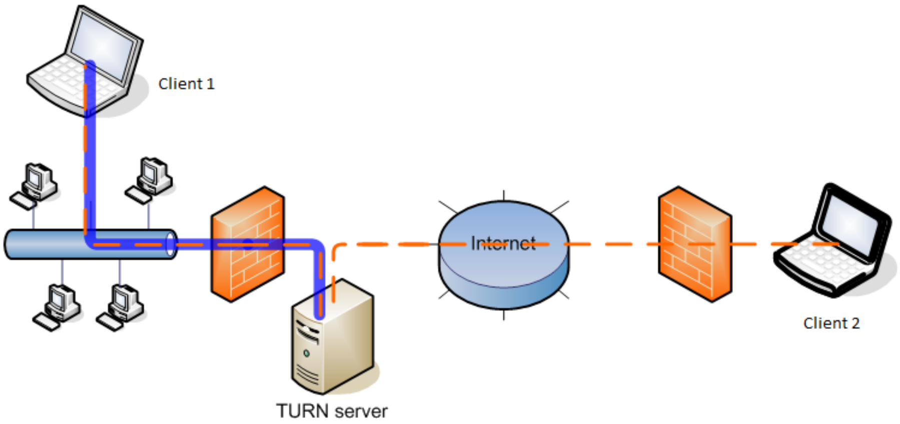


Figure 3.9: TURN based NAT traversal architecture scheme [2].

UDP hole punching - UDP hole punching can be described as follows: with the help of the mediator server, the hosts behind different NATs can obtain each other's private network address and port, as well as the public network address and port resolved by the NAT. Then they use this information to establish a connection to intercommunicate. This traversing NAT technology is simple and robust[18].

UPnP - Universal Plug and Play is a technology, which allows networking and automatic device discovery from various vendors and manufacturers. With the help of UPnP devices can join a network, obtain an IP address and discover other devices. It can automatically create port forwarding rules in the router[19].

[2] wiki.cs.columbia.edu/download/attachments/649/TURN.ppt

Chapter 4

Design of a Time Frame Based Cooperative Data Storage System

In this chapter, the time frame based design of a cooperative data backup solution is discussed. The main focus is given to the mathematical definition of the components of the design. The architecture, what is described is based on hybrid client-server-contributor model. In the center of this architecture is a tracker server, which is controlling the operations carried out by the users in the system and taking care of the fair pricing model. The chapter also describes some of the possible special attacks against the system and defenses against them.

4.1 Agreements Within the Crowd

In order to make the system more flexible, some agreements have to be made between the clients and the contributors.

The basic concept is that the client doesn't need his data to be reachable 24 hours a day and 7 days per week. The client can decide about the time frame, when he needs his data being available. The other agreement that can be made between the client and his partners is that in the specified time frame the data doesn't have to be available with 100% reliability.

In an ordinary storage system (the systems what are well known by the public, see chapter 2) this agreement was not possible to make. In a basic client-server model, regulatory options can't be offered for the clients, with data servers running nonstop. The same way it is possible to describe the solution from the company Symform, they obligate the contributors to run a server at their homes.

In the time frame based system the client can demand for contributory resources, which correlates with his needs. In other words the system will find, only those contributors to back up the data for clients, who are reliably online in that time frame, which was given by the client. And the redundancy rate for the Reed-Solomon algorithm is optimized based on the found contributors and their promised reliabilities. Time frames can be described as a repeating interval of time in a week distribution (e.g. business hours, weekend, night time, etc.) The time frame convention can be useful for employees, who work several hours each day, therefore they don't need their data being present at night. As well as the convention about the reducible reliability can be useful for clients, who just want to back up their data for crash-recovery, which is not for everyday use (photos, videos, etc.). If the client requests his data outside of the defined time frame, the data won't be guaranteed to be present.

Similar specification has to be made from the contributors' side of the system. They have to define the time frame and reliability, when they will be reachable by the clients, and what is the percentage of the reliability they will be online. These specifications of the contributors has to be kept strictly in order to make the files reachable by the clients as they desired. This fact requires an advanced handling of the reliability issue. As well as general file management has to be introduced for cases when contributors decide to leave the system.

4.2 Time Frame, Space and Reliability Model

The data and space model has to be based on the agreements made by the clients and the contributors. From the aspect of the client, the data will carry also information about the time frame and reliability demand. It is possible to define the client's data (file or directory) with three parameters:

- Time frame (demanded)
- Reliability (demanded)
- Size of the data

Analogically the contributed space can be described with three parameters:

- Time frame (promised)
- Reliability (promised)
- Size of the free space

The main benefit from this data and space model is that in theory it can be applied for the vast majority of people around the world. Each user of the system, regardless of he is contributor or client, can specify his parameters according to his needs or availability. These parameters won't bring any handicap to the users of the system, if any client needs his data be present online 24 hours every day, than he is able to set the parameter to this value. Same applies for the reliability parameter. In fact with this model the ordinary online backup mechanism can be emulated as well. Only the time frame has to be set for 24 hours a day and 7 days a week and the reliability has to be set for 100%.

In a more advanced version of this system it is possible to add a fourth parameter to these definitions, which is the bandwidth needed for the data transfer. Some data is modified relatively often, so it needs higher bandwidth than others. For example photos and movies are possible to store with smaller bandwidth, on the other hand work files, which are often changed need fast up and down link.

The following formulas describe the mathematical model of the time frames, the data and contributions.

Let $T = (b, f, d)$ be the set of all possible time frames, where

b is the start time (beginning) of the time frame, which half-hour of the week it starts (the week has 336 half-hours). $b \in \mathbb{N}, b < 336$

f is the frequency of the time frame, how many days it covers. Each week has 7 days, therefore $f \in \mathbb{N}, f < 7$

d is the duration of the time frame, how many half-hours it covers per day. Each day has 48 half-hours, therefore $d \in \mathbb{N}, d < 48$

Any time frame $t \in T$ can be defined as (each parameter of the time frame consist the time frame itself as index)

$$t = (b_t, f_t, d_t) \tag{4.1}$$

The geometrical shape of the time frame space is a torus. This means e.g. if a user starts his time frame at Saturday and it lasts 5 days (variable f), then it ends in Wednesday. Also this is the case for the hours, if a user starts his hours at 20:00 and it last him 8 hours (variable d), then he will end it the **next day** at 4:00. This **overlap** has to be applied for time zone differences as well. The server which stores every users' time frame information, has to convert time frame hours to the local time zone, e.g. in Japan the time frame from 8:00 to 16:00 is not the same as in the USA, New York city, there is 13 hours difference, what has to corrected if the client defines the time frame.

The reason why is the model based on half-hours (not hours), is this stepping provides higher flexibility for the users. While even more detailed description (e.g. minutes) would confuse the users, and in the end wouldn't bring any significant change. Simple visualization of the time frames can be done as shown in figures 4.1 and 4.2.

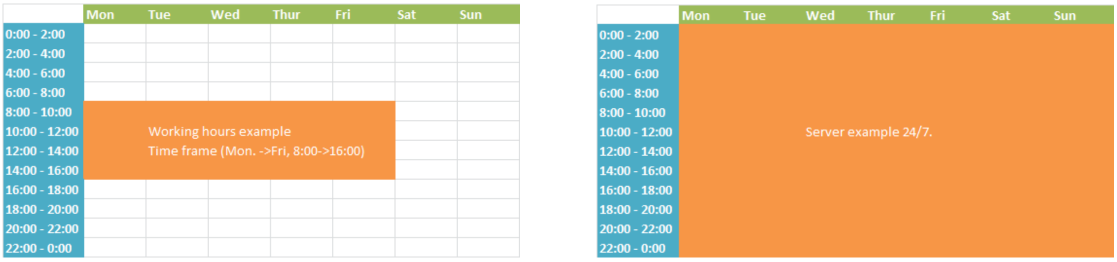


Figure 4.1: Time frame for regular work hours (left) and time frame for server (right).

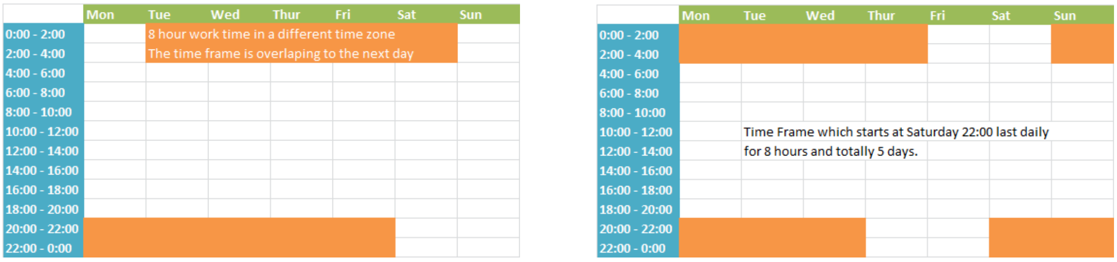

Mon
Tue
Wed
Thur
Fri
Sat
Sun
0:00 - 2:00
2:00 - 4:00
4:00 - 6:00
6:00 - 8:00
8:00 - 10:00
10:00 - 12:00
12:00 - 14:00
14:00 - 16:00
16:00 - 18:00
18:00 - 20:00
20:00 - 22:00
22:00 - 0:00
8 hour work time in a different time zone
The time frame is overlaping to the next day
Time Frame which starts at Saturday 22:00 last daily
for 8 hours and totally 5 days.

Figure 4.2: Time frame which overlaps to the next day (left), time frame which overlaps from one day to another, but to the next week as well (right).

Let $F = (s, r, t)$ be the set of all files in the system, where

s - is the final size of the file $s_f \in \mathbb{N}, s_f \geq 0$

r - is the demanded reliability $r_f \in \mathbb{R}, 0 \leq r_f \leq 1$

t - is the time frame of the file f

Based on the description each file $f \in F$ in the system can be defined as (convention):

$$f = (s_f, r_f, t_f) \tag{4.2}$$

Analogically let the set of contributions be $C = (s, u, r, t)$, where

s_c - is the size of the free space of the contribution $s_c \in \mathbb{N}, s_c \geq 0$

u_c - is the size of the used space of the contribution $u_c \in \mathbb{N}, 0 \leq u_c \leq s]$

r_c - is the demanded reliability $r_c \in \mathbb{R}, 0 \leq r_c \leq 1$

t_c - is the time frame of the contribution c

Based on the description each contribution $c \in C$ in the system can be defined as (convention):

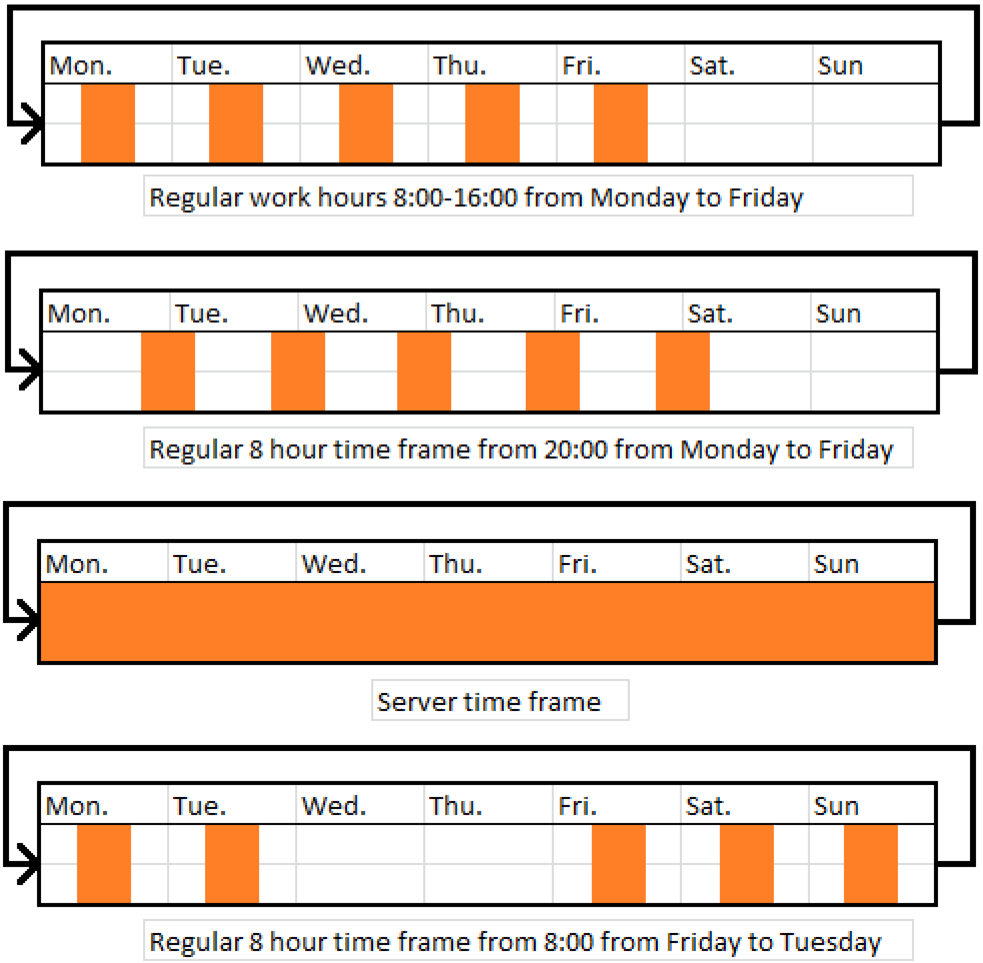
$$c = (s_c, u_c, r_c, t_c) \tag{4.3}$$

4.3 Time Frame Characteristics

Let $\Gamma \subseteq TxT$ be the "cover" relation over the set of possible time frames. This relation is important for many reasons in the design, but mainly because it is used to find the possible partner contributors for data storage for the clients. Only those contributors can store the client's data, who are the same time frame online, or their time frame covers the client's time frame.

$$\begin{aligned}\gamma = \{(q, w) | f_q \geq f_w \wedge \\ & d_q \geq d_w \wedge \\ & (s_q \leq b_w \vee (f_q \geq 7 \wedge d_q + s_q > 48 + s_w)) \wedge \\ & ((b_w - s_q) \bmod 48 + d_w \leq d_q \vee d_q = 48) \wedge \\ & ((b_w - s_q) + 48 f_q + d_q \geq 48 f_w + d_w \vee f_q \geq 7) \wedge \\ & ((b_w - s_q) + 48(f_w - 1) + d_w \leq 336 \vee f_q \geq 7)), \\ & q, w \in T\}\end{aligned} \tag{4.4}$$

In order to explain the working mechanism of this relation it is necessary to introduce an alternative visual representation of the time frames. This representation is not based on a 2 dimensional torus, but a 1 dimensional cyclic field, which basis is the half hours in the week:



Regular work hours 8:00-16:00 from Monday to Friday

Regular 8 hour time frame from 20:00 from Monday to Friday

Server time frame

Regular 8 hour time frame from 8:00 from Friday to Tuesday

Figure 4.3: Examples of time frames based on a 1 dimensional cyclic field.

