UNIVERSITY OF OSLO
Department of Informatics

Retrivability of data in ad-hoc backup

Master thesis

Trond Aspelund
Oslo University College

May 23, 2005

Abstract

This master thesis looks at aspects with backup of data and restore in ad-hoc networks. Ad-hoc networks are networks made between arbitrary nodes without any form of infrastructure or central control. Backup in such environments would have to rely on other nodes to keep backups. The key problem is knowing whom to trust. Backup in ad-hoc network is meant to be a method to offer extra security to data that is created outside of a controlled environment. The most important aspects of backup are the ability to retrieve data after it is lost from the original device. In this project an ad-hoc network is simulated, to measure how much of the data can be retrieved as a function of the size of the network. The distance to the data and how many of the distributed copies are available is measured. The network is simulated using User-mode Linux and the centrality and connectivity of the simulated network is measured. Finding the device that keeps your data when a restoration is needed can be like looking for a needle in a haystack. A simple solution to this is to not only rely on the ad-hoc network but also make it possible for devices that keep backups to upload data to others or back to a host that is available to the source itself.

Acknowledgements

The author would like to thank Mark Burgess for the feedback and help on this master thesis and for the creation of the master course in Network and System Administration. Kyrre Begnum for the introduction to User-mode Linux and the creation of MLN. The author also like to thank the other teachers involved in the master degree.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

In recent years many different kind of mobile devices have appeared on the market which are used to store information. This can be everything from mobile phones, pdas to laptop computers. Even the small devices have storage space to store large amounts of important information: every thing from contact information to important work performed while moving from one place to another. If these devices are lost or damaged the cost of retrieving what was stored on them can be very high, or the work has to be performed again. To reduce this we look at the possibility to use other devices that can be reached through the network capabilities that exist in many of these devices to backup important data. Ad-hoc networks can cover large areas depending on the number of devices and distance between them. Can we use an ad-hoc network to reliable store backup data? The only limitation is that there must be a path that connects the nodes with data

Wireless devices need to have some form of infrastructure to communicate. One form is to use base stations that they communicate with, giving the devices some mobility. This mobility can be increased by make it possible by transferring a device from one base station to another. An example of a network with base stations are shown in figure 1.1. The nodes communicate with the base stations shown as antennas. Wireless communication is marked as dotted line while wired full lines. Communication can then be routed from the base stations to other parts of the network. Ad-hoc networks do not rely on any infrastructure other than what they create by themselves. Communication paths between devices are created through the network of arbitrary devices. An example of ad-hoc network can be seen in figure 1.2. There is no central control. Links between devices will come and go as time passes by. Devices can also have limited capacities when it comes to storage space, battery time and CPU power.

If the same devices are in contact with one another relatively often, they could store information belonging to one another.

The purpose of backup is to protect file systems from user errors, disk or other hardware failures, software errors that corrupt file systems and natural disasters. Most commonly used for restoration of files deleted by users or from disk failures. [CVK98] Backups essentially creates copies of files on other mediums, often also stored at other locations to reduce the risk of loosing data. With a backup system is not only the distribution of data into the network that is important but also to retrieve it when a restoration is needed. Not only must the data be available but also not have its integrity compromised. Restoration recreates local files to a previous state. Most of the data transferred in

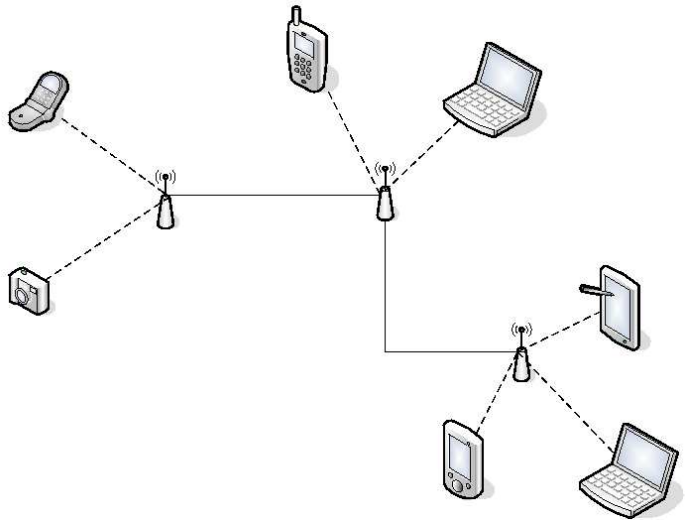

Figure 1.1: Non ad-hoc network

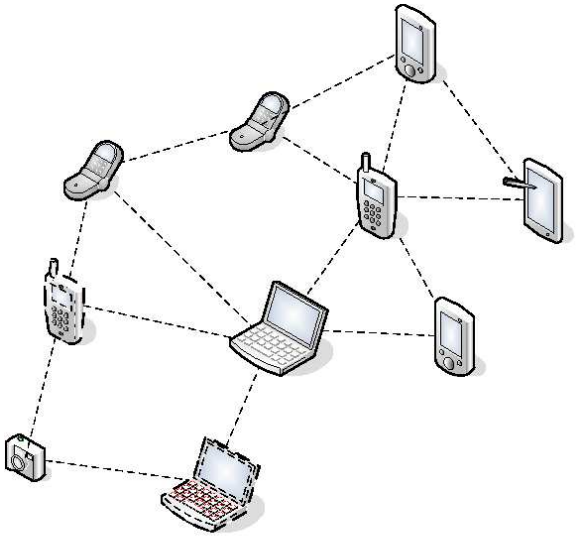

Figure 1.2: Ad-hoc network

backup systems happens when creating backups. Restoration doesn't happen as frequently. Creating a backup system in an ad-hoc network without any central control also adds several additional risks to the backup process. There is no way of knowing what others do to the data distributed out in the network so the sender must make sure it is keep confidential and that there is enough copies of the data around so that even if some parts of the network move out of range or loose it there should still be possible to retrieve it.

In an ad-hoc network changes in the infra structure and the availability of the nodes will have great impact on the performance of a backup system. One must answer some basic questions:

- Should files be transferred as one piece or divided in smaller pieces?
- If divided in smaller pieces should they all be given to the same host or divided among multiple hosts?
- How many copies would have to be made of the same piece/file to make it possible to restore the file?
- How to manage multiple versions of files?
- Can storage space be saved if several want to backup the same file?
- What methods can be used to maintain integrity and confidentiality of files?
- How can we limit the resources used on hosts storing backups?

Creating a large ad-hoc network for testing these questions would need large amounts equipment. For both ease of management and for the cost, it was chosen to use virtual machines to simulate an ad-hoc environment. User-mode Linux [uml05a], that is a virtual machine running in user space under Linux, ended up as the platform for the experiments. A program called My Linux Network (MLN) [BS05] was used to create and manage the machines.

1.1 Concepts and definitions

The following is a list of terms used in this thesis.

- Ad-hoc network - Arbitrary topology between nodes, randomly change of topology
- Backup - Make copies of data so it can be recoverd if original is lost
- Client - Host that uses the services on a server
- Convergent encryption - encryption using content as encryption key
- Eigenvalues - Can be used to rank the importans of nodes in a graph. See Chapter 3.4
- Encryption - Protect data to make it unreadable for unathorized persons
- Hash - A ”unique” value generated from a text that is smaller than the text itself

- Hop - Distance between nodes based on number of intermediate nodes
- Host - A computer attached to a network
- Meta data Properties or extra information about other data
- Node - Same as Host
- Peer-to-peer - Host acts as both server and client for each other
- Server - A host that provides services to other host
- User space Environment for normal processes without direct access to hardware or possibly damaging system calls

Chapter 2

Background

2.1 Ad-hoc network

2.1.1 Basics

Mobile devices within a small area can communicate directly with each other, but if communication is needed between devices that are further away they need some form of infrastructure. This can be achieved using base stations and a fix infrastructure among these. Most wireless internet access in homes and companies are based on using a base station and uses the same for of routing as in wired IP networks. Another approach is to create a network without any form of fixed infrastructure or central administration, an ad hoc network. ”A system is said to be ad hoc if its structure is periodically re-determined by random variables” [Bur04]. A device in a network is also referred to as a node. In a mobile ad-hoc network (or MANET), a group of mobile wireless nodes cooperates and spontaneously creates a network infrastructure [Fee99]. To be successful the network has to be flexible to provide access to devices move around and new users entering the area [CBC00].

2.1.2 Routing

The continuous movement of node in an ad hoc network makes the routing protocol important for the survival of the network. Each time a node moves the routes need to be updated or replaced to keep the network working. There are many different routing protocols that can be used in an ad hoc network. A taxonomy of different routing protocols are presented in [Fee99]. Structure, state information and scheduling are different ways of classifying routing protocols. Structure is if the protocols are uniform or non-uniform. In uniform protocols they participate equally with the routing. With non-uniform protocols not all nodes participate as much as others. State information looks at what information is exchanged between the nodes. This can be topology information like what is used in ”link state” [JMC+01] protocols or destination information like used in ”distance-vector”[PR99] protocols. Scheduling is based on when the source obtains route information. This can be either proactive, when a source tries to obtain routing information to all known destination, or discover routes when they are demanded. Protocols that generate too much traffic can increase collisions and contention. Creating optimal routes in an ad-hoc network can also be pointless as the network might have changed before they can be used so less optimal routes might be preferable.

Some routing protocols can even use location data retrieved from sources like the global positioning system to improve routing among the nodes [KV98].

2.1.3 Constraints

Ad-hoc networks have many constraints that affect the performance. Fading, multi-user and power limitations are major challenges [Ver00].

The majority of nodes in ad-hoc networks are battery powered. This energy has to be shared between transmitter power, processing power and other functionality like displays, speakers and so on [Ver00]. Newer technology utilizes this better and battery capacity has improved. The node can adjust the transmission power to the amount needed to contact an essential subset of neighbouring nodes [Per99]

Available bandwidth is also sparser than in wired networks, there are more interferences and the bandwidth must be shared among all nodes in a given area [Fee99]. Communication in wireless network can also be asymmetric while wired now uses symmetric communication.[Per99] To save bandwidth compression can be introduced at the link layer. Doing this can increase the efficiency of the compression. If compression then is made on several layers this can give a worse result than just once on a higher level. Several nodes can transmit to the same node without detecting the other nodes transmitting. This is the hidden-terminal problem. The reason for this happening is that the receiver is within range of the transmitting nodes, but they are to far apart to receive each others signal, wasting precious bandwidth. [Per99]

The movement of nodes will also lead to frequent and unpredictable connectivity changes [Fee99]. Fading will occur. Signals have the possibility to travel in multiple directions and can then cancel each other out making not only distance but also position important for quality of signals having impact on the error rate. Wired networks on the other hand have a low error rate.

2.2 Existing distributed backup systems

2.2.1 Peer-to-peer based backup programs

Pastiche is a peer-to-peer based backup system that uses three underlying technologies to perform its goals. Pastry [DR01] that is the basis for the peer-to-peer network. Content-based indexing [Man94, MCM01] that discovers common data in files. Convergent encryption [BDET00] that bases its encryption on the content of the data so several hosts can share common data without the need of sharing keys. Most replicas are placed close to the source to reduce network overhead and reduce restoration time. At least one replica is stored further away to protect against larger disasters that might take out close replicas together with the original source. [CMN02] pStore [BBST01] is another backup system. This uses Chord [SMLN$^+$03] for its peer-to-peer functionality. It also uses convergent encryption on the blocks it divides its data into.

2.2.2 Finding friends

Killijian, Powell, Bantre, Couderc and Roudier et al [KPB$^+$04] finds mechanisms that is based on reputation and rewards particular interesting for selecting hosts to use for backup repositories. It

can not rely on any trusted third parties or connectivity of a majority of the considered population of devices.

For Pastiche to be able to find or route messages among its peers, each node has three sets of states that it maintains; a leaf set, a neighbour set, and a routing table. The leaf set contains of a number of the closest nodes, half with a higher nodeId and the rest with a lower. NodeId is the id used by nodes in the network and all of them are uniformly distributed in the nodeId space. The neighbour set contains a list for the nodes with the closest proximity metric. This is critical for buddy discovery for nodes with uncommon installations. The routing table is based on prefix routing and used to send to nodes not in the leaf set. Prefix routing is based on the match of the longest prefix of the nodeId. [CMN02] The nodeId is a hash of the nodes fully-qualified domain name. When a node joins a network it generates a random nodeId and sends discovery request toward that node, nodes along the route will calculate their coverage from the abstract in the request. If the coverage is low it will use the coverage rate rather than the hops as a distance metric. [CMN02]

2.2.3 Prepare data for backup

Data on a PDA can be amongst other things like a contact database. This kind of data is regularly synchronized with a desktop computer application [KPB+04]. Data to be backed up would then only need to be the changes since last synchronization. Other kinds of devices like image-capture devices would generate unique data. Audio, pictures and not to forget video generates a huge amount of data. All this data is vulnerable until a backup can be made [KPB+04].