In figure 4.3 the number of days what the frequented time frame covers is the value f_q, the width of each bar (duration of time frame per day) is the value d_q and the value b_q is the half hours of the week (336) when the time frame starts. This visualization, show that it is enough to deal with only one overlap of time (from Sunday to Monday), which significantly simplifies the equations.

This equation 4.4 can be divided into 6 parts, for better explanation:

$f_q \geq f_w$ - The covering time frame must have higher frequency (last for more days).

$d_q \geq d_w$ - The covering time frame must have longer duration.

$s_q \leq b_w \vee (f_q \geq 7 \wedge d_q + s_q > 48 + s_w)$ - The covering time frame must start earlier, unless it last the whole week and ends after second time frame.

$(b_w - s_q) \bmod 48 + d_w \leq d_q \vee d_q = 48$ - The starting hours of the second time frame plus its duration can't exceed the starting hours plus the duration of the covering time frame, unless the covering time frame has 24 hours duration (48 half hours).

$(b_w - s_q) + 48f_q + d_q \geq 48f_w + d_w \vee f_q \geq 7$ - The ending half hour of the second time frame can't exceed the ending half hour of the covering time frame, unless the covering time frame last for a week.

$(b_w - s_q) + 48(f_w - 1) + d_w \leq 336 \vee f_q \geq 7$ - The last rule says that if the covering time frame covers every day in the week, only in that case can the input time frame relatively to the covering time frame overlap to the next week. Relative overlap means that covering time frame's start time is considered at 0 half hour, and input time frame's start hour is normalized to that. Therefore the relative start time difference is the same as before transformation.

Let $t \in T, G \subseteq C$ than $G = C \geq t$ be the subset of contributions C, where the contribution's time frame can cover t.

$$G = C \geq t = \{a | (t_a, t) \in \Gamma, a \ \in C\} \tag{4.5}$$

Let $G = C \leq t$ be the set of contributions, where the contribution's time frame is covered by t.

$$G = C \leq t = \{a | (t, t_a) \in \Gamma, a \ \in C\} \tag{4.6}$$

Let $h : T \to \mathbb{R}$ be the function, which returns the "surface" of the time frame. In other words this function returns the total amount of half hours for a time frame in a week:

$$h(t) = d_t f_t; \tag{4.7}$$

4.4 Models of Fair Pricing

The system requires a strong economic background, how contributors profit from the provided space and how clients pay for the used services. The main idea is that each client has to pay (transfer credit) to the contributors, who store their data. Without this economic background the system wouldn't work. By contributing the users earn credit, which again allows them to back up their data. Credit works as a catalyst in the system, which motivates the users to contribute in order to get reliable free space over the network by giving up their local unreliable free space.

In this section 3 fair pricing models are presented. The first two concerns about the general or local demand for the time frames to the equations. The last model is the simplest, it returns an absolute price for any contribution. It is based only on the stored data, the reliability and the covered time of the time frame.

One rule applies for each of the models. In these models always the contributor defines the price of his storage, and the client is paying that amount what his partners request. This doesn't mean that the contributors personally can define the price, but the algorithm is based on the parameters (time frame, reliability, size, used space) of the contributors and not the clients (file time frame, reliablity, size). If the contributor's time frame is e.g. "server" (24/7 with 100% reliability), then the client pays for that specific resource, even if he wanted to store only for 8 hours a day his data. The chosen pricing model is responsible for determination of the price of the contribution. **The distribution algorithm is**

responsible for finding the best prices for the client, usually the contributions with smallest surface, what still covers the client's required time frame see section 4.8.

4.4.1 Fair Pricing with General Demand

The first model tries to simulate the real world economics, where if the demand increases for some resources, their price is increased as well.

The rules of the system are that the clients are paying credit based on the parameters of the space, reliability, length of the time frame in half hours, and the **the general demand and offer rate of that time frame in the system**.

Calculating the general demand and offer rate into the pricing makes some of the time frames more valuable and some less. For example the time frames during work hours are definitely more valuable, because more people rely on these time frame and want their data be available. On the other hand time frames during night could be less valuable, because practically less people would use their data during night. Also during the time frames of work hours the possible amount of contributions can be higher as well, therefore the demand not necessarily increases the value of the time frame.

The fact that the payment doesn't depend only on the space and data parameters (size, reliability, time frame), but also on the general demand of storage space should make the system adaptive to changes in the demand or the offer. There are two cases what can happen in the given time frames:

The demand for storage space is closing to the maximum available storage space – in this case the price of the storage space automatically increases.

The amount of storage space is significantly higher than the demand for it – in this case the price of the storage space decreases.

This mechanism is again similar to the real world inflation mechanism. It should be a factor of motivation for the usage of the system. For both sides:

In case of high storage price – Contributors are motivated to provide more space.

In case of low storage price – Clients are motivated to store more data, thus providing more credit to that particular time frame.

The following equations describe the pricing model for file f which is stored in contribution c:

$$q(t_c) = C \leq t_C \bigcup G \geq t_c \tag{4.8}$$

Where $G \subseteq C$, and $q(t_c)$ returns a subset of C, where the time of these contributions cover or is covered by t_c. These contributions are competing for the storage of a data in a given time frame, therefore the general demand for a time frame can be calculated based on these contributions.

$$\lambda(t_c) = \frac{\sum_{c' \in q(t_c)} s_{c'}}{\sum_{c' \in q(t_c)} s'_c - u'_c} \tag{4.9}$$

$$price(c, f) = s_f s_c r_c h(t_c) \lambda(t_c) \tag{4.10}$$

Where λ - function returns the general demand for the time frame, what is the argument of this function. It returns the summed rate of remaining free space and total space of the contributions, what the time frame t_c can cover or can be covered by. Contribution who can cover or can be covered by the desired time frame, actually compete with each other for storing the data, this is the reason calculating the general demand for a time frame based on their used and free space ratio. This means as the remaining space get closer to zero in a given time frame (t_c), the price will go up with hyperbolic character. Analogically as the remaining free space raises the price goes down. **The price of a contribution is based on its concurrent contributions**.

This model of pricing has two mayor disadvantages:

- **Performance** - as lots of contributions has to be calculated to the equation, it can be hard to compute in practice.
- **Vulnerability** - at first sight this model seems fair, but it can be easily abused. Let's say an attacker wants to lower the price of the contributions at a specific time frame. Simple he can pretend that he has huge amount of free space what he can contribute at that time frame. This will lower the price of all other contributions, so the attacker will be able to store for lower price. It is possible to defend against this attack, by defining and upper limit for contribution size.

4.4.2 Fair Pricing with Local Demand

This model tries to overcome the vulnerability issue of the pricing model with general demand. It removes the possibility to reduce the price by adding a big contribution to a time frame.

$$price(c, f) = s_f r_c h(t_c) \frac{s_c}{s_c - u_c} \tag{4.11}$$

This model doesn't calculate the concurrent contributions to the price, but the contributions define their price based only on their parameters. This function is hyperbolic, therefore the price will rise with hyperbolic character as the used space gets closer to the total space. This is a natural behavior; as the resources run out their price raises. This model has one advantage over the previous, that it removes the performance issue.

Unfortunately by adding small contributions to the system this model can be abused again. As the small contribution fills up faster, the credit gained for the contribution raises faster as well. This means an attacker can simply create a contribution of a few MB and as it fills up, the price gets closer to infinity, because of the hyperbolic character. Then this credit can be used to store data in other time frames. To defend against this attack a lower bound for contribution has to be defined.

4.4.3 Absolute Pricing Model

The last pricing model, what is presented doesn't take into consideration the remaining space and total space ratio. This leads to an absolute price, which is the same under all conditions (time frames):

$$price(c, f) = s_f r_c h(t_c); \tag{4.12}$$

The income of a contributor is only based on the size of the files, the surface of the time frame and its reliability. This system can't be abused by creating a specific contribution. It is as well easy to compute. The only problem with this pricing model is that it creates dominant time frames. Some time frames are more popular than others. The contributors of these time frames therefore earn more credit. This can cause, they can easily fill up with their credit the less popular time frames, not leaving space for other users. This can be a huge handicap for users who can't contribute at the dominant time frames. Pricing models based on general or local demand don't have this issue, because they have hyperbolic character proportional to the demand.

4.5 System Characteristics

In this section the chosen architecture of the system is presented. The choice was made for the (hybrid) server based cooperative data storage system as was presented in chapter 2. The reason choosing this architecture is simple, this is the easiest way to implement a cooperative data storage solution which relies on peer-to-peer communication. It has three main components:

Client – Is the side, who wants to store the data.

Server – This side stores all the necessary information about peers (clients or contributors), pricing model, distribution algorithm, does the statistical analysis of the system, monitoring, etc.

Contributor – Is the side, who provides free space to the system with specified reliability and time frame.

This design is based on erasure coding reliability model to ensure the desired reliability of the data. This means each file what is selected for storing has to be processed by an erasure coding algorithm. This results in creating $k + m$ fragments from the base file, from which any m fragment can be missing and the original data can be still reconstructed. Each fragment has to be distributed to a different contributor to achieve the highest diversity and probability of reconstruction.

4.6 Data Storing Process

The data storing process starts on the client side, who wants to store his files or directories on a reliable backup storage. The client selects his data in the operation system for synchronization. Then he chooses the parameters of the backup. These parameters are the time frame and reliability as described in section 4.2 (the size parameter is the parameter of the file itself).

In the next step the client sends a request to the server with the chosen parameters about the backup. The server tries to find contributors who can ensure the backup based on the clients requirements. If the search process finished successfully the server sends back the information about the contributors to the client. After the response, the process continues with the encryption of the data. The encrypted data then is processed by the erasure code algorithm, which generates various amount of fragments depending on the specified parameters and the responded contributions (see section 4.8). This all happens behind the client's firewall on the client's computer. As the client already has the set of contributors, it starts to share the data with peer-to-peer protocol. Each contributor has to verify the transfer before saving the data to the local drive. After the data is transferred successfully to the contributors the client acknowledges it to the server. The server stores the dedicated contributors for the file fragments in the database.

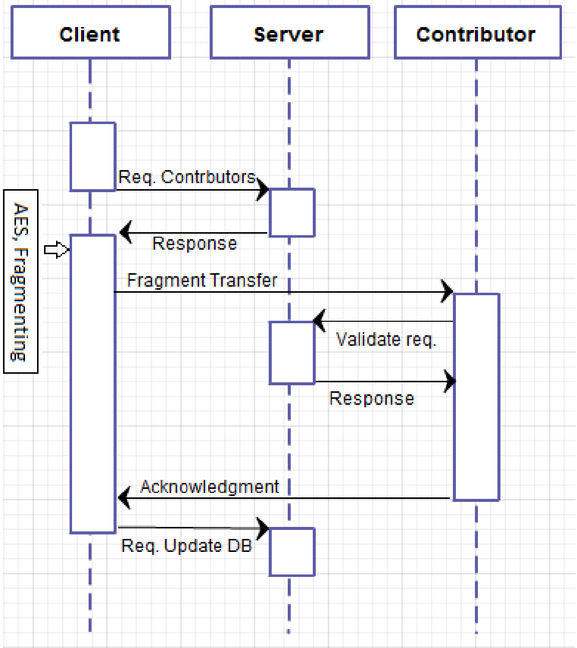


Figure 4.4: Sequence diagram of the data storing process

4.7 Data Retrieving Process

If the client's data is shared within the contributors, and there are at least k contributors of the $k + m$ (who stores the $k + m$ fragments) for data which was distributed, than the data can be retrieved (downloaded). The data retrieving process starts by asking the server about the available contributors who has the relevant data fragments. Server sends back a list of available contributors, with who the client will establish connection. The client's device will ask for establishing a secure peer-to-peer transfer protocol between him and the contributors. Each contributor verifies the request of the data, by asking the server about the permission (or other authentication method can be done). As the client retrieves the encrypted data fragments it orders them to the original sequence and runs the erasure decoding algorithm, which will generate the original, but still encrypted file. The decryption algorithm is applied on this file with the same key what was used for encryption, which will result in the original file.

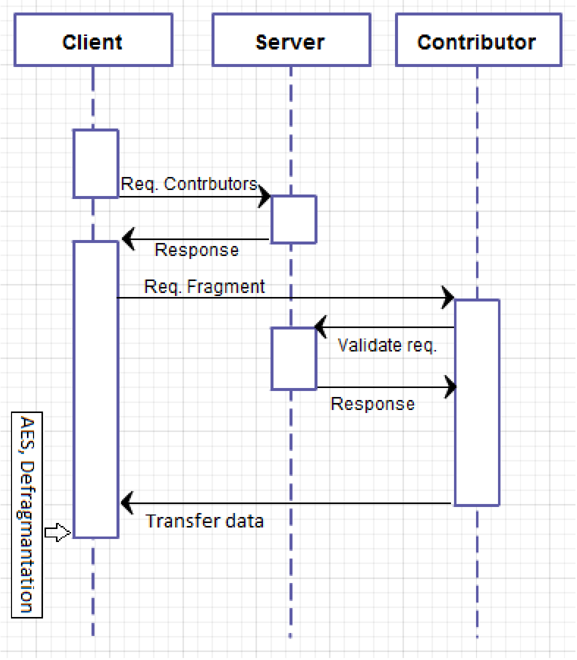


Figure 4.5: Sequence diagram of the data retrieving process

4.8 Contributor Dedication Process

The main goal of the contributor dedication process is finding a list of contributors, who can store the client's data. As it was discussed before, erasure coding is used to ensure the reliability. This process is partially done on the server side, but the majority of work is done on the clients side. The dedication process starts when the client selects a file or directory to store in the system. The server chooses a list of contributors for the time frame, what was set for the data. The relation $G = C \geq t_c$ (see equation 4.5), is usable to find the appropriate contributions. This function return every contribution, which has time frame, what can cover the time frame chosen for the file.

After the contributions are selected on the server side, the client is informed about them. The client has the responsibility to choose the ones ($G' \subseteq G$), which fits its needs concerning reliability. The client chooses G' set of contributions from the list (best those which are online at that time, in order to be able to send the data immediately), where $|G'| = n$. The client for the erasure coding uses the n value as $k + m = n$, assuming all of the chosen contributors will hold a fragment.

A complex issue remains before transporting the data to contributors, which is how to define the ratio of k and m. Finding the appropriate values for k and m is an extremely hard process, it can be described as the following:

As defined in section 4.2 each contribution has different probability $r_{g'}$ of being available at the time frame $t_{g'}$, $g' \in G'$. The goal is to set the values $k + m = n$ to a value which will result in $P_{desired} \leq P_{online}$, where $P_{desired}$ is given by the user and P_{online} has to be calculated from the selected contributions G'.

P_{online} can be defined as the following, if $\forall g' \in G' : r_{g'} = R$, where R is a constant:

$$P_{online} = \sum_{z=k}^{n} \binom{n}{z} R^z (1 - R))^{n-z} \tag{4.13}$$

This equation is based on Bernoulli binomial probability[12], it calculates the probability of **at least** k events success out of n trials. Success event refers to "being online" in the context of the cooperative data storage system. The problem is that it works only for events with equal probabilities, but the real case is that each contributor has different probability being online. The issue what has to be solved can be formalized as the following: what is the probability of exactly k or more than k contributors are online, if all the $k+m$ contributors have **different** $r_{g'}$ probability being online. To visualize this issue, the following equation shows calculation of probability of **at least** 4 success events out of 5:

$$\begin{aligned} P =& abcde+ \\ \\ & abcd(1-e)+ \\ & abc(1-d)e+ \\ & ab(1-c)de+ \\ & a(1-b)cde+ \\ & (1-a)bcde \end{aligned} \tag{4.14}$$

The equation 4.14 has complexity of $1 + 5 = 6$ probability calculations, which comes from $\binom{5}{5} + \binom{5}{4}$. Now this example in our context means to calculate the probability at least 4 contributors of 5 are online. To further demonstrate the complexity, let's assume 16 contributors and at least 8 being online from them. The complexity of the function will be

$$\binom{16}{16} + \binom{16}{15} + \ldots + \binom{16}{9} + \binom{16}{8} = 1 + 16 + \ldots + 11440 + 12870 = 39203,$$

which is far not practical to compute for each file what is selected for backup. In practice the system is designed for thousands of users, so equations can come up to 100 contributors with probability of at least 80 being online. This problem is called the Poisson binomial probability distribution[12].

4.9 Poisson Binomial Probability Approximation

The only sensible solution is to approximate these equations. The simplest approximation is done by simplifying the Poisson binomial distribution to the ordinary (Bernoulli) binomial probability function and setting the constant R to the average of all the $r_{g'}$ values where $\forall g' \in G'$.

$$R = \frac{\sum_{g' \in G'} r_{g'}}{|G'|} \tag{4.15}$$

Now as R value is calculated, it can be used in a simple Bernoulli binomial probability distribution (see 4.13).

Another way to approximate Poisson binomial distribution, for exactly k success events is with discrete Fourier transformation[4]:

$$\Pr(K = k) = \frac{1}{n+1} \sum_{l=0}^{n} C^{-lk} \prod_{m=1}^{n} \left(1 + (C^l - 1)p_m\right), \tag{4.16}$$

Where $C = \exp\left(\frac{2i\pi}{n+1}\right)$
Where $i = \sqrt{-1}$
Using this approximation is left for further research in this topic.

4.10 Poisson Binomial Probability Approximation Error Rates

The approximation of the Poisson binomial probability distribution as described in section 4.8, is not precise. The exact mathematical determination of the error rate is out of the scope of this thesis, however practical measurements were done in order to find out the error rate of this approximation. The measurements were done for various amount probabilistic events, with different random probabilities for every event. For each test 500 attempts were done. Figures 4.6 and 4.7 shows the average error rates for different amount of events, with different value of exact occurrences.

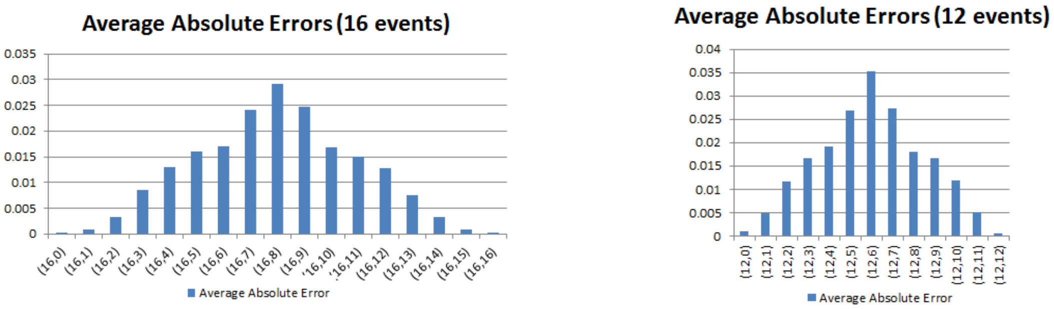


Figure 4.6: Graphs showing, the average error rate of approximation the Poisson probability distribution for 16 and 12 events.

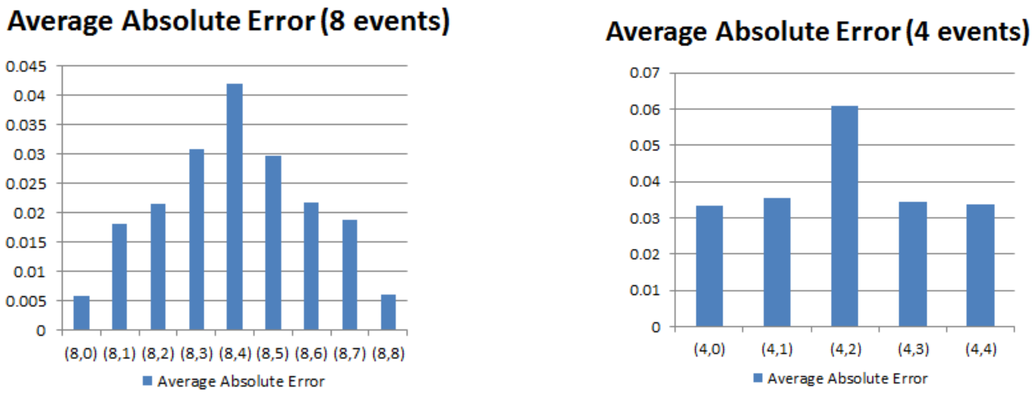


Figure 4.7: Graphs showing, the average error rate of approximation the Poisson probability distribution for 8 and 4 events.

From these measurements it is clear, the exact number occurrence probability approximation for different random events is less precise with fewer events. The highest error rate is with 4 events, where the probability of exactly 2 events occurrence is searched. By increasing the number of events the error rate decreases, which can be considered as good results, because in the cooperative storage system the events represent contributors with different probability being online. The system is designed for high number of the contributors, therefore this approximation is usable for it.

For fewer number of contributors the default equation can be used (see section 4.8). This equation gives back the precise probability but it is considered as not efficient for more than 20 events, because of its exponential computation complexity (number 20 was found during the tests).

4.11 File Maintenance Techniques and Processes

As it was mentioned before the cooperative data storage system relies on set of unreliable devices. The unreliability comes from the fact that the users are not available on the internet always, but there is a second factor of unreliability, which is the case when some of the fragments are lost completely or were removed from the contributors' devices. The reasons can be for example contributor device crash, software uninstallation, fragment physically removed by the contributor, etc.

These cases have to be treated with the process called data maintenance. The simplest solution for this issue is to recalculate the missing fragments from the remaining ones, but first the data lost has to be detected. The detection can be done by monitoring of the fragments. If the contribution, which holds the fragment was not online for certain amount of time, it can be considered as lost. For the lost fragments a new reliable contributor (best with the same reliability, as the contributor who removed the fragment or quit the system) has to be found with the already proposed methods of contribution dedication. After the contribution is found a command is send to this contribution to recover the missing fragment from the available ones. This is done by downloading the necessary fragments and with the erasure coding algorithm the missing one is reconstructed. There is no security threat for the data, because the original erasure coding of the data was done after the encryption, so the new contribution even if it has all the necessary fragments can't decode the original data.

After the relocation of the fragment, the database has to be updated, so the client can download the fragments from the new location.

Chapter 5

Implementation of the Prototype and Used Technologies

As described before, the system has free layers: the client layer, the contributor layer and the server layer. Each of them is connected through the regular IP network. The client and the contributor layer are together in one simple desktop application handling the commands, which are given by the user or the server. These two layers have to be constantly connected to the server. This connection is necessary because the server works as a general controller of the architecture. It is able to send commands to these layers in order to control the data flow and manage the system. Figure 5.1 shows detailed connections between components of the system.

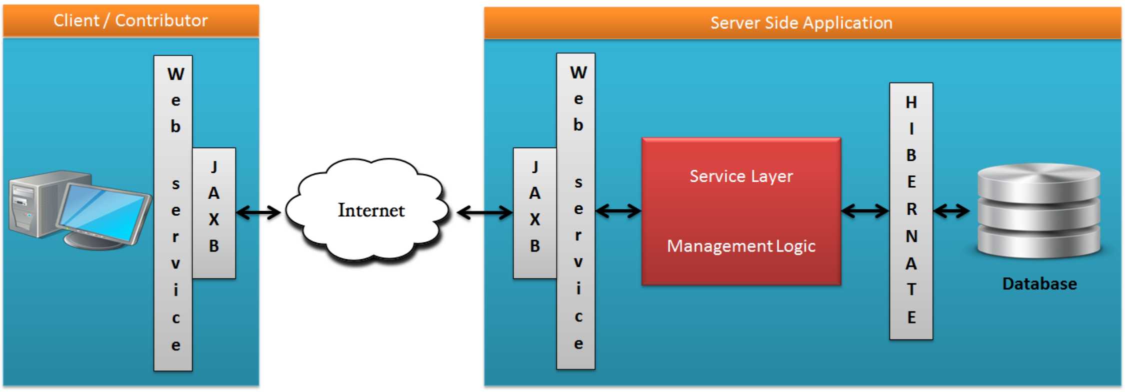


Figure 5.1: System Architecture Scheme

As shown in figure 5.2, the project is divided to 4 modules. These 4 modules inherit settings and parameters from their parent module (Parent), where the version numbers of the project dependencies are held. Also the commonly used libraries are defined in this module.

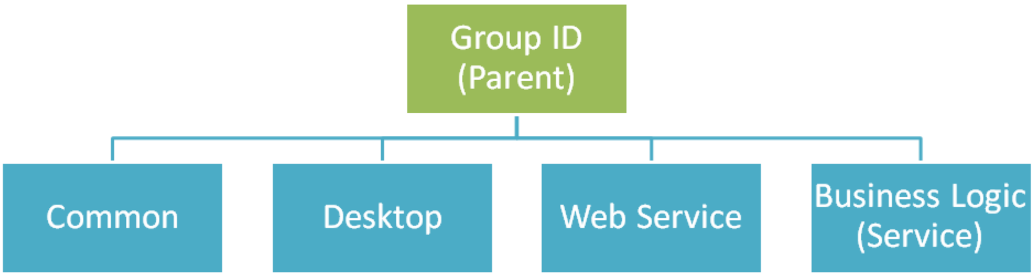


Figure 5.2: Project architecture scheme (Maven hierarchy).

5.1 Server Side Application

The server side application consists of three mayor parts.

- **Database** - where the information about contributors, clients and data is stored.
- **Management logic (service layer)** - does the management of the data according to the requests done by the users via web service.
- **Web service interface** - is the layer, how the users can reach the database, all the requests are done via this interface.

The server side is implemented in Java EE (JDK 1.6), with Spring framework[1]. The communication to the database is done with Hibernate[2] implementation of the Java Persistence API, which allows transparent interactions with the database. The database server is Postgre[3], but due to the usage of Java Persistence API it can be changed without mayor change to the code of the application. The web service is based on the Spring framework as well (Spring Web Service), which allows also transparent transformation of Java objects to Soap XML messages for the web service. The transformation of objects to XML messages is called marshalling or serialization, which in the implementation is done by JAXB[4] framework.