Chunks and blocks are essentially the same thing. Pastiche uses the term chunk while pStore uses block. Chunks or blocks are files divided into smaller pieces to make large files more manageable to work with and. The size of these chunks or blocks all depends on how the different systems are designed.

Pastiche looks like a normal file system from the kernel using the XFS device. Files are locally stored as chunks in a chunkstore and decrypted if needed. When Pastiche detects a close on a dirty file it will prepare it for addition to the chunkstore. The part of Pastiche that handles the files between the kernel and the chunkstore never deletes chunks but keep a log of what chunks a file consist of and a entry if the file is deleted. The backup backend performs the communication with other nodes for storage and restoration. It will also remove unwanted chunks from the chunkstore.

pStore [BBST01] splits files into blocks and creates file block lists that identifies how the file is to be reassembled. These are then spread on the network. Every time a new backup are made of the file it will be compared to earlier blocks and only new blocks and file block list will be distributed, saving space as only the changes are stored. The file block lists will know what blocks from earlier backups are needed to restore the file. File block lists are the only part that must be unique between data from several users. If the same file is put on the networks from several users they can use the same file blocks to restore their data. The encryption method used on the files allows for this. Directories can also be maintained in pStore and keep track of their changes. Text files are used to list files and subdirectories of each directory [BBST01].

2.2.4 Backup data

The aim of the backup process is to protect against loss of data. The utilization context will have impact on whether partial or complete backup should be used. Partial backups will minimize the amount of data to be transferred and stored [KPB+04]. Fragmentation-replication-dissemination techniques can be used to divide data and spread it over several data savers [KPB+04][DBF91].

Each node is responsible for its own backup plan. Data can be sent directly to the destination data savers like with pStore [BBST01] or it can be sent to one data saver which then will propagate data to the right number of other data savers. Data saver is the nodes that stores the backups, all nodes are data savers for the other nodes in the network. Several systems do also use caching along the retrieval path to increase data replication. In a backup system such a cache is not very well suited as there are most file insertions and few file retrievals. Some caching on the data owner might be more beneficial to ease creation of backups of new version of files [BBST01]. In Pastiche when a snapshot of the file system is made it will get a set of new chunks, chunks to delete and meta-data for files that have changed. These lists are signed before transmitted to its neighbours. The public key is also sent. The public key is used for verification. In deletions for example will the signed values in the delete set have to match width the public key received with the add set. This is to prevent deletion of chunks still needed by others or from malicious users. The receiver will get the chunks that does not exist in its repository and update their counters for all chunks. [CMN02]

To prevent loss of data if multiple data savers failures, data must be stored redundant on several data savers [KPB+04].

2.2.5 Restoration

Restoration is easy as long as it has the skeleton for the file system, it will then know the chunkId of the chunks to retrieve. Otherwise it has to retrieve the root of its meta-data from the network before it can begin. The decryption of this data is based on the hosts passphrase. [CMN02]

Recovering data is an important part of a data backup service. The main focus is here to find and retrieve data back to the original device or use another device as surrogate for the original device if needed. To retrieve the data a path to a data saver must be established either thru other nodes or wait until direct contact can be established [KPB+04].

Two different recovery modes are push recovery and pull recovery. In push recovery will the data saver automatically sends data backups to the data owner. Using pull recovery the data owner is searching for the data backups it requires [KPB+04]. This approach would be like methods used in peer-to-peer file sharing networks like Gnutella [gnu05].

2.2.6 Security of data

Killijian, Powell, Bantre, Couderc and Roudier et al [KPB+04] identify three new threats to for a service based on cooperation between mobile nodes with no prior trust relations ship; selfish devices that do not cooperate; failure of backup repositories or they trying to attack the confidentiality or integrity of the backup data; flooding of devices by fake backup requests. Mobile devices are also prone to energy depletion, physical damage, loss or theft [KPB+04]. Data can be protected with

cryptography but this is an trade-off between the level of protection and the energy and resource cost [KPB+04].

Authentication is provided in [ELBZ02] by using the Diffie-Hellman Protocol to establish a shared secret key, pStore [BBST01] on the other hand uses public private keys and does not have any shared secrets.

Each chunk in Pastiche uses a SHA-1 hash of its content to create an encryption key to protect the data. When two equal chunks are created, independent on where they were, will same encryption key be used. This is a form of convergent encryption [BDET00]. Another SHA-1 of the other hash value is used to identify the chunk in the network. pStore [BBST01] uses a similar approach. Each chunk has an owner list and a counter of how many references each owner has to the chunk. These values are used detect when they can be removed to save space. When a file is overwritten will the counter for chunks that are no longer used be decremented. Meta-data contains the handles for each chunk in the file and can be used to create decryption keys for chunks. Pastiche does create chunks out of meta-data and encrypts the content. Names for meta-data chunks are created just once so updated files would still use the same name as long as it exists. [CMN02].

The versioning scheme of pStore will also make it possible to retrieve all versions not using a corrupt file block unlike what would be with versioning systems like CVS [BBST01]. Each snapshot made in Pastiche would here have the same properties as the versioning scheme of pStore.

Elnikety, Lillibridge, Burrows and Zwaenepoel et al [ELBZ02] adds the use of erasure codes to their backup solution. By using codes like the Reed Solomon erasure codes and Tornado codes data can still be regenerated even if some blocks are lost. Their choice was the Reed Solomon erasure codes as they require less network traffic and minimal storage. It does on the other hand need more processing but less than the time needed for encrypting the data.

In pStore the data owner can delete data from the network. The public key that is part of the metadata is used for this it can also be used to make sure quota limits are not exceeded. When several data owners share a file block they will just be removed from the list when asking for deletion of the file. An impostor can also in the worst case only add or remove its self from this list and not harm the data [BBST01]. Removal of infrequently or old files are common in peer-to-peer systems. With backup this is no good. Backups are rarely accessed until needed and could lead to unavailable backups when they are needed. With expiration dates files could be deleted, and in the use of pStore old parts of files that have not changed can be removed making it impossible to do a full restore of the file even if the file itself is frequently modified [BBST01].

2.2.7 Design and security issues

Killijian, Powell, Bantre, Couderc and Roudier et al [KPB+04] list several problems with ad-hoc backup that have to be addressed. This list contains resource allocation, garbage collection of obsolete backups, integrity and confidentially backup data. Pastiche and pStore incorporates the possibility to delete data from other nodes that no longer are needed. Using signing to prevent malicious nodes from deleting others data. [BBST01, CMN02] Batten, Barr, Saraf and Trepetin et al [BBST01] discovered that in pStore digitally signing of chunks adds significant overhead to performance, bandwidth and storage. File blocks could even end up having more metadata than real data. One reason for this is the inclusion of the users public key. The hash values used as identifiers is also an form of integrity check of the data [BBST01]. Encryption is used to keep data

confidential. The convergent encryption used creates a hole in the confidentiality. If two hosts share the same data is it possible for these two hosts to know that the other host also has the same data. This kind of information can post a security risk. [CMN02] There might be data owners trying to get free backups without giving anything back to the community. To prevent this there can be incorporated a commitment period for new partners. In this period will restorations be denied but all other operations allowed. It must also be longer than the grace period. The grace period is a tolerance for the partner being down without discarding it [ELBZ02]. Pastiche queries buddies before storing a snapshot to see how their cover rate is. If this cover rate have become to low it will look for other more reliable buddies to store their data. This will help to eliminate malicious nodes in the network. [CMN02]

Killijian, Powell, Bantre, Couderc and Roudier et al [KPB+04] lists 7 threats that the data backup service must face. The first is permanent and transient accidental faults affecting a data owner. Then there is the theft or loss of data owner device. Faults that make the data saver unavailable when recovery is required are another one. We then also have accidental or malicious modification of data backups. Read access to backup data by unintended users is also a threat. Denial of service through selfishness is another threat as it would not work if it is no incentive for devices to participate. The last threat is denial of service through maliciousness where a malicious data owner tries to saturate data savers with false requests and preventing the service from others. The context of the services can remove some of these threats.

2.2.8 Experiments

Batten, Barr, Saraf and Trepetin et al [BBST01] created a network consisting of 30 nodes running pStore to simulate the chances of retrieving a file depending on the number of replicas. These 30 host where in reality run as several clients on 5 hosts. They used two sets of data, a HomeDir profile with 13MB data in 102 files and one SoftDev profile of random files from the machine with a date size of 696MB in 26959 files. Each profile was tested with 1,2 and 4 replicas. After this was spread on the network they measured how much was available for the number of nodes that where up. With 23 of the nodes down they could still retrieve 95% of the data with 4 replicas.

They also performed experiments to test the bandwidth usage and storage space usage for pStore. By comparing 4 different methods for storing the files; local tape backup, pStore without version support, with version support and with using CVS as version control. Versioning with pStore used the least amount of bandwidth, CVS using most. pStore without versioning is a little higher because of the added overhead. CVS and pStore used about the same amount of space. Without versioning using some more space as long as block sharing is possible. The local tape backup uses much more space then the other alternatives the uses block sharing [BBST01].

Pastiche uses a local chunkstore to store all files and directories created. This adds extra overhead in creation and modifications of files. Cox, Murray and Noble et al [CMN02] tested out its performance by running three tests; wide create wide mkdir and deep mkdir. The wide tests created 1000 files or directories in the same folder while the deep mkdir created 1000 directories inside each other. The wide create ran 186% slower than ext2fs and wide mkdir 174%. The reason create files is slower is that the XFS devices makes an extra call to the file system handling applications when creating files. Deep mkdir was 38% slower than ext2fs. The difference between the wide and deep mkdir lies in the fact that the container file must be rewritten for each added entry. For the deep test

there is only one entry.

2.3 Reducing Resource Usage

Most devices working in an ad-hoc network have limited resources. On mobile devices many of these boil down to limited power supply. Transmitting and receiving signals, calculation and displays all use the limited battery capacity. Limited physical size and cost also have limitations.

The energy that is used for the communication determines the area each device can find other devices. The ad-hoc network then expands this out to a larger area. To maintain this network in working condition intermediate devices will have to retransmit signals using of their own limited resources.

Limiting the amount data to be backed up will help saving space on other devices and transmitted cost. This can be achieved by compressing blocks before they are transmitted. However one must be careful. Compressing a block can make it larger than the original depending on the entropy [OW93] of the block and how the compression algorithm works. Adding compression on the other hand will increase the processing of each block. This will reduce the life-time, time until recharge is needed, of the device. Some types of data are already compressed and performing another compression will most likely not reduce size significantly. Images on a digital camera are examples of files that already have some form of compression. Digital photos are also files that can take up large amounts of storage space.

Each time a new block is created it has to be encrypted and an identifier has to be made. Encryption will prevent unwanted people from reading the data. The identifier is made to ease finding the block later. This means that the block has to be processed three times for these operations. First hashing of the content to create the key for the convergent encryption then encrypting the file and last creating the identifier for the encrypted file.

Compression and encryption is all performed on the device that wants to backup its data. That devices are also the one with most to gain of a backup been made. If the cost to perform these actions is too high, then loss of data is an acceptable risk. Not performing encryption and compression can of course be an acceptable alternative.

Dividing files into smaller blocks can many advantages. It will reduce the size of each piece transmitted into the network as the error rate can be high in ad-hoc networks. This will reduce the amount of data to retransmit and save resources. Retransmissions performed on higher levels can conflict with those on lower levels in wireless communication. When backups are made of several versions of a file, will each equal block increase the number of copies and the chance to be able to retrieve the versions having that block. The chance will always depend on the block that has the least available copies. Smaller block will also make it possible that several equal blocks are stored on the same backup node and it can then save storage space. Dividing files into to many blocks however will increase the size of the directories that manage what blocks belong to which files. The added meta data in each block will also increase the overhead in the network.

If each block is stored with names based on their identifier a search for a block would not have to do as many computations on each node in the backup network. The ewer resources that are used for the basic operations of the network, the better it is for the devices in the network.

Chapter 3

Methodology

3.1 Important aspects

The task of experiment is to measure some of the aspects introduced into ad-hoc backup that does not exist in normal forms of backup. The changing structure of such network infrastructure will have impact on retrieval of the data. The focus of the experiment will therefore be to:

- Measure retrieval of data from an ad-hoc based backup.
- Look at how much of the data is accessible as the number of nodes changes.
- See how the distance (number of hops) to the data changes.

3.2 Creating an ad-hoc network

Building up an ad-hoc network consisting of physical devices to perform measurements would be a big challenge. It would require a large amount of equipment that would cost more than founds available. In addition it would also require a large area to operate in. Such an area should not interfere or be interfered by other devices working in the same frequency areas. Such interference could affect the measurements. To create a suitable network, nodes would also have to be placed at a sufficient distance between each other to prevent nodes to communicate with larger parts of the network than wanted. The approach chosen instead is to use virtual machines and think of these as the nodes in the ad-hoc network. In this way the communication can be free of external disturbance. This approach is however not without limitations. Any physical properties of the communication media would have to be simulated or excluded from the test. Ad-hoc networks will have routing protocols dealing with keeping paths to other nodes. The application itself would not have to deal with this. Because of this will the backup application work as other peer-to-peer based applications in more stable networks. A simulation can therefore be achieved in a fixed network structure. Some aspects would however be removed from such a simulation. Delays introduced by movement of nodes and significant changes in available bandwidth would not any longer be present. These could however be introduced into the experiment by adding delays and bottle neck in the network

but would just increase the uncertainties in the measurements. This is also not a experiment that focuses on performance of the network itself.

3.3 User-mode Linux and MLN

3.3.1 User-mode Linux

User-mode-linux [uml05a] is a patched Linux kernel that can be run in user space on a Linux system. User-mode linux is Linux ported to Linux [Dik01a]. There is also projects working on porting it to other platforms like PPC [uml05b], Windows and Freebsd. [Dik01b] Using user-mode Linux you can create large networks all running on the same host. Most Linux software can be run on a user-mode Linux virtual machine. The exception to this is software that works closely with the hardware. Each User-mode Linux machine has one or more file system that will be mounted inside the virtual machine. From the host the user-mode Linux looks like normal user space processes but from the inside the virtual machine it looks like a host kernel [Dik02a]. Running many machines can then demand a lot of disk space. To limit this is there also support for something called COW file system. COW stands for copy on write and it stores only local modifications for each machine to separate files, but uses on file to store the file system for the unmodified files. The file system that the COW system uses as the base for the host file systems can not be modified after its initial use [Dik01b]. Another file system is the hostfs that makes it possible to mount folders on the host into the virtual machines. The virtual machines can communicate to the outside world or other virtual machines. A complex network infrastructure can be made virtually but can on the same time be given access to the outside world. User-mode Linux uses switches daemons to communicate between the virtual machines or TUN/TAP device to communicate with the hosts network. The TUN/TAP device can also be used between the virtual hosts and bridge them together. [uml05a]

Some of the benefits with user-mode Linux are that the machines can be stopped and started without having to reboot a physical machine. This can then also be done remotely as long as the user can log into the host machine. Resources can be shared among several users like web users can be given their own server instead of virtual hosts on one web server [Dik02b]. If the user-mode Linux is compromised and they can break out of the user-mode Linux they will only become a regular user on the host without root access [Dik02c].

3.3.2 My Linux network (MLN)

My Linux network (MLN) [BS05] is program made to ease the creation of networks with User-mode Linux. With a configuration file for your network it will create all nodes based on file system templates and perform all configuration needed to have a running network. Common settings for nodes can be inherited from super classes so little configuration is necessary for each individual node, making it easy to configure large networks of similar nodes. Thru the functions of the MLN program can the network be built, modified, started or stopped. You can start and stop either individual machines or the whole network. With help of the configuration file it is easy to make several equal networks or just make small changes between them. MLN can also change and modify a already created network [BS05]

3.4 Node importance and Eigenvalues

A Network can be represented as a graph. An imaginary network consisting of 4 nodes is drawn in figure 3.1. The graph shows the communication paths between the nodes. Some nodes will have a more important position in the graph than others. A node is more central if it is connected to many nodes that are also connected to many nodes. Removal of an important node will affect the graph more than removal of a less important one. Important nodes have many important neighbours.

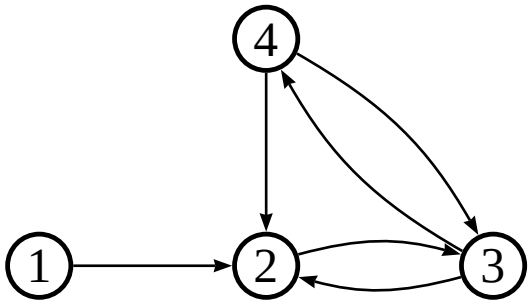

Figure 3.1: Exsample graph

A graph can be represented in a matrix called adjacency matrix. Figure 3.1 is a small graph. Table 3.1 and 3.2 shows two different adjacency matrices for figure 3.1. The difference between these two adjacency matrices is that one is for a directed graph (table 3.1) and the other for a undirected graph (table 3.2). In the directed graph all arrows have an arrow that shows the direction, like drawn on figure 3.1. In the undirected graph the communication can go both ways on the links, ignore the arrows and the double up links on the graph. If there is a link between node one and two it is represented with a 1 in row one column two. If the graph is undirected there would also be a one in row two column one. An undirected graph will always be symmetrical about the diagonal. The values in the diagonal are also zero because the node is not connected to itself.

$$I_i = k \sum jA_{ij}I_j \tag{3.1}$$

$$A\vec{I} = \lambda\vec{I} \tag{3.2}$$

To calculate the importance of node we have to sum up the importance of all nodes linked to that node. This can be done by using equation 3.1. I is here the importance and A is the adjacency

	1	2	3	4
1	0	1	0	0
2	0	0	1	0
3	0	1	0	1
4	0	1	1	0

Table 3.1: Directed adjacency matrix

	1	2	3	4
1	0	1	0	0
2	1	0	1	1
3	0	1	0	1
4	0	1	1	0

Table 3.2: Undirected adjacency matrix

matrix. Another method for calculating this importance is to use the eigenvalue. The equation for the secular eigenvalue (λ) is given in equation 3.2. λ is the same as $1/k$. $\vec{I}$ is the eigenvector for A. One matrix can have several eigenvalues and the eigenvector for the largest eigenvalue represents the importance or the centrality of the nodes. The largest eigenvalue is always a positive number. The combination of eigenvalues and eigenvectors are sometimes called eigensystem. [Bon87][Bur04]

In the example network in figure 3.1 we can see that nodes 2 to 4 are more important than node 1. In an undirected graph node 2 looks like the most important node. For a directed graph this is somewhat harder to determine. Using the eigensystem we can now mathematically determine what the true rank of these four nodes. The eigensystem values for the directed graph is shown in table 3.3 and in table 3.4 for the undirected. For the directed graph we can now see that node 3 and 4 are equally important and node 2 is then next. In the undirected graph node 2 is the most important as predicted, then comes node 3 and 4. Node 1 is the least important in both graphs. If node two is removed we can see that nodes 3 and 4 will lose contact with node 1. In such a small graph it is possible to reason which nodes are most important but when the graph becomes large this will be a difficult task. Using eigenvalues this can be calculated independent of the complexity of the graph.

This method is used to determine the page rank in Google [HK03], detect bottle necks in systems, virus attacks as well as ranking of importance of nodes. In this project we will use this method

	1.6	-1.0	-0.6	0.0
1	0.4	1.0	2.6	1.0
2	0.6	-1.0	-1.6	0.0
3	1.0	1.0	1.0	0.0
4	1.0	0.0	1.0	0.0

Table 3.3: Directed adjacency matrix Eigensystem

	2.2	-1,4	-1.0	0.3
1	0.5	1.7	0.0	-2.2
2	1.2	-2.5	0.0	-0.6
3	1.0	1.0	-1.0	1.0
4	1.0	1.0	1.0	1.0

Table 3.4: Undirected adjacency Eigensystem

to rank the nodes at the different layouts of the network.

3.5 Connectivity

Connectivity is an important value used to tell how easy information is spread throughout a system. A network with higher connectivity, χ, has a larger chance that a message can be passed directly between any two nodes. Connectivity can be a number between 0 and 1, where 1 is that there is a connection between every node. It can be calculated using equation 3.3. This value depend on the number of nodes (N), adjacency matrix (A) and a vector, $\vec{h}$, that is 1 if a node is available and 0 otherwise. [Bur04]

$$\chi = \frac{1}{N(N-1)} \vec{h}^T A \vec{h} \tag{3.3}$$

3.6 Network setup

The network for the experiment consists of 30 nodes that are part of the backup network. This is the same number of nodes as used in [BBST01]. In addition there are two other nodes. One is a gateway to the outside world and the host machine, the other one as a central control centre for running the experiments. All nodes use the same application that is shared over NFS. [ea00][PJS+94] NFS was chosen instead of using the hostfs possibility of User-mode Linux because earlier experiences with hostfs have sometimes had problems with detecting changes in the files on the host. All data that is used during the experiments is on the different virtual machines. Results are stored directly on the NFS share for easier processing of the results. To simplify the experiment is all nodes able to communicate directly with each other. This would eliminate the need for using any routing protocols. The network structure between the nodes is created by telling each node to which other mode they should establish communication. Nodes are started from higher to lower node number. Meaning that node 30 is started before node 29. Links between nodes are established from nodes with a lower number to a higher one. All of these links are bidirectional so all communication will float both directions after they are established. Connections is established when the network is started. There is a little delay between each node starting so they will be up and running before any connections are attempted to be established to them. The graph is drawn in figure 3.2. From this picture can we see that the graph is very complex. Even if the names of the nodes were given, getting an understanding of the network would be difficult. Table 3.5 shows the adjacency matrix

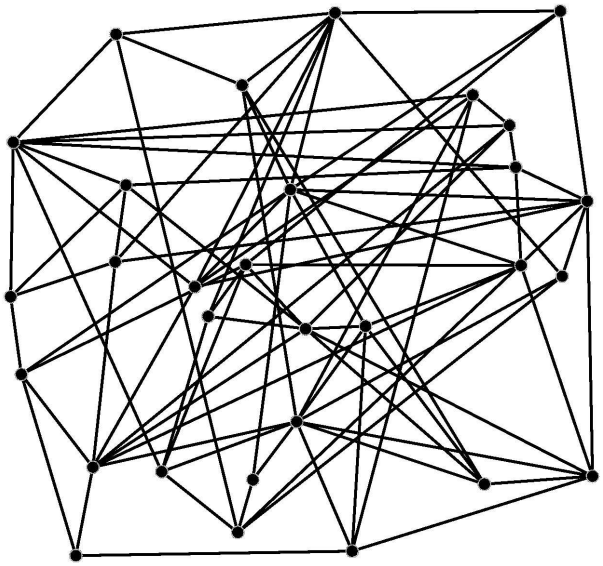

Figure 3.2: Graph of Network

of the network and this gives a better understanding on how the connections between the nodes are established. Each 1 tells that there is a link between the nodes. As this is undirected it is symmetric about the diagonal. The values in the diagonal are the number of link for that node. Connections between nodes are in advance chosen randomly.

Each node have will get connected to $(maxnodes - nodenumber)/6$ higher numbered nodes. This approach is so that node 1 can connect to nodes between 2 and 30, node two to nodes between node 3 and 30 and so forth. The reason for the higher numbered nodes only can connect to fewer nodes is that they might already have connections from a lowered number node and this is to limit the number of connection not to make the network to dense. The number of connections between nodes varies from 3 to 10 connections. One drawback with this approach is that there can be few connections between higher numbered nodes.

3.7 Test application

Each node in the network runs an application to handle the necessary communication and functions. The application has four main functions:

- Prepare files for backup
- Transfer blocks
- Search for blocks
- respond/forward searches

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
1	5	0	1	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0
2	0	5	0	0	0	0	0	0	1	0	0	0	0	0	1	1	0	1	0	0	0	0	0	0	0	1	0	0	0	0
3	1	0	6	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	0	0
4	0	0	0	5	0	0	0	1	1	0	1	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0
5	0	0	0	0	5	0	0	0	0	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0	0	0	1	1	0	0
6	0	0	0	0	0	4	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	1	0
7	0	0	0	0	0	0	4	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	1
8	0	0	1	1	0	0	0	6	0	1	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0
9	0	1	1	1	0	1	0	0	8	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	1	0
10	1	0	0	0	0	0	0	1	0	6	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	1	1	0	0
11	0	0	0	1	0	0	0	0	0	0	5	1	0	0	0	0	0	0	0	0	1	0	1	0	0	1	0	0	0	0
12	1	0	0	0	0	0	0	0	0	0	1	5	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	1	0	0
13	0	0	0	0	0	0	0	0	1	0	0	0	4	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	1	1	0	0	0	0	1	6	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	1
15	0	1	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	1	0	0	1	0	0	1	0	0	0	0	0	0
16	0	1	0	0	1	1	1	0	0	0	0	0	1	0	0	8	1	0	1	0	0	0	0	0	1	0	0	0	0	0
17	0	0	0	0	1	0	0	1	1	1	0	0	1	0	0	1	9	0	0	0	0	1	0	1	1	0	0	0	0	0
18	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	5	0	0	1	0	1	0	0	0	0	0	0	0
19	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	4	0	1	0	0	0	0	0	0	0	1	0
20	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	1	0	1	0	0	0	0
21	0	0	0	1	0	0	0	0	0	0	1	1	0	0	1	0	0	1	1	0	8	0	0	0	1	0	0	0	0	1
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	3	0	0	0	0	0	1	1	0
23	0	0	1	0	0	0	1	0	0	1	1	0	0	1	0	0	0	1	0	0	0	0	8	0	0	1	1	0	0	0
24	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	0	0	1	0	0	0	5	0	1	0	0	0	0
25	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	1	1	0	0	0	1	0	0	0	6	1	0	0	0	0
26	0	1	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	1	0	0	1	1	1	8	0	1	0	0
27	0	0	1	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	5	1	0	0
28	1	0	1	0	1	1	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	10	0	1
29	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	5	1
30	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	1	1	5

Table 3.5: Network Adjacency Matrix

Before a file can be backed up into the network it has to be identified, divided into blocks and the file block created/updated. A block file is a part of the file. In addition to the part of the file it also contains some additional metadata like who created it, when and size.