Upon the Hibernate framework, Spring Data JPA is placed as well, this module offers easy transaction management and generalized database access repositories, which allow developers to completely avoiding to write SQL scripts.

5.2 Web Service

The web service layer creates an additional interface between the management logic and the client side application. The communication protocol is based on SOAP[5], which consist of small XML message transfers between the client and the server. In the prototype many type of messages are present (see table 5.1).

[1] http://www.springsource.org/
[2] http://www.hibernate.org/
[3] http://www.postgresql.org//
[4] http://jaxb.java.net/
[5] http://en.wikipedia.org/wiki/SOAP_%28protocol%29

Request	**Description**	**Response Description**
Get Contributions	Requests a page of contributions from the database for uploading the data.	Returns a page (max 32 pieces, with cheapest price) of contributions, those who are currently online.
Update Contribution	Requests to update a contribution for the user.	Returns the updated contribution or in case error message.
Delete Contribution	Requests to delete a contribution from the database, and all the fragments connected to it.	
Get Files	Requests a list of files, chosen by the user with the corresponding fragment and contribution information.	Returns the list of files and the additional information encapsulated to the file.
Update File	Requests to update a list of files, usually called before uploading the file.	Returns the same list of files, but consistent to the database.
Delete File	Requests to delete a list of files.	
Update Fragment	Requests to update a fragment for a file which is currently uploaded.	
Update Configuration	Requests to updated (create) a new configuration (Time-Frame and Reliability) for the user.	Returns the consistent configuration.
Delete Configuration	Requests to delete a configuration from the database.	
Register User	Requests to register a new user in the database.	
Sign In User	Requests for sign in.	Returns every consistent information for the user from the database (Files, Contributions, Configurations).
Sign Out User	Requests to sign out the user, it is used only for logging purposes, there is no session management in web service.	
Get Status	Periodical polling request of the status updates of the user, and updating the database about the user being online.	

Table 5.1: SOAP requests and responses and their descriptions.

5.3 Database Architecture

The database architecture is described with the ER-diagram 5.3.

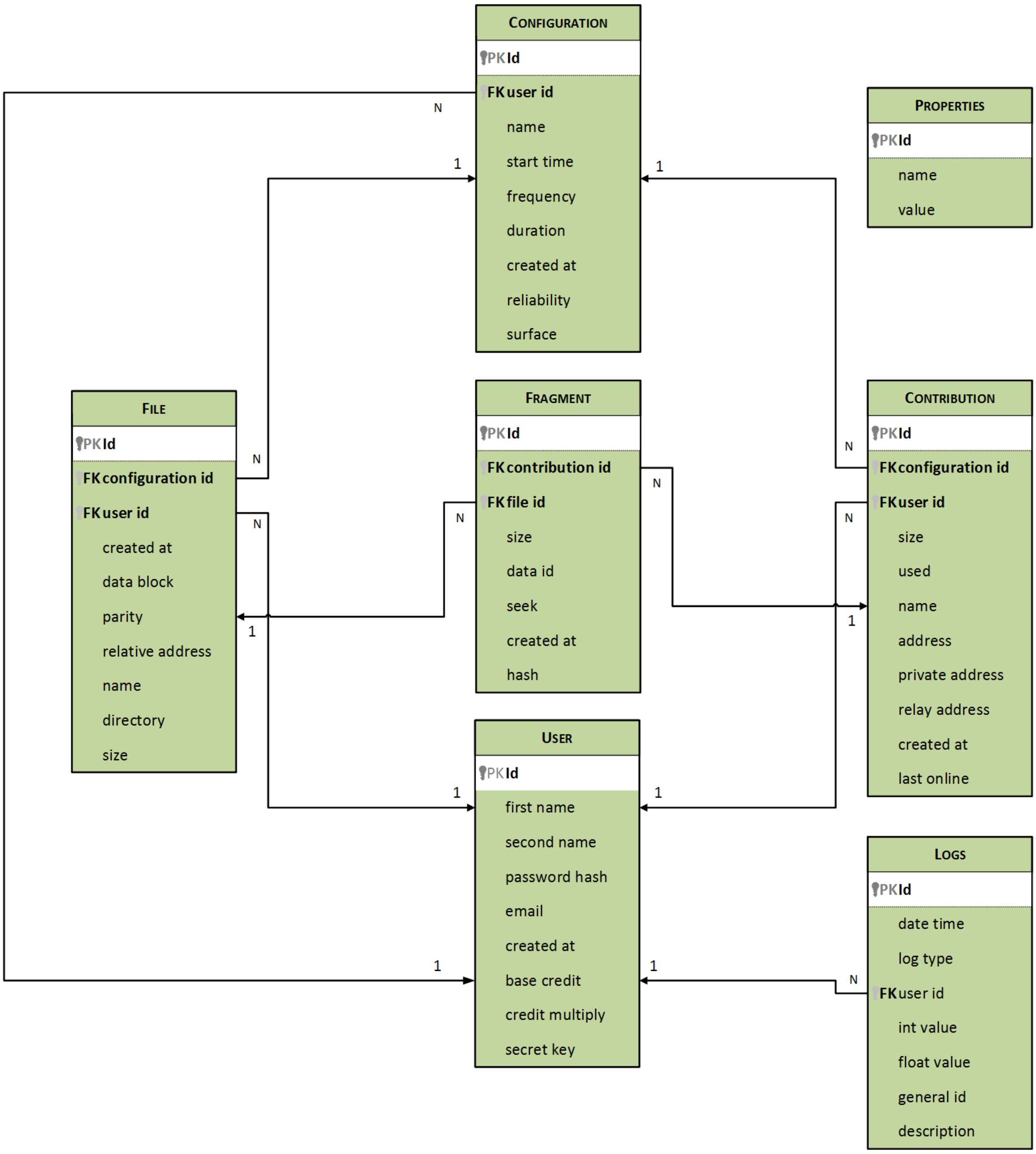


Figure 5.3: Database ER-diagram

The figure 5.3 shows the entities and their connections of in the database. The key features of these tables are the following:

Contribution – This table holds the address, size, configuration, reliability of the contributions of each user. The distribution algorithm works based on these fields with the relation of the File table. When was the contribution last time available is also stored in this table, if it is less then 20 seconds before current time, the contribution is considered being available.

User – This table holds all the information about the users as like: name, email address, password hash, secret key for the encryption, etc.

Logs – The logs table is used for tracking the important actions, what were made (generally requests) in the system. As well as statistical analysis can be made about the system with the help of this table.

File – The file table contains the references to the files of the user. Also it contains the time frame information with the demanded reliability. The directory field indicates if the file is a compressed directory or not. This table also holds the size of the each file. The reference to the contribution table indicated where the fragment can be found.

Fragment – The fragment table is used to keep track about the file fragments in the system, how big are the fragments and which part of the real file they hold.

Configuration – This table meant to keep track of the time frames used be the users, as well as the reliability for each time frame.

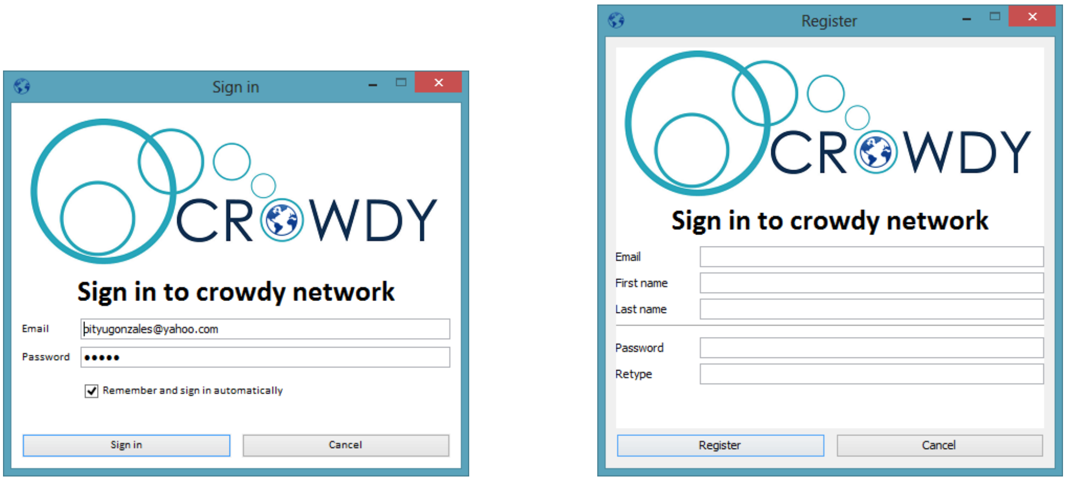
Properties – This table contains the server parameters which can change by time. These parameters are used in the applications as constant variables.

5.4 Client Side Application

The client side application was designed to be user friendly, yet still allowing the users to do every mayor action what is necessary (see table 5.1) The application is a Java SE application based on Swing[6] graphical user interface library. The core of the application is a tray icon, where from the users can open the dialogs to manipulate their profile, register new account, sign in, etc.

The usage process starts with the sign in of the user. If the user have already signed in once and he checked the “Sign in automatically” checkbox, then this process is done after the application start up. Otherwise the user has to sign in manually:

[6] http://docs.oracle.com/javase/tutorial/uiswing/



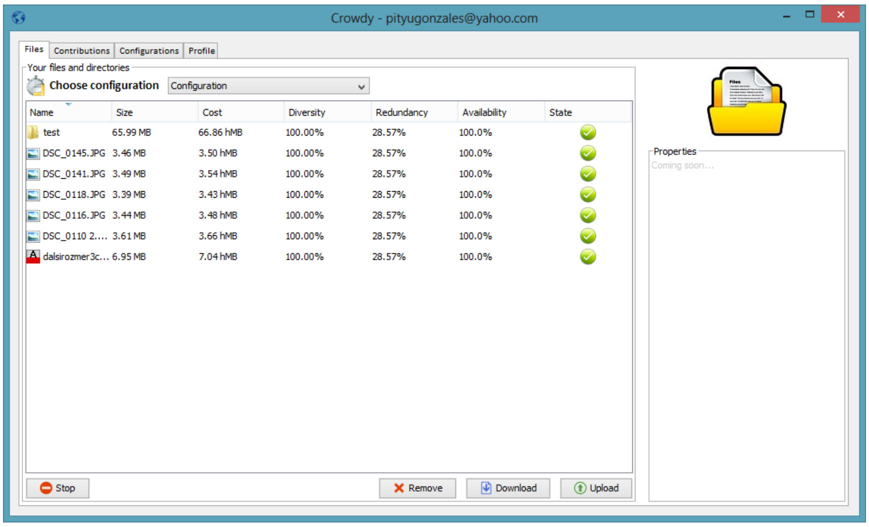
Email pityugonzales@yahoo.com

Password •••••

☑ Remember and sign in automatically

Figure 5.4: Sign in and Register dialog of the client side application

The second dialog on the figure 5.4, allows the user to register new profile. After the user has signed in, he is able to see his files and other properties. The following dialog shows the user's uploaded files, their availability, redundancy rate of the erasure coding (Reed-Solomon), the configuration where the files are held. If the users wants to upload a directory, the directory is first compressed into one zip file and this file is uploaded.

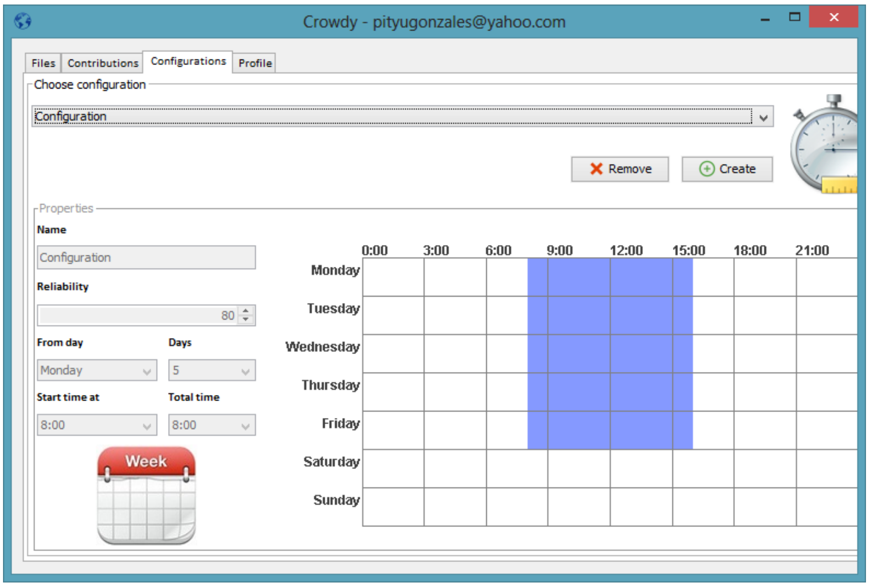


Choose configuration: Configuration

Name	Size	Cost	Diversity	Redundancy	Availability	State
test	65.99 MB	66.86 hMB	100.00%	28.57%	100.0%	✓
DSC_0145.JPG	3.46 MB	3.50 hMB	100.00%	28.57%	100.0%	✓
DSC_0141.JPG	3.49 MB	3.54 hMB	100.00%	28.57%	100.0%	✓
DSC_0118.JPG	3.39 MB	3.43 hMB	100.00%	28.57%	100.0%	✓
DSC_0116.JPG	3.44 MB	3.48 hMB	100.00%	28.57%	100.0%	✓
DSC_0110 2....	3.61 MB	3.66 hMB	100.00%	28.57%	100.0%	✓
dalsirozmer3c...	6.95 MB	7.04 hMB	100.00%	28.57%	100.0%	✓