3.7.1 Data files

There are two different types of files created that are backed up. One called ”Block file” and the other ”File Block List”. The ”Block file” is the most numerous files and they contain the data of the files and have some additional meta data. The ”File Block List” is unique for each file that is backed up. This file have information about the original location of the file, lists of all the blocks that belongs to the different versions of the file, size of file and block size of the data. One major difference between these two files is that ”Block files” can be shared among several users out in the network. ”File Block List” files are unique for each source. For added security ”Block files” can use convergent encryption while ”File Block List” files should use some form other form of encryption. No form of encryption is implemented in the application. Both files are stored in bencode format like the one used in bittorrent [bts05]. More about this format can be found in appendix A. All the blocks are stored with a filename based on their hash value. This hash value is 20bytes long. To prevent the filenames from containing invalid characters is it written with hexadecimal characters so each byte is written using 0-9 and A-F making each filename 40 characters long.

3.7.2 Searching for files

The search method implemented in the application is very simple. The hash values for all blocks from that host are read from a folder where all those blocks are stored. In a real scenario this would be based on the content of the ”File Block Lists”. This implementation however does not look for ”File Block Lists” in the network, but only for ”Block files”. The search propagates into the network breadth first. When a search is started a list of wanted blocks are sent to all neighbour nodes. This can be one request. When a search is received it will forward this to all its neighbours. The exception to this is that it will not send the same information twice. If already received a search for the same block from the same origin (the node that started the search) it will not be forwarded. The node will also look to see if it has any of the requested blocks and if so respond directly to the origin of the search. This response will contain information of which blocks it have and the hop count of the search. Testing to see to see if block is present is just a check if a file with that name is present.

For each response received on the search an entry is stored in a log like the one in figure 3.3. The format of the file is timestamp, IP of host that have the block, hop count and last hash value. The line with ”Search started” is a separation between each search to simplify later processing of data.

3.7.3 Controlling the application

To ease the management of all 30 nodes they can be controlled remotely. Another application sends the appropriate commands to perform the experiment.

```
1115341950 10.1.18.2 3 f520edc41113cb9a4d67726757fb10db3de109a9
1115341950 10.1.18.2 3 f9efd72c03b1a91db3939c420b27b4f560e9b1c2
1115341950 10.1.18.2 3 fb9e336df7c70bba9c3f28bcbe56b4e6951f2b73
1115341950 10.1.18.2 3 fc48c9bec2741b7c3c045fd730a7f3bfcac6f9b5
1115341950 10.1.18.2 3 fd3fafad4dc75ffffb2ef5e9206fcc99522cda33
1115341950 10.1.18.2 3 fe7f47315bd8372136da40e3f3219eeb3f377fc3
1115342021 --- Search started -- 6
1115342021 10.1.16.2 1 00978d49cc6dd43f381a11396bbd3f3ab108e6c9
1115342021 10.1.16.2 1 03322b30528741ee6ce9f3057a368c6d6a16812d
1115342021 10.1.16.2 1 05ef37266d25534f22f519fb2a5a0d31590aafdb
1115342021 10.1.16.2 1 0632f1a66c36da25497b11da57aa818062ed23f4
1115342021 10.1.16.2 1 071da03b6e4b0b8a925eb6053199b0c36561efdc
1115342021 10.1.16.2 1 088da4c3dbcbf5b6e2a1fe7cd6fe64faa1aba899
```

Figure 3.3: Example of log file

3.8 Distributing data

For the experiment to have something to measure it needs some kind of data. Files from the local ”/usr/bin” folder was chosen. Of the 343 files available at the time 311 were selected to be used for the experiment. The files removed were files that were smaller than 30kB and larger than 1.1MB. One reason for this was to remove files that only would become one block and files that would become many blocks. Every file was then divided into blocks. All blocks created are 16kB except the last block of each file that depends on how much is left of the file. Each host then choose randomly 25 files each. The block for these files was then randomly distributed to 5 other nodes. For this experiment it was chosen to distribute out the blocks randomly without any form of connection between parts belonging to the same file. In this approach would the chance that all parts of one file ending up be as shown in equation 3.4.

$$f = (\frac{c}{n-1})^b * s \tag{3.4}$$

The chance for a complete file (f) on one host being number of copies (c) of each block divided on number of nodes (n), (-1) is for not distributing to itself, in the power of the number of blocks (b) in the file multiplied with the number of other nodes having the same file (s). Each block contains the source of the block. If a node received more copies of the same block the source was added to that block making each node only have one copy of each block. The ”File Block List” files is not distributed and used in the experiment.

Some blocks are equal in several files. There is a total of 21 blocks that occurred two or three times. These blocks were present in a total of 13 files. 9 of these files were identical with one or two other files. Of the four last files there were only partial matches between the files. Three out of sixteen blocks matched for the first pair and two out of three for the second pair. This shows that the method used here to find identical blocks is not the most optimal solution.

3.9 Performing the experiment

Basically all nodes perform a search with different number of nodes. Starting from 30 nodes and reduce it down to 1. Performing search with only one node is not necessary as it would not give any valuable results. Only one node performs searches at a given time. This will prevent the nodes from interfering with each other. As delay of searches are not measured should other traffic not make any impact on the results. However in worst case scenarios could other traffic create situation were connections between nodes are lost. This could impact the results as connections are not re-established. To make it possible to get results from all the nodes searches have to be done in several batches. Table 3.6 shows what nodes are tested on the same time. Each row in the table represents number of nodes that is in the network when a search is performed. The top row tells what node belongs to that column. The first batch is the easiest to understand. Here we start with all 30 nodes and remove one after each other until there is only one left. What node to remove next was randomly chosen. The first batch performs searches for all remaining nodes. For the second batch the removal order was the opposite of the first one, having node 15 removed even before any search is performed. This time however searches are not performed by every node only by the ones show in table 3.6. These two first batches would then have performed searches for about 75% of the searches. The next searches would then divide the table in two so search 3 would use nodes from right half and search 4 would use nodes from left half. Batch 5 to 8 is dividing nodes in similar manner. The batches over 9 however are only using nodes marked with appropriate number. The reason the numbers ends up as triangle is that when it gets half way thru the number of searches, the number of nodes that should be accessible is less than the available number of nodes. The results for each search are stored log files for each node.

3.10 Process results

To process the log files they get divided up into parts for each search. This will generate 30 measurements for each of the different number of nodes, for 30 nodes, 29 nodes and so on. The hash values in these files are then used to figure out what files they belong too. The number of occurrences of each file is then calculated. Distance to the files is also preserved by storing values of how many copies are available within a distance of different number of hops. Several hosts can have made backup of the same files. This would make a big difference in the number of copies of different files. To compensate for this we can normalize the results so that we say that the maximum number of available copies is 100%. The number of copies at different hops would then be more comparable.

	29	14	16	18	25	2	6	7	30	19	28	12	8	5	21	26	9	22	24	1	11	20	13	23	3	27	17	10	4	15
30	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
29	2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
28	2	2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
27	2	2	2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
26	2	2	2	2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
25	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
24	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
23	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
22	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
21	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
20	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
19	2	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
18	2	2	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
17	2	2	2	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
16	1	2	2	2	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
15	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
14	2	2	2	2	2	2	2	2	2	2	2	2	2	2	4	3	1	1	1	1	1	1	1	1	1	1	1	1	1	1
13	2	2	2	2	2	2	2	2	2	2	2	2	2	4	4	3	3	1	1	1	1	1	1	1	1	1	1	1	1	1
12	2	2	2	2	2	2	2	2	2	2	2	2	4	4	4	3	3	3	1	1	1	1	1	1	1	1	1	1	1	1
11	2	2	2	2	2	2	2	2	2	2	2	4	4	4	4	3	3	3	3	1	1	1	1	1	1	1	1	1	1	1
10	2	2	2	2	2	2	2	2	2	2	4	4	4	4	4	3	3	3	3	3	1	1	1	1	1	1	1	1	1	1
9	2	2	2	2	2	2	2	2	2	4	4	4	4	4	4	3	3	3	3	3	3	1	1	1	1	1	1	1	1	1
8	2	2	2	2	2	2	2	2	4	4	4	4	4	4	4	3	3	3	3	3	3	3	1	1	1	1	1	1	1	1
7	2	2	2	2	2	2	2	8	4	4	4	4	4	4	4	3	3	3	3	3	3	3	7	1	1	1	1	1	1	1
6	2	2	2	2	2	2	8	8	6	4	4	4	4	4	4	3	3	3	3	3	3	5	7	7	1	1	1	1	1	1
5	2	2	2	2	2	8	8	8	6	6	4	4	4	4	4	3	3	3	3	3	5	5	7	7	7	1	1	1	1	1
4	2	2	2	2	8	8	8	8	6	6	6	4	4	4	4	3	3	3	3	5	5	5	7	7	7	7	1	1	1	1
3	2	2	2	9	9	8	8	8	6	6	6	9	4	4	4	3	3	3	10	5	5	5	7	7	7	10	10	1	1	1
2	2	2	11	9	9	11	8	8	6	6	12	12	13	4	4	3	3	13	14	14	5	5	7	7	15	10	10	15	1	1

Table 3.6: Overview nodes in each searh

Chapter 4

Results

4.1 Analyseing distribution of source data

311 files were selected and files targeted for backup among the nodes. Because each node was to have 25 files each, there was a total 750 files to be used for backup in the network. From this we can see that many files would come from several sources. Table 4.1 describes how these 750 files are distributed among the nodes. 86 files only exist once in the network, while 2 files exist 7 times. The largest number is it of files with two copies, counting 113 files. If we start to sum up the numbers in the table we see that there is only 725 files distributed and not 750 that there should be from our calculations. Further study of the files given out to nodes shows that 25 files has been given to the same node two times. This is the reason for the difference. Because the searches are based on blocks and that this error happened before any data is distributed from the source node, it will only affect how many files/blocks each node has. Instead of 25 files on each node will the number now vary between 25 and 22 files.

Count	Frequency
1	86
2	113
3	63
4	28
5	16
6	3
7	2
Sum	311

Table 4.1: Frequency of distribution of same file

With the varying number of blocks from file to file will the number of block not be proportional with the then number of files. Table 4.2 shows how the frequency of how many times the same blocks has been distributed out. As each block should be distributed out 5 times this count will always be multiple of 5. In the actual experiment the results do however show that of some files

there exists only 4 complete copies. Further investigation to the reason of this shows that by mistake has some nodes distributed the data to itself. When a search is performed in the experiments it does not look for matching blocks in its own storage of blocks received from other nodes. Because of this will other nodes searching for the same block have a small chance of getting more copies than what they would normally. It will however only happen for files that other nodes have distributed to themselves and that not would be distributed to the searching node and that the searching node not have distributed the files to itself.

Count	Frequency
5	965
10	1292
15	741
20	421
25	189
30	48
35	8

Table 4.2: Frequency of the same block distributed out

In table 4.3 we can see how many blocks have ended up on the same host. One host holding backups of the same block this many sources with other words. We can see from the table that 720 of the blocks stored only represent data from one source. 1 block however is the storage for data from 17 blocks that was sent out to be backed up. This difference will have impact on searches performed. Each time space is saved when different copies of the same block are backed up to the same node will also the number of possible responses on searches for that block be reduced.

4.2 Importance of nodes

With so many nodes interacting in the network figuring out how important the different nodes are can help understand why something happens when the networks changes. Using the Eigenvalue method described in chapter 3.4 we can rank the nodes. The rank is based on the adjacency matrix in table 3.5. Because the experiment had to be run in several batches will the rank of all nodes for the different number of running node be presented in several tables. Table 4.4 to 4.13 shows this rank. The first column in each table show how many nodes are running, it will always go from a higher number of nodes and end on 3. Second column shows what node has been removed from the previous row. First row will never have any node number here. The rest of the line is the rank of the nodes. Highest ranked node is on the left and rank decreasing toward the right side. Each number represents the number. Two or more nodes can be placed in square brackets. Such nodes have the same rank.