Figure 5.5: Files tab

The user is able to add new configuration through the "Configurations" tab:



Choose configuration: Configuration

Properties

Name: Configuration

Reliability: 80

From day: Monday

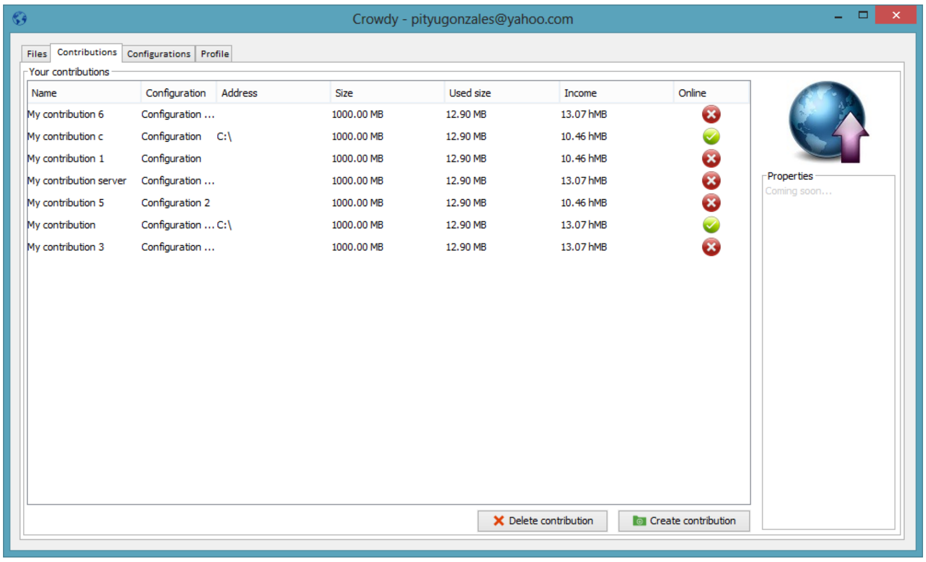
Days: 5

Start time at: 8:00

Total time: 8:00

Figure 5.6: Confgurations tab

This tab contains the time frame and the reliability information of the configurations. On the right bottom part the time frames are visualized as shown in the section 4.2. By clicking on the create button the user adds a new configuration to his profile.



Your contributions

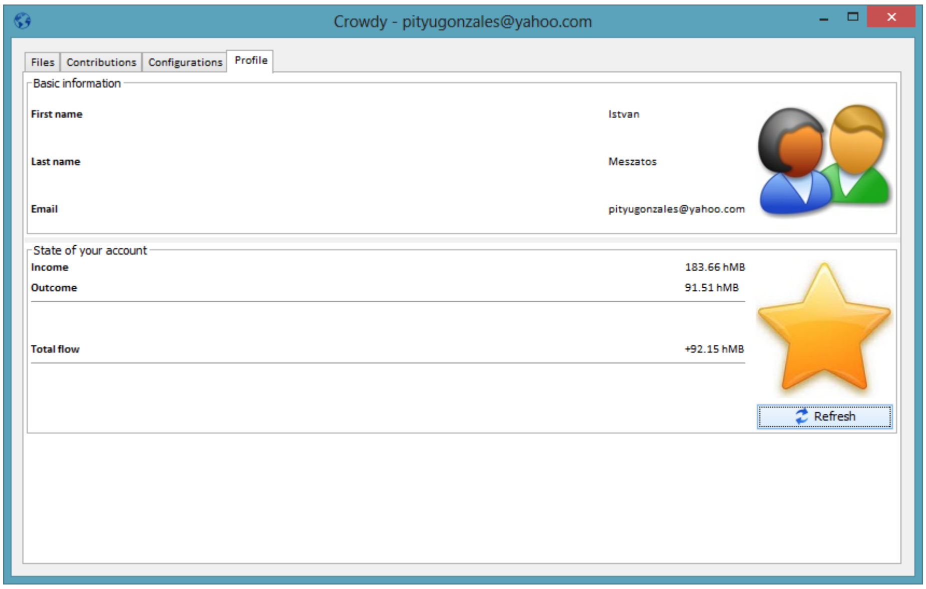
Name	Configuration	Address	Size	Used size	Income	Online
My contribution 6	Configuration ...		1000.00 MB	12.90 MB	13.07 hMB	✗
My contribution c	Configuration	C:\	1000.00 MB	12.90 MB	10.46 hMB	✓
My contribution 1	Configuration		1000.00 MB	12.90 MB	10.46 hMB	✗
My contribution server	Configuration ...		1000.00 MB	12.90 MB	13.07 hMB	✗
My contribution 5	Configuration 2		1000.00 MB	12.90 MB	10.46 hMB	✗
My contribution	Configuration ...	C:\	1000.00 MB	12.90 MB	13.07 hMB	✓
My contribution 3	Configuration ...		1000.00 MB	12.90 MB	13.07 hMB	✗

Properties

Coming soon...

Figure 5.7: Contribution tab.

On the contributions tab the user us able to see his contributions status, how much profit he has from each contribution. This tab also indicates whether the contribution is currently online or not.



Basic information

First name Istvan

Last name Meszatos

Email pityugonzales@yahoo.com

State of your account

Income 183.66 hMB

Outcome 91.51 hMB

Total flow +92.15 hMB

Figure 5.8: Profile tab.

On the profile dialog the user can see his account status, his total income and outcome in MBs. And by clicking the refresh button this status is updated to a consistent version with the database.

The continuous data consistency is ensured with polling technique. Every 10 seconds a **Get Status** request (see table 5.1) is sent to the server to inform the server that the user is online and his contributions are as well reachable. Every 60 sec a new **Sign In** request is sent to the server to update every changed information. The reason for this is that some data for the user can be changed by other users. This data can be:

- **File outcomes in MB** - how much real MB a file costs.
- **Contribution Income in MB** - how much real MB a contribution produces.
- **Contribution used space** - how much space is used from total contributed.
- **File availability** - what percent of fragments is available.
- **File diversity** - how efficient is the file distribution.

5.5 Contribution Dedication Process and Price Model Implementation

The key part of this work is how to design the distribution algorithm. Generally the algorithm has to minimize the cost of storing the data, yet equally distribute the data between contributors. The contribution selection process is based on the coverage function $G \geq t, G \subseteq C, t \in T$, as presented in section 4.3. The first step is to select all the possible contributions, which can store the data, and cover the stored data's time frame. From the selected list of contributions, the ones who are not online has to be filtered out. The others are sorted by their relevance to the data's time frame, this means the contributions with time frames, which are similar to the data's time frame are preferred. But as well as the used and total space ratio is calculated to the dedication process, due to equal distribution of data.

$$V = r_c \frac{s_c + 1}{(u_c + 1) * h(t_c)} \tag{5.1}$$

Where $c \in C$.

This equation is implemented in the database directly. It ensures load balancing of data among contributors, as well as finding the minimal coverage of the contributors time frame, which ensures lowering the price of the data.

5.6 Client Application Structure

The client side application consist of 6 mayor layers (MVC design pattern, data transfer, file access and web service):

- The view layer - which is responsible for visualizing the data.
- The controller layer - which is responsible for human interactions and change of the data and is usually implemented in the same java class.
- The model layer - which is a consistent copy of the current state of the database for the user. Not the whole database is held in the client side, but only the necessary information what is important for the user, and also only the information what he is granted to see.
- Web service interface - this interface is responsible to keep the database consistent to the clients interactions, but also keep the client model up to date.
- Data transfer layer - which is responsible to send and receive data from the contributors.
- File access layer - this module is responsible for the file operations, Reed-Solomon codes, encryption, interleaving and hash value calculation.

Each of these layers has its own package in the source code (under module Desktop). The class MainSession is the starting point of the application, it holds the current user who is signed in and also many data, which is necessary for working of the layers. Although the application structure is highly decoupled, due the usage of Spring dependency injection. This allows that the code doesn't contain important language level connections between layers or classes. This makes the application flexible for refactoring, also easier to test with JUnit. The main operations of the application are parallel, which is implemented with the thread pool design pattern (no more than 20 threads can be created for data transfer and the coding algorithms, if in case there are more request for the thread pool, the calling thread is blocked).

5.7 The Common Module

The Common module contains, all the data access objects (DAOs), which are used by the hibernate API, these object as well are referenced for JAXB for the web service, so the data base communication and the web service communication is implemented with same classes. This makes the code easy to understand and refactor again. This module is linked to all other modules with Maven, so the same classes from this module on the client side, as well as on the server side.

Some the web service messages are not based on the data access objects, or they contain other important information. These classes are stored in this module as well. The ObjectFactory class is responsible for instantiation of these classes for JAXB.

5.8 NAT Traversal Implementation

The internet communication issue, known as the NAT traversal problem (introduced in section 3.10), is partly solved in this prototype. STUN protocol was implemented in the server side, which determines if the client is behind NAT or not.

This is done by sending all the IP addresses with the RequestState message, what the client application knows about itself (for each of its IP interface). When the server receives the message, it knows the source address of the message (which has to be a public IP address), than this address is compared the list IP addresses sent by the client. If any of the interfaces of the client has the same IP as the source address known by the server, than the client is not behind NAT (Although it can be still behind firewall, which makes him unreachable).

If the client is behind NAT, he can't contribute, unless in case if he sets port forwarding on his router. This is an uncomfortable solution for NAT traversal, because the client has to set port forwarding manually on the router. Unfortunately there are no reliable open-source solutions for Java for UPnP protocol, or UDP/TCP hole punching. Relay based NAT traversal (TURN) is possible to implement, but it would be unfair with the users with public IP address, because all the data transfer would done through their device, using their bandwidth. This haven't been implemented yet. The prototype in this thesis implements only the port forwarding solution of NAT traversal.

Most of the future work on this application concerns the NAT traversal issue.

5.9 Technologies and Frameworks

In this section the frameworks and libraries are presented, what were used to implement the prototype. The focus was given to the maintainability and scalability of the system, therefore the technologies have been chosen according to that. The whole system (covering each layer) is based on Java EE and Java SE. The following technologies are used on the server side:

Postgre DB - This database server is used mainly, because it is free and has a reliable support.

Maven - Maven is a project management tool, which does the compilation, automatic tests and other actions necessary to develop the application. The main benefit from Maven is the automatic transitive dependency resolving.

Hibernate - Hibernate is used as the implementation of the Java Persistence API. Hibernate gives us the flexibility to change database server without the necessity of making changes in the code.

Spring Data - Spring Data library is used with Hibernate to access data in database (DAO).

Spring WS - Spring WS is used to implement the REST web service infrastructure for communication between the clients and the server. It creates a transparent infrastructure where XML parsing doesn't have to be done by the developer.

JAXB - This library is used to marshal and un-marshal the objects to and from XML for web service communication.

On the clients side the following libraries and technologies were used in the implementation:

Java.Security, Java.Crypto - These libraries are used to cover the cryptographic aspects of the system, like AES encoding and decoding.

Spring WS - Spring WS is used to implement the REST web service infrastructure for communication between the clients and the server. It creates a transparent infrastructure where XML parsing does not have to be done by the developer.

JAXB - This library is used to marshal and un-marshal the objects to and from XML for web service communication.

These libraries and frameworks have reliable support and are well documented, so maintenance of the project can be easily achieved. Also JUnit framework was used to test the application.

Chapter 6

User Feedback, Tests and Measurement Results

In this chapter subjective and objective results are presented about the system. The prototype was published over the internet to test users. As well as two surveys were done for receiving feedback about the concept. The first survey was done in order to find out current data backup and storage trends among the everyday users. The results were used in a calculation, which simulates the behavior of the system. The second survey was done for receiving feedback from the test user about the usability, missing features and concept evaluation in general. The system has been tested in real usage for 10 days by 23 users. Some of the users were using the prototype constantly with contribution of their space, some user who were not technically able to contributed (NAT traversal issue). These users tested download, upload and general usage of the system to the system.

6.1 Real Time Testing

The prototype and the concept itself was published through various channels for the wide public. In this section the test session and the feedback from the test users is described.

The prototype application was published as a portable java web-start application, from the website: **www.crowdybackups.com/global**. This page was propagated through social media and several blog sites, but most of the test users where found personally. Due to the web-start based publication, the test user received updates of the software each time when an error was corrected.

The application was downloaded by 23 test users. The 23 test users created 15 contributions. This contribution set in total allocated 89GB of distributed space. Together the 23 test users uploaded 1053 MB of data in exact 50 files. For the 50 files 418 data fragments were created by the Reed-Solomon algorithm.

In the attachments detailed measurements are presented about the test users' behavior in the system. The measurements contains, the real time changes in contributor space, file reachability and user credit revenue (income and outcome), based on the 3. pricing model presented in section 4.4.3.

18 of the test users gave feedback through a survey, which consisted questions about the application performance, user interface usability, missing functionalities, comparison with Symform's concept and comparison with the ordinary online storage system concept. The

answers had to be set as a rate from 1 to 5, where one meant negative user experience or unimportant feature and 5 meant positive user experience and very important feature for the user. The results of the user feedback for the usability and user experience are shown in figure 6.1. Subjective importance of the features of an online storage system are shown in figure 6.2.