In table 4.8 we can see that node 25 have the highest rank with 30 nodes but as soon as one node is removed its rank becomes significant lower and only have 8 nodes with a lower rank. We can also see that it gets about the same low rank in table 4.9, also there with 29 nodes. We can see in both table 4.8 and 4.9 that the rank of the nodes doesnt change very much with the reduced number of

Count	Frequency
1	720
2	469
3	291
4	229
5	145
6	125
7	73
8	43
9	34
10	22
11	11
12	4
13	1
14	3
15	1
16	2
17	1

Table 4.3: Frequency of Duplicate blocks to same node

nodes	removed	rank
6		20 ,[1 , 24],[9 , 22 , 11]
5	9	20 ,[1 , 24],[22 , 11]
4	22	20 ,[1 , 24], 11
3	24	[1 , 20], 11

Table 4.4: Rank of nodes, Part 5

nodes	removed	rank
6		28 , 5 ,[12 , 30], 19 , 8
5	5	28 ,[12 , 30],[8 , 19]
4	8	28 ,[12 , 30], 19
3	12	[28 , 30], 19

Table 4.5: Rank of nodes, Part 6

nodes	removed	rank
3		[12, 25], 18

Table 4.6: Rank of nodes, Part 9

nodes	removed	rank
3		[17, 24], 27

Table 4.7: Rank of nodes, Part 10

nodes	removed	rank
30		25 ,[24 , 10], 23 , 15 ,[17 , 14 , 29],[1 , 27 , 8], 9 , 19 , 20 , 30 ,[4 , 7], 16 ,[26 , 18],[2 , 28 , 21 , 11],[6 , 5 , 12 , 13], 3 , 22
29	29	24 , 15 ,[11 , 23 , 14], 28 ,[20 , 4], 30 ,[21 , 8 , 10 , 19 , 2], 16 , 17 ,[9 , 18 , 7],[5 , 25], 12 , 26 ,[1 , 13], 27 , 3 ,[22 , 6]
28	14	11 , 13 , 27 ,[24 , 23], 28 ,[1 , 6 , 15 , 22],[17 , 26 , 9 , 7], 12 , 5 , 30 ,[3 , 20 , 21 , 25], 19 ,[18 , 4],[10 , 8],[16 , 2]
27	16	28 , 26 , 3 , 10 , 23 , 17 , 27 , 8 , 21 , 9 , 25 ,[12 , 1], 11 , 4 , 18 , 5 ,[2 , 24], 15 , 20 , 30 , 22 , 6 , 19 , 7 , 13
26	18	28 ,[27 , 25], 26 , 3 ,[10 , 23], 17 ,[9 , 12], 8 ,[11 , 21 , 1 , 24],[5 , 4], 6 , 20 ,[19 , 22 , 30], 15 , 7 , 2 , 13
25	25	28 , 10 , 3 , 27 , 26 , 23 , 1 ,[8 , 17], 11 , 12 , 9 , 5 , 4 , 21 , 24 , 20 , 30 , 6 , 22 , 2 , 13 , 7 , 15 , 19
24	2	28 , 10 , 3 , 27 , 23 , 26 , 1 , 8 , 17 , 12 , 11 , 5 , 9 , 21 , 4 , 24 , 22 , 20 , 30 , 6 , 13 , 7 , 19 , 15
23	6	28 , 10 , 3 , 27 , 23 , 26 , 1 , 8 , 17 , 11 , 5 , 12 , 9 , 21 , 4 , 24 ,[22 , 20], 30 , 7 , 13 , 19 , 15
22	7	28 , 10 , 3 , 27 , 26 , 23 , 1 , 8 , 17 , 11 , 12 , 5 , 9 , 21 , 4 , 22 , 20 , 24 , 30 , 13 , 19 , 15
21	30	28 , 10 , 3 , 27 , 23 , 26 , 1 , 8 , 17 , 11 , 5 , 12 , 9 , 4 , 24 , 20 , 21 , 22 , 13 , 19 , 15
20	19	28 , 10 , 3 , 27 , 23 , 26 , 1 , 8 , 17 , 11 , 12 , 5 , 9 , 4 , 24 , 20 , 22 , 21 , 13 , 15
19	28	8 , 10 , 23 , 17 , 3 , 26 , 11 , 27 , 4 , 9 , 1 , 24 , 21 , 20 , 12 , 5 , 13 , 15 , 22
18	12	8 , 17 , 10 , 23 , 3 , 26 , 27 , 9 , 11 , 4 , 24 , 1 , 20 , 5 , 13 , 21 , 22 , 15
17	8	23 , 10 , 17 , 27 , 3 , 26 , 11 , 9 , 24 , 1 , 20 , 4 , 5 , 13 , 21 , 15 , 22
16	5	23 , 10 , 3 , 26 , 11 , 27 , 17 , 9 , 24 , 1 , 20 , 4 , 21 , 13 , 15 , 22
15	21	23 , 10 , 3 , 26 , 27 , 17 , 9 , 1 , 24 , 11 , 20 , 13 , 4 , 22 , 15
14	26	10 , 3 , 23 , 27 , 17 , 9 , 1 , 13 , 11 , 24 , 4 , 20 , 22 , 15
13	9	10 , 23 , 27 , 3 , 1 , 17 , 11 , 20 , 24 ,[22 , 13], 4 , 15
12	22	23 , 10 , 27 , 3 , 1 , 17 , 11 , 20 , 24 , 13 , 4 , 15
11	24	23 , 10 , 27 , 3 , 1 , 11 , 17 , 20 , 4 , 13 , 15
10	1	23 , 27 , 10 , 3 , 11 , 17 , 4 , 13 ,[15 , 20]
9	11	[23 , 27], 10 , 3 , 17 , 13 ,[20 , 15 , 4]
8	20	[23 , 27], 10 , 3 , 17 , 13 ,[15 , 4]
7	13	[23 , 27], 10 , 3 , 17 ,[15 , 4]
6	23	[10 , 27],[17 , 3],[4 , 15]
5	3	10 ,[17 , 27],[4 , 15]
4	27	[10 , 17],[15 , 4]
3	17	15 ,[10 , 4]

Table 4.8: Rank of nodes, Part 1

nodes. When the number of nodes gets below 10 we start to see that many nodes begins to have the same rank. A reason to this is that the network begins to become sparsely populated and parts of the network are no longer connected.

When run with only two nodes only two pairs had connection, node 1 to 16 and 12-28. With only two nodes there would not be any difference in the ranking between the nodes. In table 4.8, 4.9, 4.10, 4.11 and 4.13 with 3 nodes there are more than one possible solution. This means that there are several equal Eigenvalues. Order the nodes based on these values would also give a correct view of the rank of the nodes even if it would be different. Table 4.9 and 4.13 has also two similar maximum Eigenvalues at with 4 nodes.

4.3 Connectivity of experiment network

Looking at the connectivity for the network we can see that it starts at less than 20%(see table 4.14.) This is because our network only has 85 links when all nodes are available. The largest possible number of links in out network is given by the following formula:

$$Maxlinks = \frac{N(N-1)}{2}$$

N is here the number of nodes that exist in the network, this is 30 for the network used in the experiments. In total there could be 435 links if all nodes where connected to each other.

nodes	removed	rank
29		[10 , 19], 23 ,[8 , 1 , 3 , 4 , 29],[26 , 27 , 2], 12 , 17 , 30 ,[18 , 13 , 21 , 7], 16 , 20 , 22 , 9 , 24 ,[25 , 5 , 6 , 14 , 11], 28
28	4	6 , 26 ,[9 , 8],[23 , 21], 22 , 12 , 5 , 30 ,[10 , 13 , 14 , 24 , 18 , 28 , 29 , 19], 16 , 17 ,[27 , 2],[20 , 11],[7 , 3], 25 , 1
27	10	[12 , 13 , 28], 5 , 22 ,[8 , 17],[25 , 16], 2 , 23 ,[7 , 30], 27 ,[24 , 3 , 20 , 6 , 29 , 11], 14 ,[1 , 26 , 18],[21 , 9], 19
26	17	24 ,[8 , 27 , 28], 9 , 14 ,[22 , 23 , 3 , 19 , 21], 25 , 26 ,[5 , 20 , 11 , 2 , 18 , 30], 12 ,[29 , 16], 13 , 6 ,[1 , 7]
25	27	26 , 28 , 25 ,[9 , 21 , 23], 12 , 3 , 16 ,[29 , 30], 14 , 6 , 11 ,[2 , 7],[1 , 5 , 8], 24 ,[20 , 13 , 19], 18 , 22
24	3	26 , 28 , 16 , 21 , 25 , 12 , 30 , 23 , 11 , 14 , 6 , 9 , 7 , 2 , 29 , 19 , 5 , 13 , 18 , 24 , 1 , 20 , 8 , 22
23	23	28 , 16 , 25 , 26 , 21 , 12 , 30 , 9 , 6 , 29 , 19 , 2 , 5 , 11 , 14 , 7 , 13 , 1 , 24 , 20 , 22 , 18 , 8
22	13	28 , 26 , 25 , 21 , 12 , 16 , 30 , 6 , 29 , 19 , 9 , 2 , 11 , 5 , 1 , 7 , 14 , 20 , 24 , 22 , 18 , 8
21	20	28 , 21 , 25 , 16 , 12 , 26 , 30 , 6 , 29 , 19 , 9 , 2 , 11 , 5 , 7 , 1 , 14 , 22 , 18 ,[24 , 8]
20	11	28 , 16 , 25 , 30 , 6 , 29 , 21 , 19 , 12 , 9 , 26 , 5 , 2 , 7 , 14 , 22 , 1 , 18 ,[24 , 8]
19	1	16 , 28 , 25 , 29 , 30 , 6 , 21 , 19 , 9 , 26 , 2 , 5 , 12 , 7 , 14 , 22 , 18 ,[24 , 8]
18	24	16 , 28 , 25 , 29 , 6 , 30 , 21 , 19 , 9 , 2 , 5 , 26 , 12 , 7 , 22 , 14 , 18 , 8
17	22	16 , 25 , 28 , 21 , 6 , 30 , 19 , 9 , 29 , 2 , 26 , 5 , 12 , 7 , 14 , 18 , 8
16	9	16 , 21 , 28 , 30 , 25 , 19 , 5 , 12 , 6 , 26 , 7 , 29 , 2 , 14 , 18 , 8
15	26	16 , 21 , 30 , 19 , 28 , 5 , 25 , 6 , 29 , 7 , 12 , 14 , 2 , 18 , 8
14	21	16 , 28 , 5 , 30 , 6 , 19 , 7 , 29 , 14 , 25 , 12 , 2 , 8 , 18
13	5	16 , 30 , 7 , 6 , 29 , 28 , 14 , 19 , 25 , 12 , 2 , 8 , 18
12	8	16 , 30 , 7 , 6 , 29 , 28 , 14 , 19 , 25 , 12 , 2 , 18
11	12	16 , 30 , 7 , 6 , 29 , 14 , 19 , 28 , 2 , 25 , 18
10	28	16 , 7 , 30 , 29 ,[6 , 19], 14 , 2 , 25 , 18
9	19	7 , 16 , 30 , 14 , 6 , 29 , 2 , 25 , 18
8	30	16 ,[6 , 7 , 2], 25 ,[18 , 14 , 29]
7	7	16 ,[6 , 2], 25 ,[18 , 29], 14
6	6	[2 , 16],[18 , 25],[14 , 29]
5	2	[16 , 25],[18 , 14 , 29]
4	25	29 ,[16 , 18 , 14]
3	18	29 ,[16 , 14]

Table 4.9: Rank of nodes, Part 2

nodes	removed	rank
14		23 , 10 , 3 , 27 , 26 , 17 , 9 , 1 , 11 , 24 , 20 , 13 , 4 , 22
13	4	23 , 10 , 3 , 27 , 26 , 17 , 1 , 24 , 20 , 9 , 11 , 13 , 22
12	10	26 , 23 , 3 , 24 , 20 , 11 , 17 , 9 , 27 , 1 , 13 , 22
11	17	23 , 26 , 3 , 20 , 11 , 27 , 24 , 1 , 9 , 13 , 22
10	27	26 , 23 , 20 , 11 , 24 , 3 , 1 , 9 , 13 , 22
9	3	26 , 20 , 24 ,[11 , 23], 1 ,[13 , 9 , 22]
8	23	[20 , 26], 24 ,[11 , 1],[22 , 13 , 9]
7	13	[20 , 26], 24 ,[11 , 1],[9 , 22]
6	20	26 ,[24 , 11],[22 , 9 , 1]
5	11	[24 , 26],[22 , 1 , 9]
4	1	[24 , 26],[9 , 22]
3	24	26 ,[22 , 9]

Table 4.10: Rank of nodes, Part 3

nodes	removed	rank
14		16 , 21 , 30 , 19 , 28 , 25 , 5 , 12 , 7 , 6 , 14 , 2 , 18 , 8
13	14	16 , 21 , 19 , 25 , 5 , 28 , 12 , 30 , 6 , 7 , 2 , 18 , 8
12	16	21 , 12 , 28 , 30 , 25 , 19 , 5 , 18 , 6 , 7 , 2 , 8
11	18	21 , 12 , 28 , 30 , 25 , 19 , 5 , 6 , 7 ,[2 , 8]
10	25	28 , 30 , 21 , 12 , 5 , 19 , 6 , 7 ,[8 , 2]
9	2	28 , 30 , 21 , 12 , 5 , 19 , 6 , 7 , 8
8	6	[21 , 28], 30 , 12 ,[19 , 5], 7 , 8
7	7	[21 , 28],[12 , 30],[19 , 5], 8
6	30	[5 , 12 , 19 , 21 , 28], 8
5	19	[12 , 28],[21 , 5], 8
4	28	[12 , 21],[5 , 8]
3	12	21 ,[8 , 5]

Table 4.11: Rank of nodes, Part 4

nodes	removed	rank
7		[23 , 27], 10 , 3 , 17 , 13 , 4
6	4	[23 , 27], 10 , 3 , 17 , 13
5	10	[3 , 23 , 27],[13 , 17]
4	17	[3 , 23 , 27], 13
3	27	[3 , 23], 13

Table 4.12: Rank of nodes, Part 7

nodes	removed	rank
7		16 ,[7 , 2],[6 , 25],[18 , 14]
6	14	16 , 2 ,[7 , 6 , 25], 18
5	16	[2 , 18],[7 , 6 , 25]
4	18	25 ,[6 , 7 , 2]
3	25	7 ,[6 , 2]

Table 4.13: Rank of nodes, Part 8

After 9 nodes have been disconnected the connectivity is down to 10%.