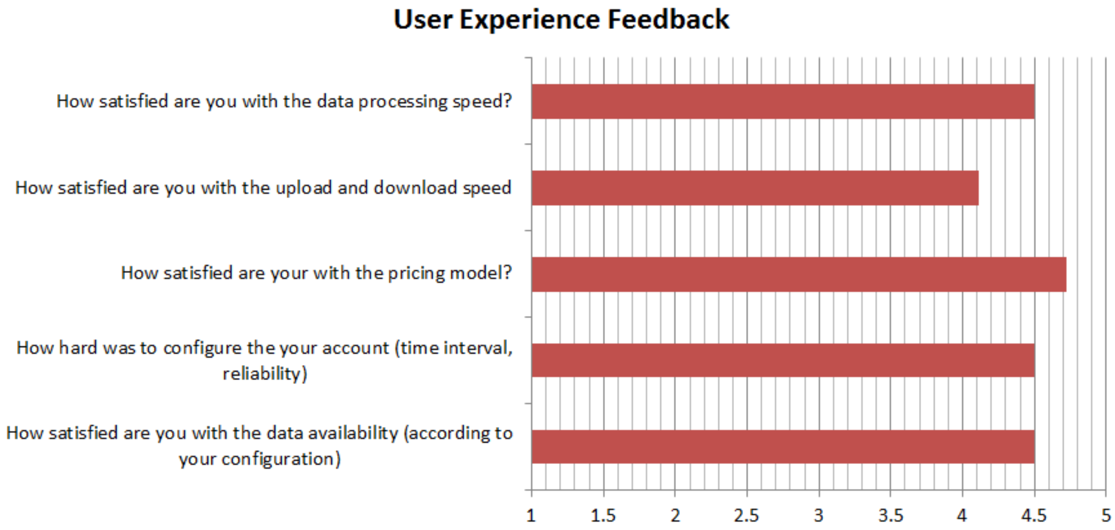


Figure 6.1: The graph shows the average user experience for the usability aspects of the prototype and the concept.

Based on data presented in figure 6.1, it is possible to conclude that the prototype and concept was satisfactory for most of the test users. The speed of the data processing, upload and download wasn't impractical for usage. The simplified absolute pricing model, what was used in the prototype was welcomed by the test users.

To understand the concept of the time intervals (which was completely new for the users) was without problem as well, although the test users were informed personally about the usage and the concept of the time frames.

Based on my experience, a video tutorial is necessary, explaining the usage of the application. (The attachments contain this video)

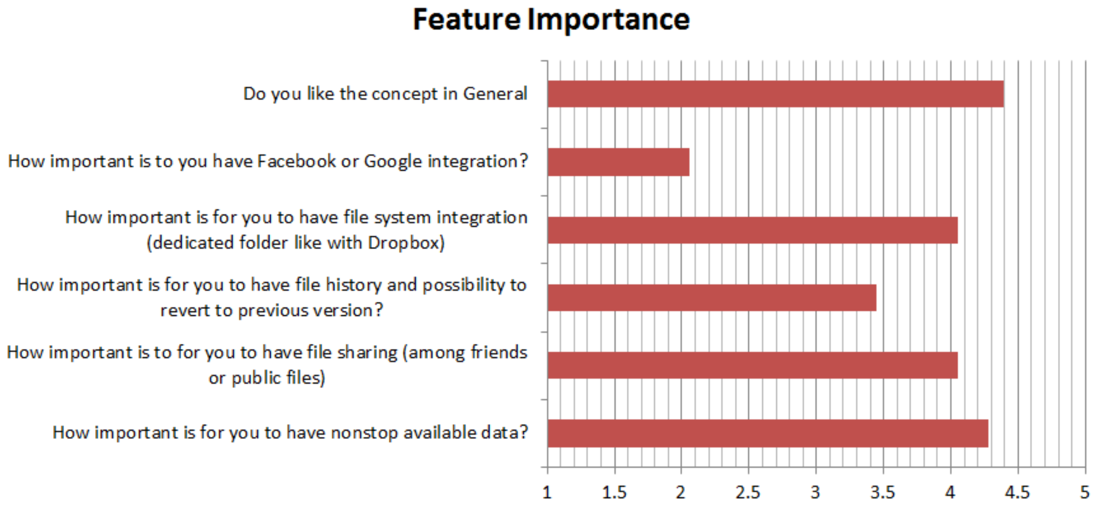


Figure 6.2: The graph shows the average rate of various features for an online storage system.

From figure 6.2 is possible to conclude, that the most important features, what are missing from the prototype are: file system integration (dedicated folder), file sharing and ability to have data being nonstop online. The last feature is already reachable by the time frame based cooperative storage system.

Two questions were asked about the preferences concerning the need for the data availability and how would the user like to increase his online storage space. In the last question, 3 models were choosable: Symform's 2GB contribution for 1GB reliable space, regular online data storage system freemium based model, and the time interval based model, which was presented in this thesis.

What is your preference in having data online?	**Count**
Some of my data I need nonstop (share, edit, etc.), some just I back up online and don't use it	13
Nonstop, 100% available data, but small storage	5

Table 6.1: User preferences for online data storage.

Table 6.1 shows, that most of the users need some of their data being nonstop online and some just want to use for back up. This could be satisfied by the time frame based cooperative data storage system, by simple creating a "server" configuration with 100% reliability and a backup configuration, which is for only long term data storage with smaller time interval.

What would you choose from the following, to increase the size of your online reliable storage?	**Count**
Flexible but more complex credit system, which depends on your time being online, reliability, how much you contribute. Doesn't require server.	8
Simple contribution system. I would run a server to be always available (24/7), in ratio of 2GB contributed space for 1GB reliable storage space online.	7
I rather not contribute at all, I prefer freemium data storage.	2
I rather pay for more storage, instead of any contribution	1
Other	1

Table 6.2: Increase of online storage space preferences.

Table 6.2, shows that both Symform's model and the model presented in this thesis are both acceptable for the users. This fact comes from the trend, that many of the users already have a server (or a simple personal computer), which is running all the time. Mainly these users are people, who constantly rely on computer usage.

6.2 Other Sources of User Feedback

The concept was published for the wide public in other sources as well. The basic idea was discussed with business mentors, angel investors and technology professionals from several companies. The following list shows the main communication channels, how the project was discussed in order to receive feedback from the potential users as well as from business mentors and angel investors.

Starcube startup accelerator - Starcube startup accelerator is a business course held in Brno. This course lasts for 3 month, where business and technology ideas are discussed with mentors, angel investors, as well as technology professionals [1]. I personally with two other, attended to this course in order to test the idea from the business aspect. Through the course many possible usages and opportunities were discussed for the concept and in general was greatly welcomed. At the event of Trial Starcube show, the idea was among the best 5 project participating in the course.

Project landing page - To represent the idea on the world wide web a landing page, describing the concept was created. It can be found at the host address: www.crowdybackups.com. This landing page hosts three versions of the concept, which are Family, Business and Global. The Global version is, what was described in this thesis. The other two are very similar technologies (ideas), but on a different user scale.

Social media - The idea was represented on the popular social site Facebook as well. This page opens the simplest possible communication channel with users, who were interested in using the prototype.

[1] http://www.starcube.cz/

It is important to mention that the process receiving user feedback about the concept was done more that 6 month. The design of the application and concept itself changed few times during this process.

The following list describes the main feedbacks from mentors and business investors, to whom the idea was presented.

Scale One of the first feedback for the concept presented in this thesis, was an advice to make it in smaller scales instead of creating a global service. This was for example creating an application for companies, which own many computers and working with lot of data. The other possibility is household scale, where the data would be backed up within the devices of a house hold (family, mobile phones, tablets, computers). The specification of this simpler system are differing from the presented one. Due to the small amount of users, the contribution selection process has to be simplified, it doesn't need complex time frame mathematical model. As well as the model of fair pricing is unnecessary, because each of the registered devices can be trusted, and these devices can agree on the amount free space shared for backup.

Feedback from security and backup specialists - Based on the feedback from the backup and security specialists, the main concern about this technology, is how to ensure the reliability. This feedback was given mainly for the smaller scale versions of the technology (for internal business and household usage), but it is applicable for the global version as well. The simple calculation based on the approximation of the Poisson probability distribution won't be sufficient enough, to ensure the desired reliability. Due to unpredictable changes of the user behaviors (concerning their configuration) or the unpredictable software un-installations the system has to provide higher redundancy.

Adding project as an extension to other applications - This idea was given by a business mentor at Starcube. The concept is to add the application (family or business) as a side project of some already known applications, like anti virus systems, CAD applications, operating systems, etc. This would easily create a good sales channel, while providing higher value for the base application.

6.3 Computer and Storage Space Usage Trends

It is necessary to measure the current trends of data backups and user storage space, in order to find out what are the possible limits of the time frame based cooperative data storage system.

A survey was published over the internet, with 106 responses. The survey contained questions about the users' available **unneeded** storage space on their devices (including NAS, computers and home servers), desired backup storage space and time frame, which indicates when are the users most probably online and with what percentage of reliability.

The survey was mainly (around 70%) completed with the current (or former) students and young graduates working at CERN nuclear research facility in Geneva[2]. the choice for this communication channel was made, because this community holds around 1500 members, coming from each continent. It was an excellent possibility to test the idea on a global scale.

The table 6.3 show the average values concerning user trends.

[2] http://directory.web.cern.ch/directory/

Category	**Average value**
Unneeded storage	59.42GB
Unneeded storage with NAS device	63.97GB
Required backup space	45.85GB
Reliability	86.09%
Reliability with NAS device	86.91%
Time frame frequency	6.5 days
Time frame duration	29 half hours - 14.5 hours

Table 6.3: Average user trends.

To calculate the average of the start time of the time frame, doesn't provide any important information, but the mode value can give some idea about the users start time trends: **Mode** was 18 half hour, which means Monday 9:00.

With the time frame, storage and required backup trends it is possible to reflect, in which time of the day, how much data is required to be available and how much free space is available as well. Also based on this information it is possible to visualize the practical limits of storage on different reliability percentages.

The figures A.1, A.2 (in attachments), 6.3 and 6.4 were calculated based on the survey, but with the same implementation as the prototype. This graphs give rough idea about the possible limits about the system and used mechanisms.

The figure A.1 and A.2 shows the total amount of free space over week distribution, and the relevant possible reliable storage space with the percentage of reliability. These graphs were calculated with the usage of Poisson probability approximation as well as the contribution time frame coverage functions (see 4.3).

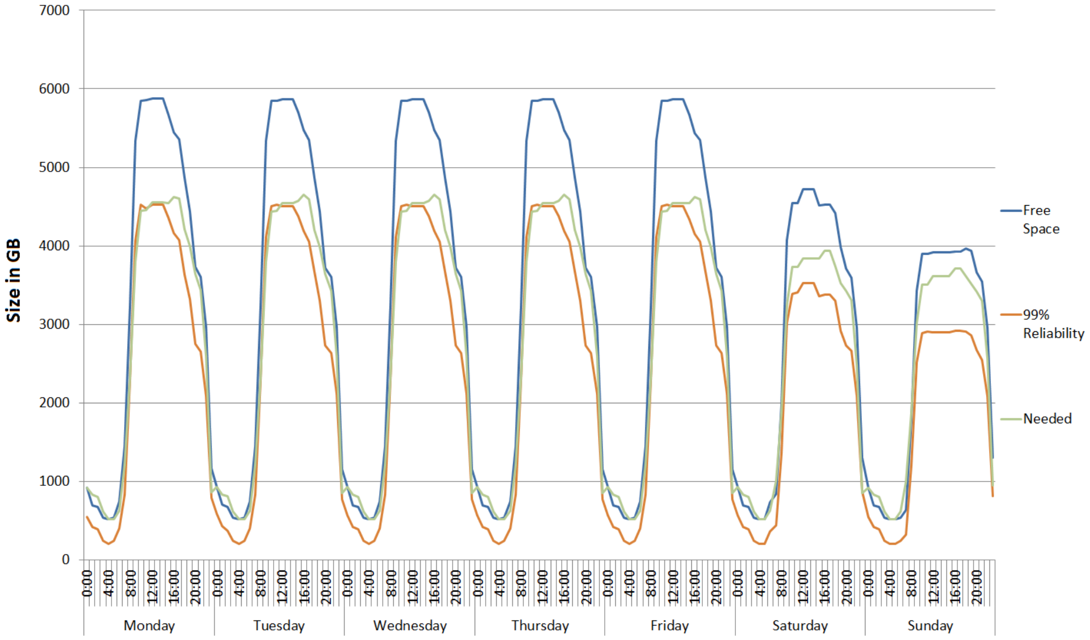


Figure 6.3: Needed backup storage and available backup storage over week distribution in GBs.

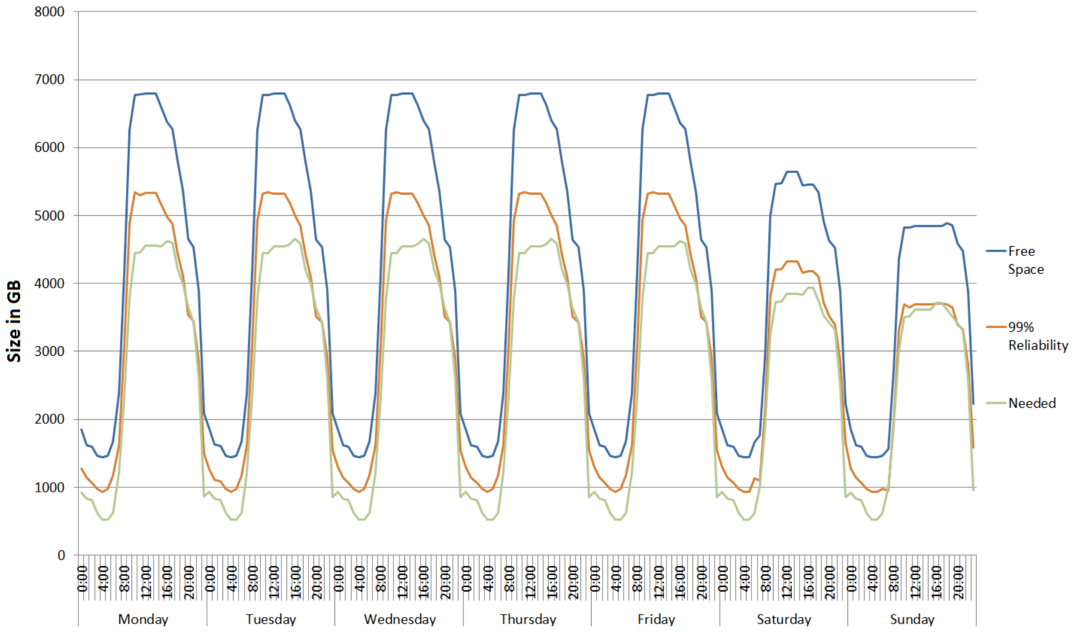


Figure 6.4: Needed backup storage and available backup storage over week distribution in GBs, including NAS devices.