The connectivity drop further down to 5% after 16 nodes are removed (see table 4.14 and 4.15.) As more nodes are removed the connectivity drops fast. With only 6 nodes left, the connectivity starts to drop below 1%. In table 4.16 right, no nodes can even talk to each other when there is less than 6 nodes left. We can see from all tables (table 4.14 to 4.17), that when the number of nodes left are few the network gets useless to use for the purpose of this experiment.

4.4 Retrivable copies

In figure 4.1 we can see how the number of files that is available changes depending on the number of hosts running. As long as there is over 13 hosts running almost all files can be retrieved. At this stage there are 47 files that can not be retrieved of the 726 files that can be reached when there are over 22 hosts running. From 13 hosts the number of files that can be retrieved drops dramatically down to 29 files at 4 hosts. With less than 4 hosts none of the files can be retrieved.

Figure 4.2 shows the number of available copies of files based on the number of hosts. This graph is not so stable with many hosts as figure 4.1. The number of files starts to drop from the start when numbers of host are reduced. This graph looks quit linear with a drop of a little over 300 copies for each host removed. The reason for this constant drop is that the number of hosts decreases. It is then one less host to find blocks on and the number of copies will get smaller as the result of less available blocks to make up complete files. When the number of hosts gets small, the number of copies of files is not much larger than the number of files that can be retrieved. That the numbers of copies are so much larger than the number of files, over ten times, even if each host only distributes at most five copies of the files comes from the fact that the same file comes from several hosts. This makes more copies available when searching.

That the maximum number of files available is 726 even if we sum up the number of copies from table 4.1 we should only have 725 files. One explanation of this difference can be that the files that consist of the same blocks can make more copies available than what was distributed into the network.

Number of nodes	Removed	Connectivity	Removed	Connectivity
30		0.1954		
29	29	0.1839	15	0.1862
28	14	0.1701	4	0.1747
27	16	0.1517	10	0.1609
26	18	0.1402	17	0.1425
25	25	0.1287	27	0.1333
24	2	0.1218	3	0.1218
23	6	0.1172	23	0.1103
22	7	0.1126	13	0.1034
21	30	0.1080	20	0.0966
20	19	0.1034	11	0.0897
19	28	0.0851	1	0.0851
18	12	0.0782	24	0.0805
17	8	0.0667	22	0.0759
16	5	0.0621	9	0.0667
15	21	0.0552	26	0.0575
14	26	0.0460	21	0.0460
13	9	0.0368	5	0.0391
12	22	0.0345	8	0.0368
11	24	0.0276	12	0.0322
10	1	0.0207	28	0.0276
9	11	0.0161	19	0.0230
8	20	0.0161	30	0.0161
7	13	0.0138	7	0.0115
6	23	0.0069	6	0.0069
5	3	0.0046	2	0.0023
4	27	0.0023	25	0.0000
3	17	0.0000	18	0.0000
2	10	0.0000	16	0.0000
1	4	0.0000	14	0.0000
0	15		29	

Table 4.14: Connectivity of network Part 1

Number of nodes	Removed	Connectivity	Removed	Connectivity
14		0.0529		0.0506
13	4	0.0483	14	0.0437
12	10	0.0391	16	0.0299
11	17	0.0299	18	0.0253
10	27	0.0253	25	0.0207
9	3	0.0184	2	0.0207
8	23	0.0046	6	0.0184
7	13	0.0115	7	0.0161
6	20	0.0046	30	0.0115
5	11	0.0023	19	0.0069
4	1	0.0023	28	0.0023
3	24	0.0000	12	0.0000
2	22	0.0000	8	0.0000
1	9	0.0000	5	0.0000
0	26		21	

Table 4.15: Connectivity of network Part 2

Number of nodes	Removed	Connectivity	Removed	Connectivity
6		0.0046		0.0023
5	9	0.0046	5	0.0000
4	22	0.0046	8	0.0000
3	24	0.0023	12	0.0000
2	1	0.0000	28	0.0000
1	11	0.0000	19	0.0000
0	20		30	

Table 4.16: Connectivity of network Part 3

Number of nodes	Removed	Connectivity	Removed	Connectivity
7		0.0161		0.0138
6	4	0.0161	14	0.0115
5	10	0.0092	16	0.0023
4	17	0.0069	18	0.0000
3	27	0.0023	25	0.0000
2	3	0.0000	2	0.0000
1	23	0.0000	6	0.0000
0	13		7	

Table 4.17: Connectivity of network Part 4

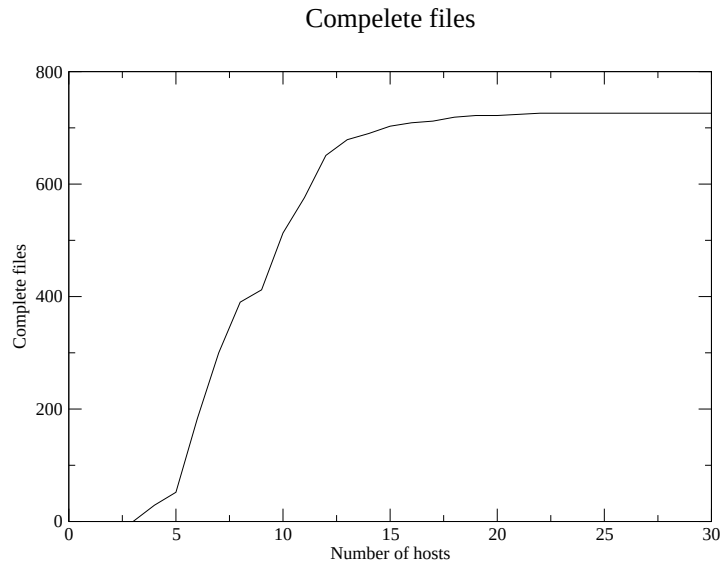

Figure 4.1: Files available with increasing number of hosts

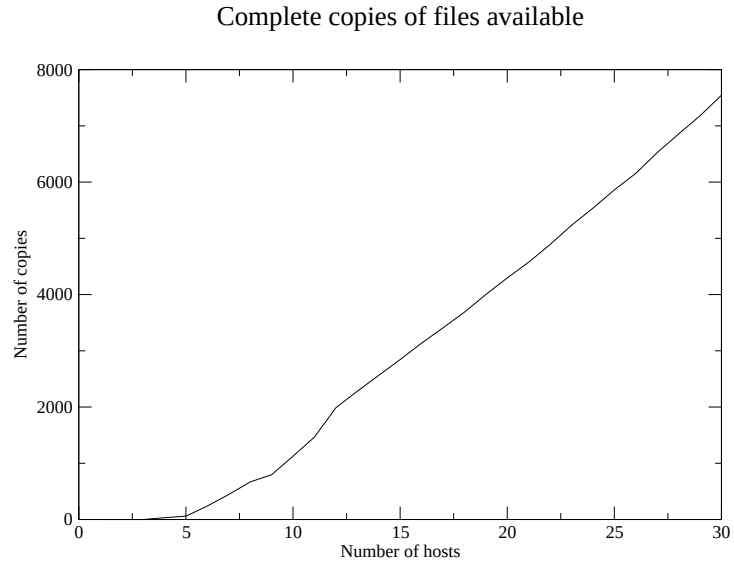

Figure 4.2: Copies available with increasing number of hosts

4.5 Distance to data

The graphs in figure 4.3 to 4.9 and in appendix B shows how many copies are available at a distance of up to the given hop counts show in the graphs. The left graph shows the real values while the right one it is normalized to the maximum available copies for that file. The horizontal line shows the average number of copies. The standard deviation of the values is represented with the bar. For each hop is the number of available copies for each file plotted in the graph. This value is based on the number of complete copies and the largest value of incomplete copies. If a file has all copies of one block available it will still not count more to the total number of copies of the block with the second smallest value. All the blocks with the smallest value would be the ones that count toward the complete number of copies. If there are no complete copies this value would be the only one used. Each value at the hopes shows the number of copies that are available at up to that many hops away from origin of the file. It is the sum of the number of copies at all lower hop counts. On the right graph all values should end up with 100% of the total available copies as this is the base for the normalization.

A selection of the most interesting graphs is selected for a further study. These are graphs that shows some changes in the values as the number of nodes changes.

4.5.1 30 nodes

With all nodes present should all data be retrievable within some distance. From the graph (figure 4.3) we can see that within a distance of four nodes can we reach all the other nodes. The average number of copies is here about 11 copies and it has a standard deviation of about 4 copies. We can also se that there are a few values below the standard deviation of the graph. These values are of two different types, files that the maximum available copies are below the standard deviation threshold and leftovers from blocks that exists in several files. The leftover blocks are from the files that only shared a subset of the blocks. The values that are over the standard deviation are most likely from files that have origin in many nodes. We can also see that up to seven copies of some files can be retrieved from only a distance of one hop. Most values lies however under 3 copies. At a distance of two hops almost all files can be found with at least on copy.

From the right graph in figure 4.3 we can see that few nodes can reach all possible copies before we get to a distance of 4 hops, were all seems to be able to retrieve all copies. The complete copies within 1 hop is most likely some of the extra copies that is in the bottom of the graph on the left side.

4.5.2 24 nodes

From the graph in figure 4.3 to the graph in figure 4.4 we see a steady decrease in the average. It has sunk to a value of 11 copies. The highest number of available copies has also sunk from over 20 to less than 17. The values that are under the standard deviation have increased. Most copies can still be reached within a distance of 4 hops. The number of copies within 1 hop is still about the same as it was with 30 nodes. This is most likely caused by the fact that it only relies on if a neighbour node is removed. So far only 6 nodes have been removed and the results are a combination of two batches of the experiment. Only nodes removed that have a block that a neighbour wants will

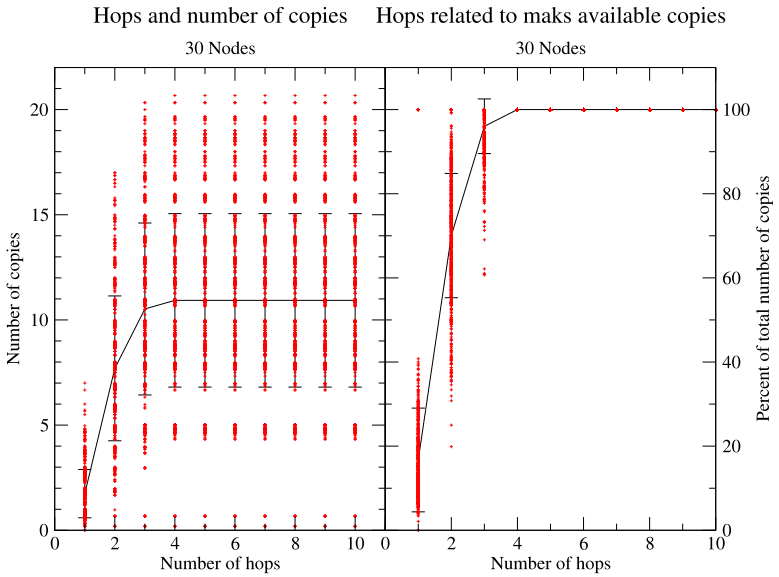

Figure 4.3: Reachable files with 30 nodes

affect the results. The right graph is not too much different from the graph for 30 nodes. There is one noticeable difference however. There is more variation in the number of copies at 4 hops also leading to a larger standard deviation here.

4.5.3 18 nodes

Figure 4.5 shows how the copies are distributed with 18 nodes. The number of files reachable within 1 hop is now reduced to a maximum of 5 cpoies. The average value is about 1, not to far off from the earlier values. The average number of retrievable copies is now a little over 5 at its highest. The number of copies is now spread equally among the different values. The distance to retrieve all available copies is still at about 4 hops. The major difference in this graph compared to the earlier graphs is the in the right graph, plots of the normalized number of copies at hop 3 are now much longer than in the earlier graphs. This shows that the increase in distance from 3 to 4 hops now have a greater impact on the retrievability of files.

4.5.4 13 nodes

Figure 4.6 show results for the network with 13 nodes. This is the one that have to look the furthest to get all copies of the files, a total of 6 hops. The average number of copies is now down to less than 4 copies. Still over half the available copies can be reached within the 2 first hops.

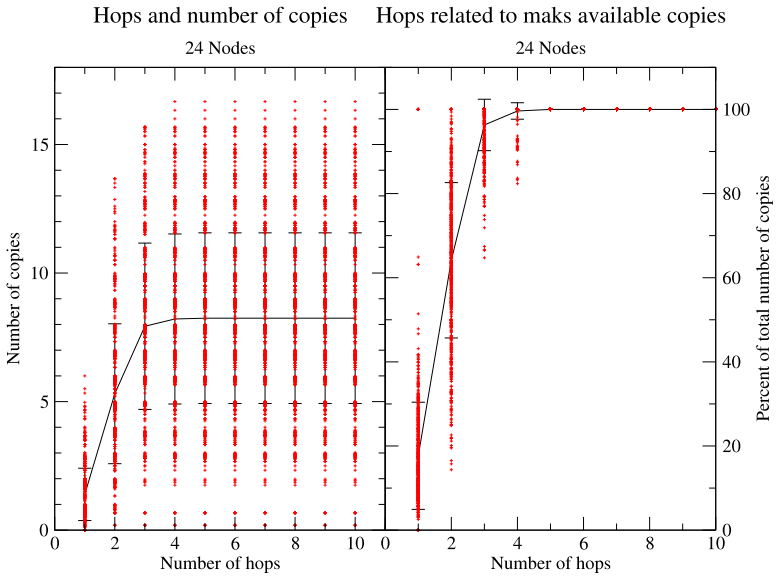

Figure 4.4: Reachable files with 24 nodes

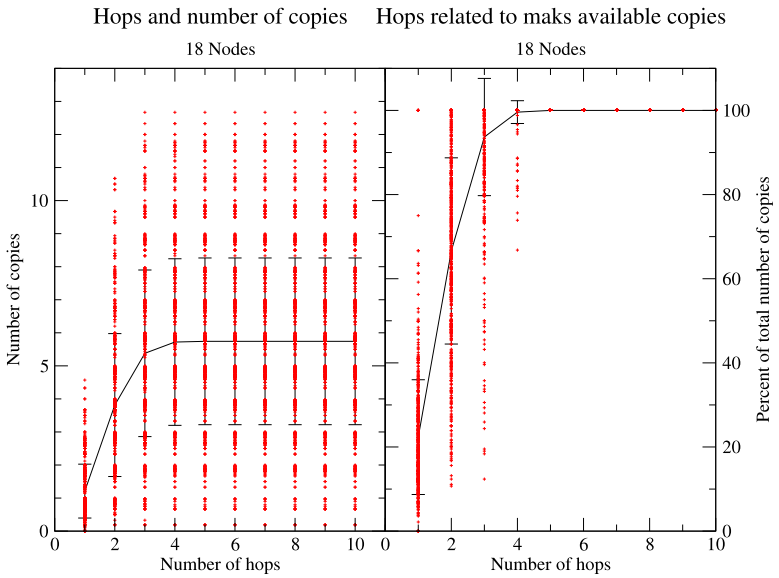

Figure 4.5: Reachable files with 18 nodes

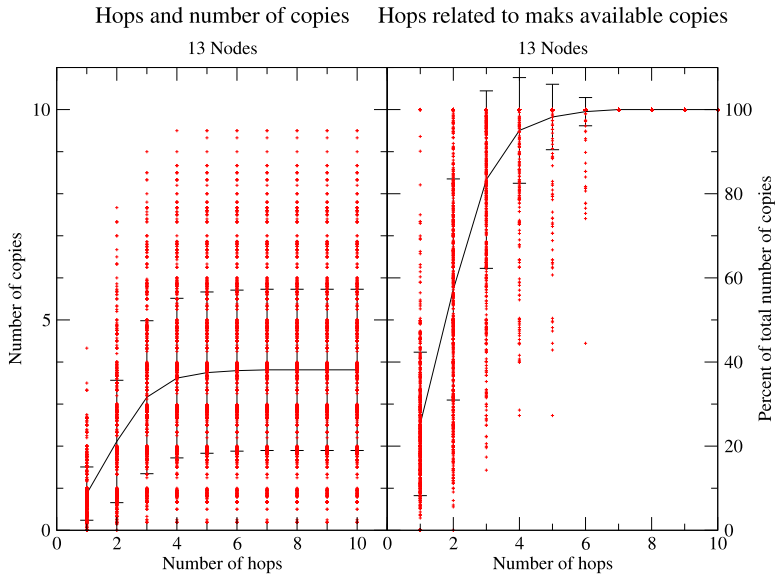

Figure 4.6: Reachable files with 13 nodes

4.5.5 10 nodes

With only 10 nodes left we reach a significant boarder in the results from the experiment. This is the last graph were the standard deviation is above 1 copy. From this stage on will many nodes no longer be able to retrieve complete copies of their files.

4.5.6 6 nodes

At 6 nodes reaches the average down to a level of 1 copy. Half the files have no longer any chance of retrieving a complete copy of the file.

4.5.7 4 nodes

With only 4 nodes as shown in figure 4.9 there are not more than 2 copies available of each file. The average is only a little over half a copy. We can se here that only a few files can be retrieved. Even the standard deviation of the values can only pass the 1 copy line. 90% of the available copies can however be reached within a distance of 1 hop. This is also the last measurements that not reach all available copies within a distance of 1 hop.

4.5.8 Different number of nodes

For the other number of nodes there is not too much that stands out. The graphs for these are shown in appendix A. Going from a network with 30 nodes down to a network with 2 nodes we see a

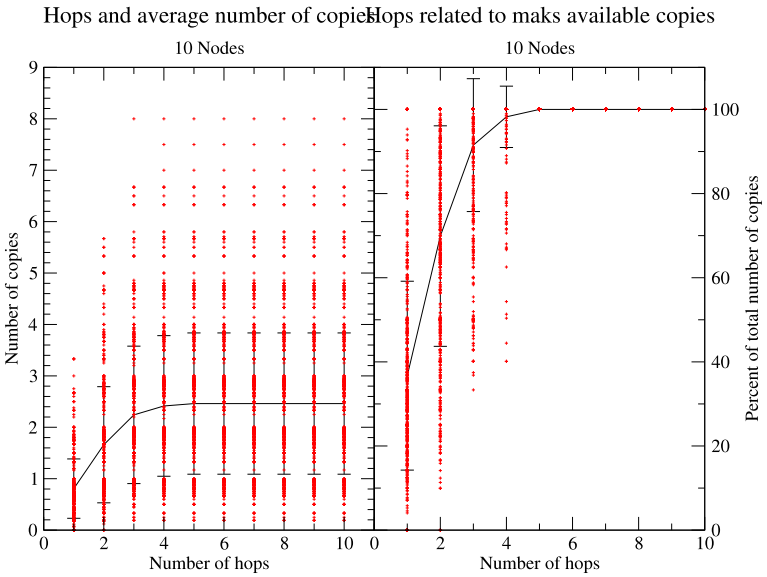

Figure 4.7: Reachable files with 10 nodes

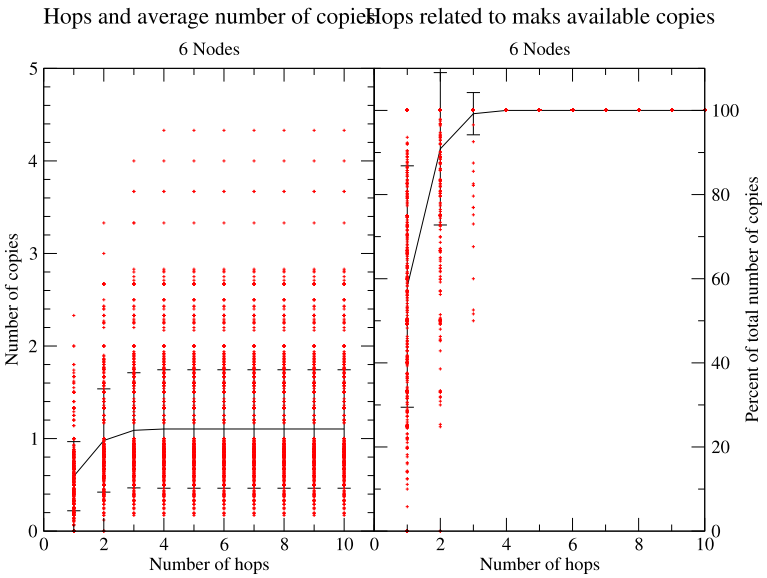

Figure 4.8: Reachable files with 6 nodes

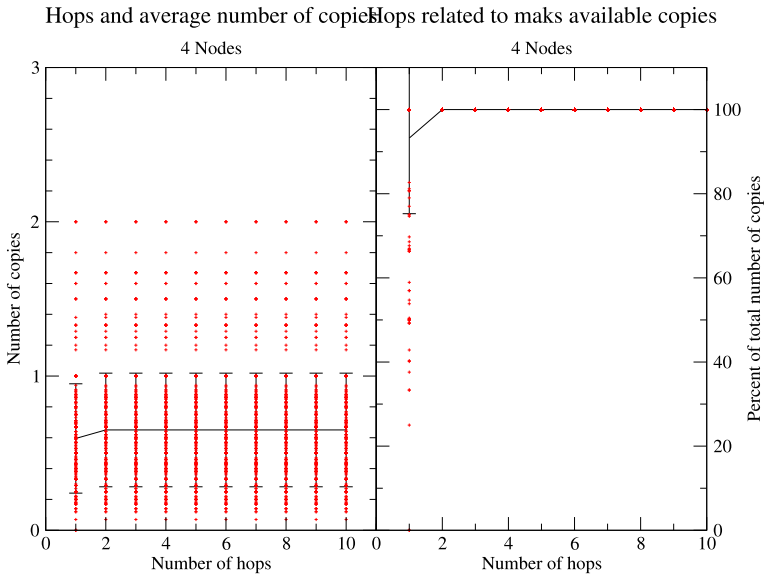

Figure 4.9: Reachable files with 4 nodes

steady drop in the number of available copies. This is to expect as fewer nodes also gives fewer sources and therefore also fewer blocks. With fewer than 4 nodes blocks are also only reachable if there is a direct contact between the two nodes, source and the node that stores the backup.

Chapter 5

Conclusions and Discussion

5.1 Measurements

All the connections between the nodes are based on the same network layout for all searches. The network is however seen from different ”angles” by the different nodes. They see the network as a tree with themselves as the root. Because of this can the different measurement performed at the different nodes be seen as samples of measurements in a larger network. Each measurement does therefore take 30 different samples.

Comparing the results measured in chapter 4.4 to the ones done by Batten, Barr, Saraf and Trepetin in [BBST01] we can see that their network could get 95% of the files after 23 of the nodes are down. Our experiment could only recover 95% of the files with no more than 16 of the nodes down (see figure 4.1.) From this we can see that trying to save the entire file on the same remote host yields better results when it comes to recovery. The reason for this could be that their approach only needs to find one host holding the file while the method we used must find one host for each of the different blocks of the file.

The results show that for most of the measurements all available copies can be reached within a distance of about 4 hops. Only at 12 and 13 nodes can we see that this distance increases with 1 to 2 hops. One reason for this can be that at this point is the number of paths in the graph reduced so much that to reach some nodes must a longer path be taken. If we also look in table 4.9 we see that node 5 is removed here at 13 nodes. This node was the third highest ranked node before this. As this is a high ranked node, many paths between nodes would now have to change. Changing paths would always lead to a longer path. This happens because communication will most likely reach its destination using the shortest path that should use less time.

The network used in the experiments is based on random creation of links among the nodes. This seemed like a good approach to create a network for this experiment. The random selection of links between nodes can however create links that would be very unlikely in a real scenario. Nodes would most likely create clusters of nodes that are close together and these clusters would again communicate with other clusters. Nodes would then only communicate directly with a few close nodes and communication to further away nodes would have to take a longer path to reach its destination. Networks with different nodes clustering together would also increase the chance that different cluster would not be able to communicate with each other. When links between two

clusters is broken there might not be any other paths that connect them or the distance of paths available can be very long, many number of hops.

5.2 Backup models

Implementing backup environments can be seen in the same way as managing other policies. Ad-hoc backup distinguishes itself from other backup methods. We will here try to look at how ad-hoc backup relates to other forms of backup and use the six policy maintenance architectures in Burgess et al [Bur04] to place them.