The figure 6.3 and 6.4 were calculated with the same methods as mentioned before, but these graphs also show the desired backup storage over the week distribution.

The most significant result can be seen in the figure 6.4. The interpretation of this graph is the following:

If the users would contribute all their unneeded storage space with the promised reliability and promised time frame, all of the required storage space could be guaranteed using algorithms and techniques presented in this thesis, with the reliability of 99%

This assumption is valid under condition, if the users want to reach their data the same time, when they can contribute. Which condition is partly true, concluded from the results of the second survey, presented in 6.1

Figure 6.5 shows the comparison of available space offered for free to 106 users by the different online data backup solutions (Dropbox, Google Drive and Skydrive) and the available free space offered by the time frame based cooperative data storage solution.

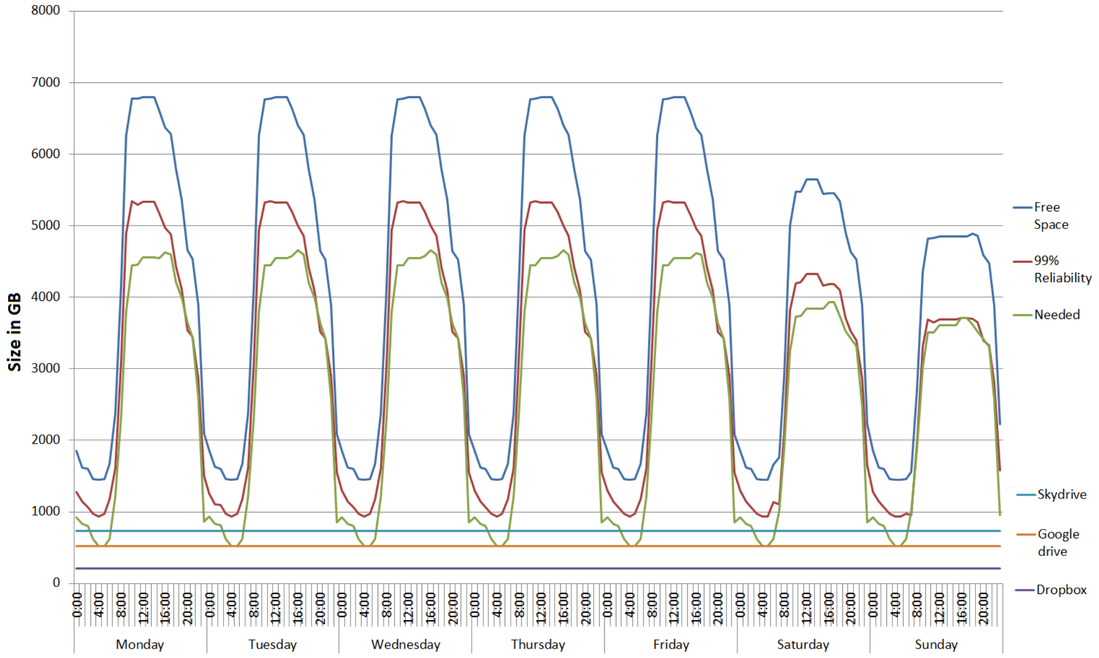


Figure 6.5: Comparison of the cooperative data storage system with the available regular storage systems in GBs.

This figure clearly shows, the theoretical limits of this system are much higher than at the regular client-server based systems. During business hours the cooperative data storage system offers 23.4 times more space as Dropbox, 6.7 times more space as Skydrive and 9.4 times more space as Google drive for free, only using clients computers, NAS devices and home servers. Exact values from these surveys can be found on the CD attached to this thesis.

Chapter 7

Conclusion

In this thesis a possible scheme and a prototype of a distributed peer-to-peer data storage system was presented. This storage system was designed to be usable for everybody with internet access. The solution focuses on giving freedom to the users about their contribution, they are not obliged to run a server for contributing to the system. Instead a design element, called time frame was introduced, which allows them to select what time of the week they are most probably available for contribution. The same time frame can be used to define the requirements for data storage. The specification was based on the assumption that the users don't need their data to be nonstop available. This allows the system successfully work with a set of unreliable users, who have limited but predictable accessibility to internet. **Introducing the time frames creates a completely new model for online data backups, which doesn't only consist the size of the data, but also what time and with what reliability can it be reached**

This thesis provides a design which covers most of the security and reliability issues, for a crowd sourced system for data backups. The main possible attacks against the system were discussed and a solution was proposed for them. Concerning the reliability erasure coding, especially Reed-Solomon codes were applied in the design of the system. These codes provide high flexibility and efficient usage of bandwidth and storage space. Using this solution, however requires an approximation of Poission binomial probability calculations, which was also successfully implemented with tolerable error rate. One of the most important aspects of the concept is the fair pricing model, which is considered as the catalyst for exchanging data within clients and contributors. Three fair pricing models were proposed and presented, with their benefits and disadvantages.

The implemented prototype based on the presented scheme and the concept itself was published through various channels.

The application prototype was tested by 23 users through 10 days. These users gave feedback (via one of the surveys) about the usability of the application and the concept. The survey also provided comparison with the already existing solutions (regular data center solutions and Symform§s solution). The results were satisfactory. The mayority of the test users considered the concept as a good solution for a flexible data backup system. The users found the speed and the usability of the application convenient, although based on their feedback many further features would be necessary to implement. These features are mainly the integration with the file system and file sharing in general.

The concept was discussed with angel investors, business mentors and technology specialist. This publication was done through the Starcube startup accelerator (2013 Brno), where the project was represented by a 4 member team (where I was considered as CEO and idea holder, the other team members provided great help in the business and marketing part of the project). The main feedback from this publication channel, is that the application should be also designed as a private small scale data backup solution for companies and households. The concept of cooperative data backup solution was within the 5 best ideas, in trial Starcube show. It was also presented at the final Starcube show by another team member as one of the emerging projects.

In this thesis user trends about computer usage (remote availability), data backups and free unused storage space was also studied. This was done by publishing a second survey, which was completed by 106 people, from all around the world. The results of this study are one of the most important of this thesis. Based on the simulation results, what was done using the data from the survey it is possible to state: **If the users would contribute all their unneeded storage space with the promised reliability and time frame, all of their required storage space could be guaranteed using algorithms and techniques presented in this thesis, with the reliability of 99%** (see figure 6.4). This statement is valid under condition, if the users accept that fact that their data will be available only the time, they can contribute (they are reliable online).

This system would offer significantly more free storage space for the user, compared to the ordinary data storage solutions as Dropbox, Google Drive Skydrive (see figure 6.5).

I personally would like to continue to develop this system. Based on the experience earned from the feedbacks and the testing of the prototype, I believe this system could be widely used as an alternative solution to the current online data storage solutions.

Bibliography

[1] Ann Chervenak, Vivekanand Vellanki, and Zachary Kurmas. Protecting file systems: A survey of backup techniques. In *Joint NASA and IEEE Mass Storage Conference*, 1998.

[2] Frank Dabek, Emma Brunskill, M. Frans Kaashoek, David Karger, Robert Morris, Ion Stoica, and Hari Balakrishnan. Building peer-to-peer systems with chord, a distributed lookup service. In *In Proc. of the 8th IEEE Workshop on Hot Topics in Operating Systems (HotOS-VIII*, pages 71–76, 2001.

[3] D. Eastlake, 3rd and P. Jones. Us secure hash algorithm 1 (sha1), 2001.

[4] M. Fernandez and S. Williams. Closed-form expression for the poisson-binomial probability density function. *Aerospace and Electronic Systems, IEEE Transactions on*, 46(2):803–817, 2010.

[5] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography (Chapman & Hall/Crc Cryptography and Network Security Series).* Chapman & Hall/CRC, 2007.

[6] Mark Lillibridge, Sameh Elnikety, Andrew Birrell, Mike Burrows, and Michael Isard. A cooperative internet backup scheme. In *In Proceedings of the 2003 USENIX Annual Technical Conference*, pages 29–41, 2003.

[7] Zhewen Lin and Tiantong You. Tt-stun protocol design for effective tcp nat traversal. In *Broadband Network and Multimedia Technology (IC-BNMT), 2010 3rd IEEE International Conference on*, pages 970–974, 2010.

[8] Thomas Mager, Ernst Biersack, and Pietro Michiardi. A measurement study of the Wuala on-line storage service. In *P2P 2012, International Conference on Peer-to-Peer Computing, 3-5 September, 2012, Tarragona, Spain*, Tarragona, ESPAGNE, 09 2012.

[9] Alfred J. Menezes, Scott A. Vanstone, and Paul C. Van Oorschot. *Handbook of Applied Cryptography.* CRC Press, Inc., Boca Raton, FL, USA, 1st edition, 1996.

[10] J. S. Plank. A tutorial on reed-solomon coding for fault-tolerance in raid-like systems. Technical Report CS-96-332, University of Tennessee, July 1996.

[11] R. Rivest. The md5 message-digest algorithm, 1992.

[12] V. I. Rotar' and V.V.I. Rotar. *Probability theory.* World Scientific Publishing Company, Incorporated, 1997.

[13] A. Shokrollahi. Raptor codes. *Information Theory, IEEE Transactions on*, 52(6):2551–2567, 2006.

[14] Bernard Sklar. *Digital Communications: Fundamentals and Applications (2nd Edition).* Prentice Hall, 2 edition, January 2001.

[15] Michael Szydlo and Yiqun Lisa Yin. Collision-resistant usage of md5 and sha-1 via message preprocessing. In *Proceedings of the 2006 The Cryptographers' Track at the RSA conference on Topics in Cryptology*, CT-RSA'06, pages 99–114, Berlin, Heidelberg, 2006. Springer-Verlag.

[16] O.P. Verma, R. Agarwal, D. Dafouti, and S. Tyagi. Peformance analysis of data encryption algorithms. In *Electronics Computer Technology (ICECT), 2011 3rd International Conference on*, volume 5, pages 399–403, 2011.

[17] Weijun Xiao, Jin Ren, and Qing Yang. A case for continuous data protection at block level in disk array storages, 2009.

[18] Zhang Yamei and Cai Pengfei. Research on using udp to traverse nat under p2p network environment. In *Advanced Computer Theory and Engineering (ICACTE), 2010 3rd International Conference on*, volume 3, pages V3–32–V3–35, 2010.

[19] Lu Yiqin, Fang Fang, and Liu Wei. Home networking and control based on upnp: An implementation. In *Computer Science and Engineering, 2009. WCSE '09. Second International Workshop on*, volume 2, pages 385–389, 2009.

[20] Li Zhao, R. Iyer, S. Makineni, and Laxmi Bhuyan. Anatomy and performance of ssl processing. In *Performance Analysis of Systems and Software, 2005. ISPASS 2005. IEEE International Symposium on*, pages 197–206, 2005.

Appendix A

Attachments

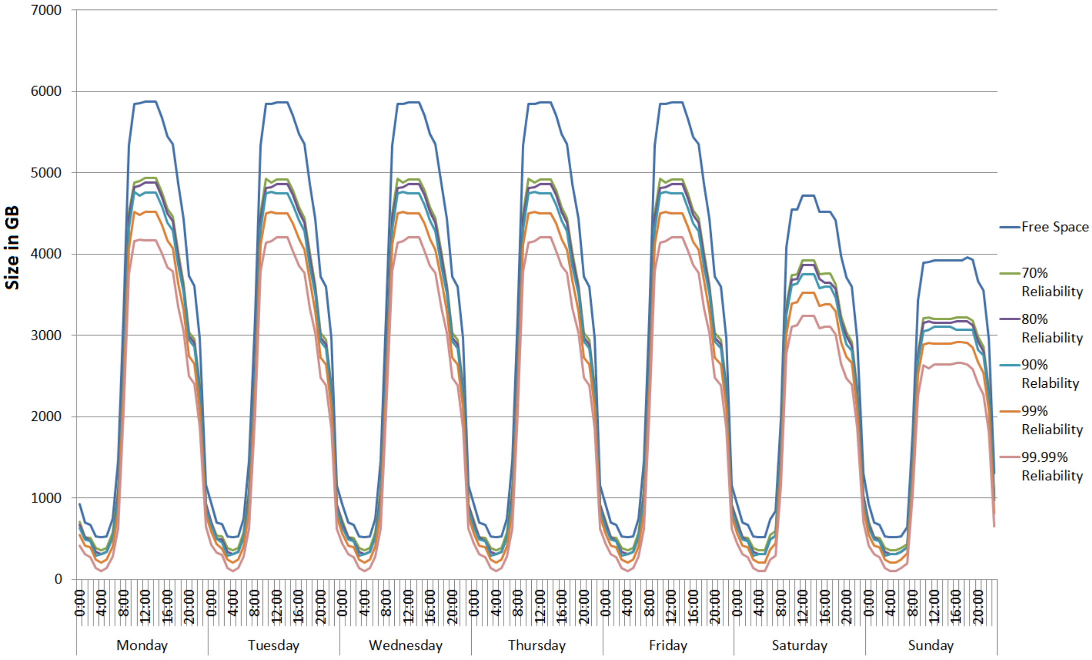


Figure A.1: Reliable storage space distribution over the week distribution in GBs.

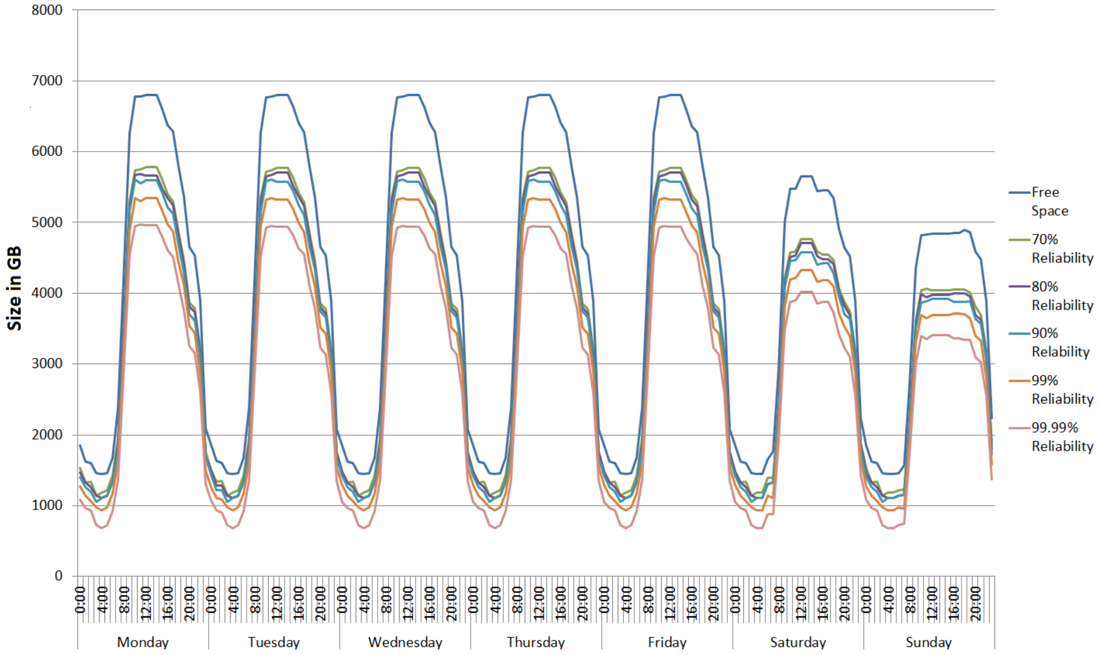


Figure A.2: Reliable storage space distribution over the week in GBs, including NAS devices.

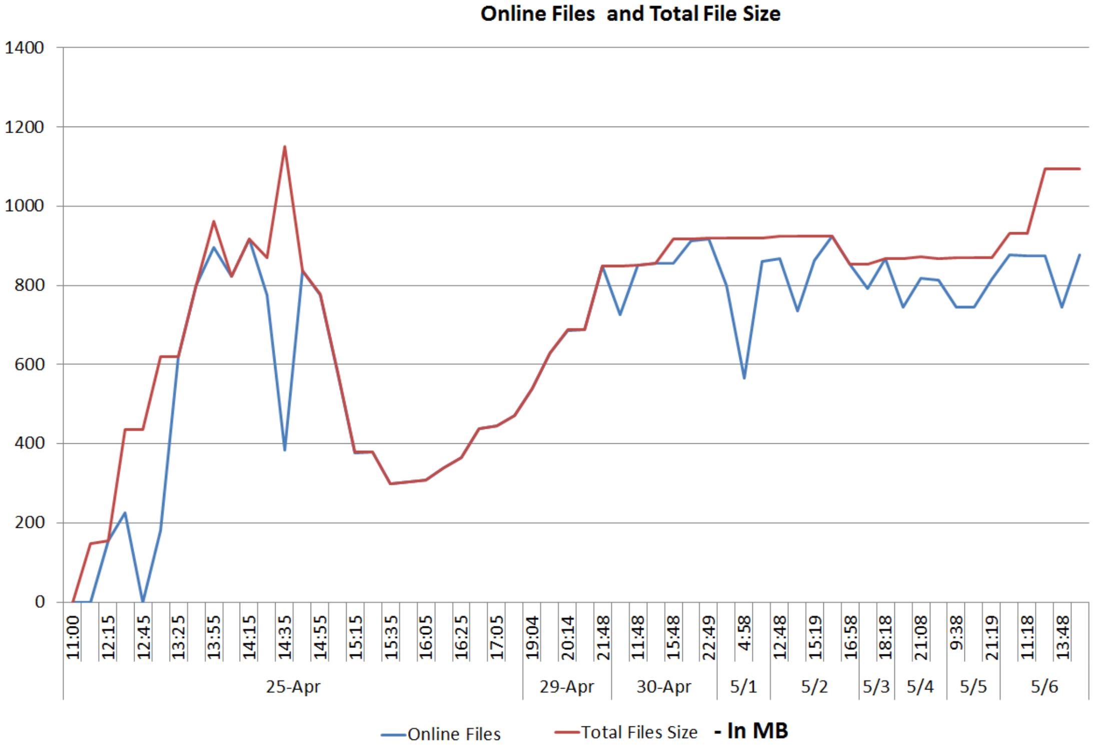


Figure A.3: Graph shows the total size of online reconstructable data over 10 days test session. The data availability changes are due the varying availability of the test users.

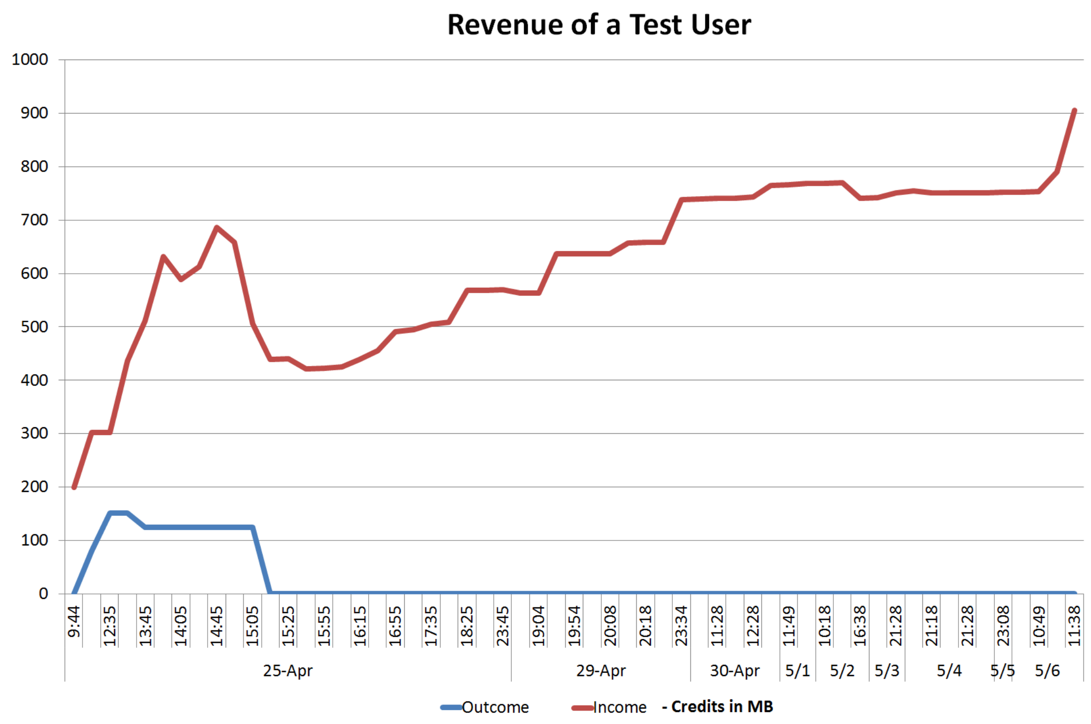


Figure A.4: Graphs shows the income and outcome credits (in MB) for a test user over 10 days.

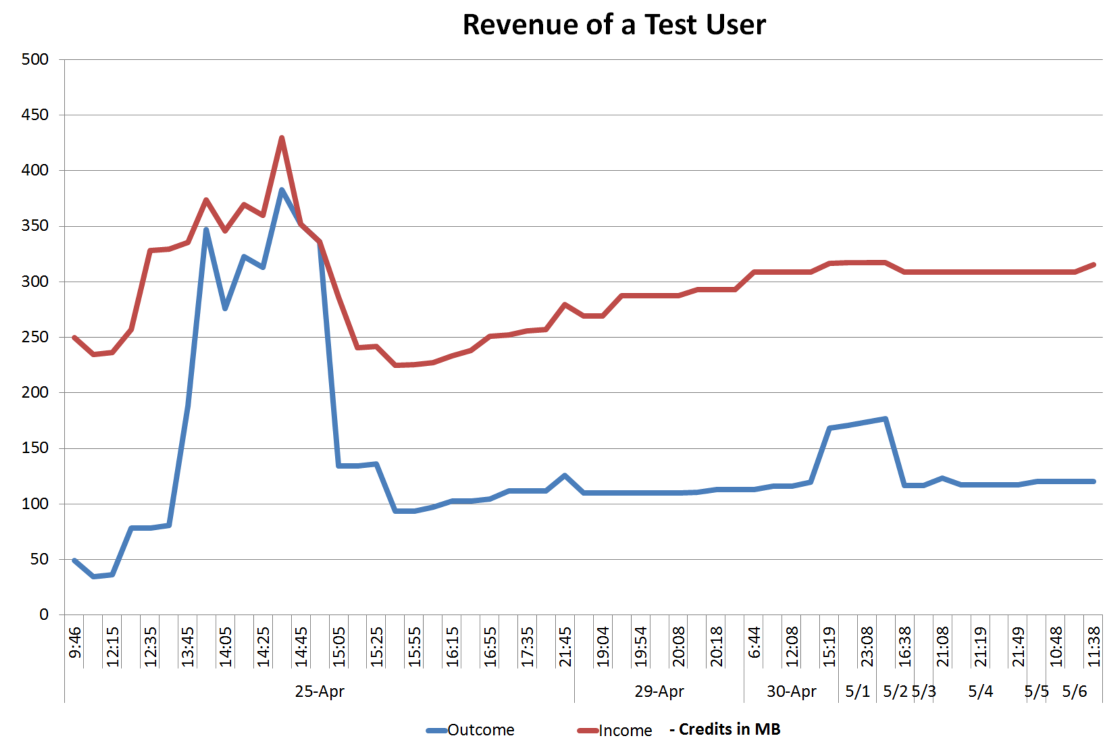


Figure A.5: Graphs shows the income and outcome credits (in MB) for a test user over 10 days.

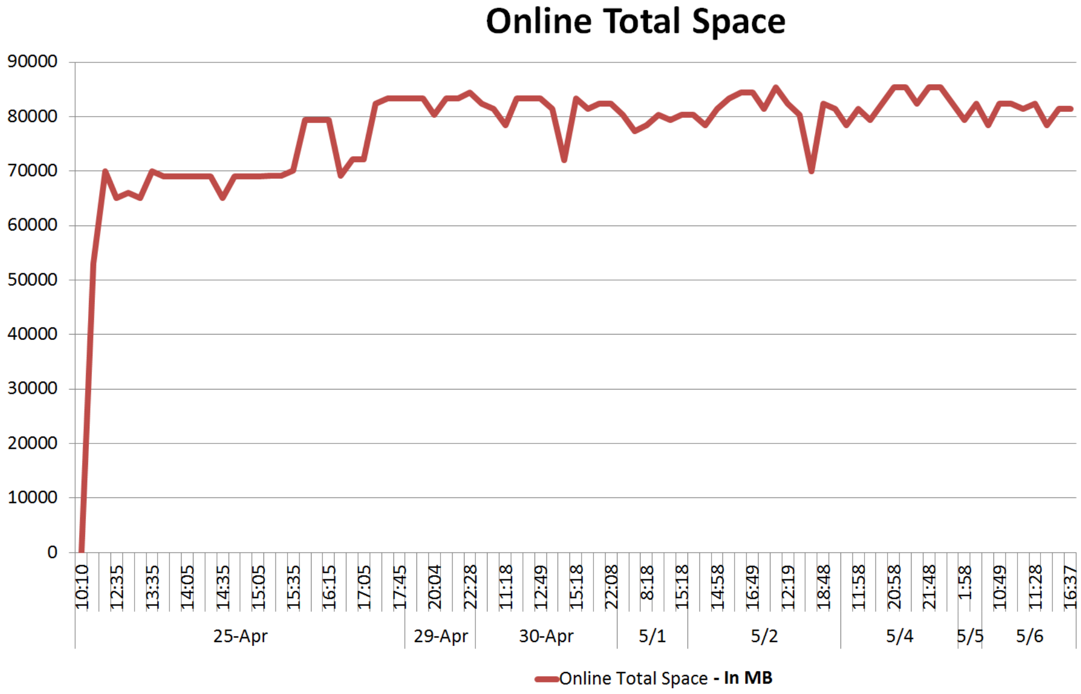


Figure A.6: Graph shows the total size of online data stored at contributions (not necessary reconstructable) over 10 days test session. The data availability changes are due the varying availability of the test users.

Appendix B

CD Content

The CD contains:

- Source code of the application - 4 modules
- Project video - idea presentation
- Manual video - how to use the application
- Poster
- Compilation manual
- Compiled code: JAR and WAR files
- Database scheme
- Test results
- Source code of Latex