Backup is a part of host configuration. The current configuration of a host does not only consist of a policy that keeps the system operational, but also locally created. Configuration management tools like cfengine[Bur93] or a complete reinstallation can bring the host back up to a stable state after an unintentional modification of the system. Locally created data on the other hand must rely on some form of backup to be restored. A very common approach for backup is to have routines that periodically take backups of what is stored on central servers. For closely managed applications like databases, web servers and other similar applications and services is it easy for the system administrators to control the backup process. User created data can also have central backup by having all data stored on central shared folders over the network. The decisions that have to be made to perform these types of backups can still be a complicated process.

The six policy maintenance architectures given in Burgess et al [Bur04] is: Star model, Star model in intermittently connected environment, Mesh topology with centralized policy and local enforcement, Mesh topology with partial host autonomy and local enforcement, Mesh topology partial autonomy and hierarchical coalition and Mesh topology with partial autonomy and inter-peer policy exchange. These can also be applied to backup methods.

Model 1: Star model

In this model we have a central manager that decides and control all networks and hosts (see figure 5.1. All connections are also expected to be reliable. This is an idealized case but could to some extent relate to a backup system that only backups data from the local machine. The reliability would then rely on the configuration, that files are where they are supposed to be, backup program, it runs when it is supposed to and that the medium the backup is made to is working. These would be important for any backup to be successful.

Model 2: Star model in intermittently connected environment

This model is a more realistic approach to the first one. It also incorporates the unreliability of the environment. Even if network connection is quite reliable in modern network there are still a chance that it might be down or the host it tries to reach is down. A central server retrieving files from other machines could be an example of this model. In figure 5.2 we can see that the major difference is the connection between the manager and the nodes.

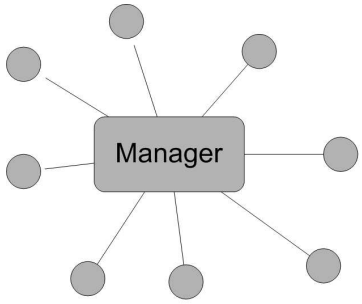

Figure 5.1: Star model

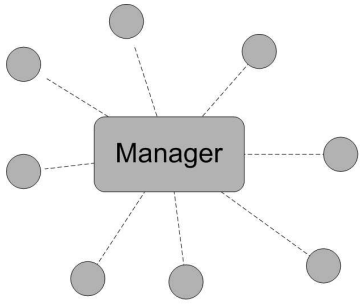

Figure 5.2: Star model in intermittently connected environment

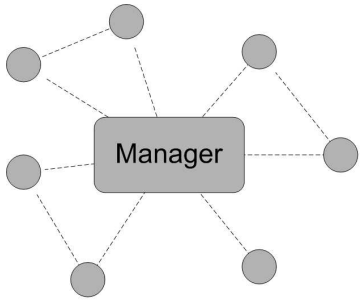

Figure 5.3: Mesh topology with centralized policy and local enforcement model

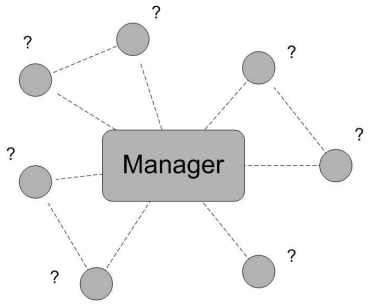

Figure 5.4: Mesh topology with partial host autonomy and local enforcement model

Model 3: Mesh topology with centralized policy and local enforcement

Hosts with this model will be more independent. They will get the policy, what to backup, from a central server but do perform the backup themselves. In a company this could be that employees are obligated to backup the files on their portable devices. This would be defined by the company policy. The user would then have to transfer files for backup to some other medium like a CD or a central storage that gets backed up. There is also now connection directly between nodes (see figure 5.3.)

Model 4: Mesh topology with partial host autonomy and local enforcement

Model 4 is a more loos form of model 3. This would be a not so strict policy. Making it possible to change or choose not to follow certain parts of the policy. Model 4 could be seen more like a recommendation of what to backup. Cfengine [Bur93] uses this model and it can be configured to perform backups working after this model. It can not perform any more strict form of backups as the host can decide by themselves if they want to follow given instructions. The question marks in figure 5.4 represents the freedom of the host.

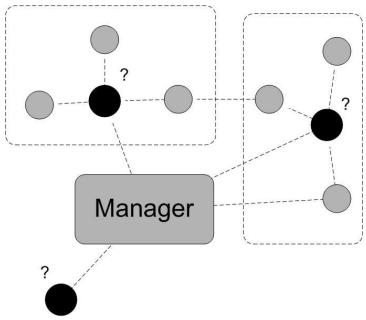

Figure 5.5: Mesh topology partial autonomy and hierarchical coalition model

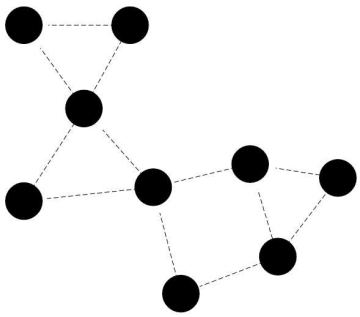

Figure 5.6: Mesh topology with partial autonomy and inter-peer policy exchange model

Model 5: Mesh topology partial autonomy and hierarchical coalition

Hierarchies can be formed of groups that want the same kind of policies (figure 5.5. Each group can then decide what rules they want to follow or not at a central point. Policies can then also be assembled from several groups. When it comes to backup there can be several central controllers that each represents one form of services. These services can specialize in backup of data from a certain application and groups or single hosts can then accept backup policies depending on what their task is. Groups can be made that performs backup of common data that is used by a large number hosts.

Model 6: Mesh topology with partial autonomy and inter-peer policy exchange

In networks without any form of central control each host is responsible for coming up with their own backup plan (see figure 5.6) . They can however share their policies with other hosts. To perform backup not only do they need a policy but they would also need to store their backups somewhere. Other hosts in the network would be able to perform this task. Creating a peer-to-peer based backup solution. This solution would both work ad-hoc networks and more static networks.

5.3 Security

In any form of backup is it important to keep backups safe and prevent unwanted people from accessing the data. Without any form of control over data store on untrusted remote nodes this becomes even more important. Some form of encryption of the data is required. A simple choice would be using some form of symmetric encryption based on a user's password. Using pub/private keys instead would make it possible for signing of data. This would also make it possible to delete previously stored data when not needed anymore and save storage space in the network. For unique data this would be a good choice. Some data on the other hand can be equal. If a recovery is needed but some blocks from the original source are no longer available a complete recovery is no longer possible. But there is still a small chance that someone else also has saved the same block. The chance for two equal blocks from two different sources is higher for program and operating system files. For user data this is more limited to common parts in file formats or several people have the same file. Using convergent encryption [BDET00] will at least increase the chance to retrieve the file a little. This will on the other hand also add a small security risk for other people having the same data to find out who else has the same file. The content would however not be any more compromised as they already have to know the content for any large security risk.

5.4 Implementing a backup application

Creating a backup application meant to work in an ad-hoc network has many limitations not found in regular backup methods.

To make other nodes more willing to keep backups, the work they must perform should use as little as possible of their resources. Most heavy work like hashing and encryption should be done on the source. Not only is this the most logical place to perform these operations but the source is also the one that have anything to benefit of other nodes keeping their backup. Searches should also need as little work as possible of the backup buddies. This would mean having easy ways to identify blocks they have when receiving searches and forward them to other possible nodes keeping the wanted blocks.

Implementing a function to delete unwanted blocks in the network seems like a good way of reducing the data amount in the network. There is however a limitation on how efficient this is. For a block to be deleted some form of authentication have to be implemented. This can be by transmitting the public key with blocks when they are backed up and sign the request for deleting them. The addition of public keys will increase the data amount some, but this should not be a limiting factor. Verification of the requests would also increase the work that has to be done on the nodes keeping the backups. With the layout of the network changing repeatedly, many such a requests might also never reach the nodes. This would infect have less chance to completely succeed than a recovery. A recovery would only need to find one complete copy but a delete request should find as many of them as possible.

With the changes in the network, recovery of files could be a big challenge. If the backup is stored on devices meet on a bus, when performing recovery this device have to be found again. That can be big challenge. Storing on several devices will increase the chance but would still need to find at least one of them. On vacation or business travels this would be even more difficult as you

might not be able to return to the area where the backup was performed. A solution to this would be to move the backups to an Internet based network when synchronization is performed. So just not synchronize your own data but also synchronize data keep for others. Having such a service available it would also be possible to perform backup directly. This would of course require that the device is connected to the Internet in some manner. It could be directly or through other nodes in the ad-hoc network that also are attached to the Internet. This would save space for other data on portable devices as the backup can be freed from these but kept other places. This approach would also make it easier to perform the deletion of data from the network. To even increase the chance to retrieve the backups can they also include an identifier for this recovery network so after the synchronization they can inform the owner were to look for backups if needed.

The internet based network can also be used to identify what nodes are safe to use as backup nodes and find potential bad backup buddies. Every time a backup is made an identifier for the node that keeps the backup is stored. When synchronizing with a central storage this information can be saved and compared with backups arrived over the internet. List over good backup buddies can then be created on the device and these can be preferred when backups are performed. Every time a backup is performed, but no recovery is needed, can then help finding good backup buddies that will increase the chance of successful recovery after loss of data.

5.5 Conclusion

Ad-hoc networks consist of vulnerable devices that are exposed to many dangers. Because these devices are small and often are transported from place to place they can easily be damaged or lost. Information stored on such devices will be updated or add whiteout the possibility of making backups to more permanent forms of storage. If devices are lost or stolen recovery of its content will be very difficult. Recovery of data from damaged devices can be costly. Using other similar devices to backup important data can be a life saver.

With the rapid change of structure in such networks could they benefit from using small blocks that are stored on nodes they meet. Too large block could have problems with not being able to be transmitted correctly if changes happen to frequently. Recovery however benefits from having complete files stored on the same device. Only one with a copy of the file would then be needed to perform recovery. Recovery of files will for the most part not happen when within a short period after the backup is made. The backup might then no longer be available or reachable by any attached ad-hoc network. Recovery is also the most important part of any backup. If recovery can be made then there is also no point in making the backup in the first place. The good solution for backup in an ad-hoc environment would be to combine the ad-hoc network for making backup with a larger internet based network for recovery. Users could then subscribe or have their own more permanent backup server attached to the internet based network. This would not only make recovery easy it can also make it possible for nodes that receive backups for others to free up space. This would help reduce the possibility for old and no longer needed backups remaining in the ad-hoc backup network, removing the problems of implementing methods to deletion of unwanted backups.

Appendix A

File format

The files that is distributed in the backup application uses a format called bencode[bts05] to encapsulate its content.

A.1 Bencode

D5:indexli4e4:texte6:numberi5ee

Table A.1: Example of Bencode

Bencode format consist of four data types: Directory, List, String and Integer. All information is stored using these data types. An example of bencode is shown in table A.1. A bencoded file starts with a directory entry. The directory entry starts with a ”d” as an identifier. Each directory can have an unspecified number of entries. The directory is then finished off with an ”e”. Each entry consists of 2 parts. It first has a string as an identifier. The second one can e any of the four data types. The List is simpler than directories. It starts with an ”l” and ends with an ”e”. Between there can be any number of any of the data types. A string distinguishes itself from the other data types by not having any start character. Instead it starts with a number that tells how many characters there is in the string. A ”:” is between the length and the string. This is so a string can start with a number. Strings are very flexible as they can contain any character. A character can here be though of as any byte value. Strings can therefore be used to hold binary data. Integers are the last of the four data types. It starts with an ”i” is then followed by the number written in text and is terminated by an ”e”.

Using directories to identify the kind of data can then make it possible to encapsulate any kind of data using this encoding scheme.

A.2 File content

Even if both file formats use the bencode format is the content of the files somewhat different. The purpose of the ”Block file” is to be storage for data belonging to files. Everything in this file is

stored in the root directory and using strings as data types. The only exception to this is the owner field that is a list of owners for the file. One file can have several owners when a node has gotten a copy of the block from several sources. This is to save disk space and prevent conflicting filenames. The other values are hash, date, data and size. Date is when the file was made, hash is the hash value for the block and size is the size of the data. Data is the content of that part of the file. The file will contain the length of the data twice. Once in size, but also as the length of the other string belonging to the data entry.

”File Block List” files contain some more information than ”Block files”. The content is here divided in two parts. Basic information added when file first selected for backup and information specific to each version. The basic information is: who is the owner (owner), when was it first created (date) and were the file located in the files system (filename). The ”versions” filed is a directory that have one entry for each version in the file. Each version contains information about when this version was prepared for backup (date), size of file (size), normal size of the blocks (chunksize) and last an entry of all the hash values for all the blocks in that version (hash). The size of the hash entry would be size divided on chunksize rounded up to nearest whole number multiplied by 20. 20 is the size of each hash value. If convergent encryption is to be added another hash entry would also have to be added. The two hash entries would be required because one would be used to decrypt the files and the other to perform searches for them in the network.

Appendix B

Distance to copies graphs

This is the additional graphs that shows the relations between hops and number of copies. Further explanation of the data in these are in chapter 4.5.

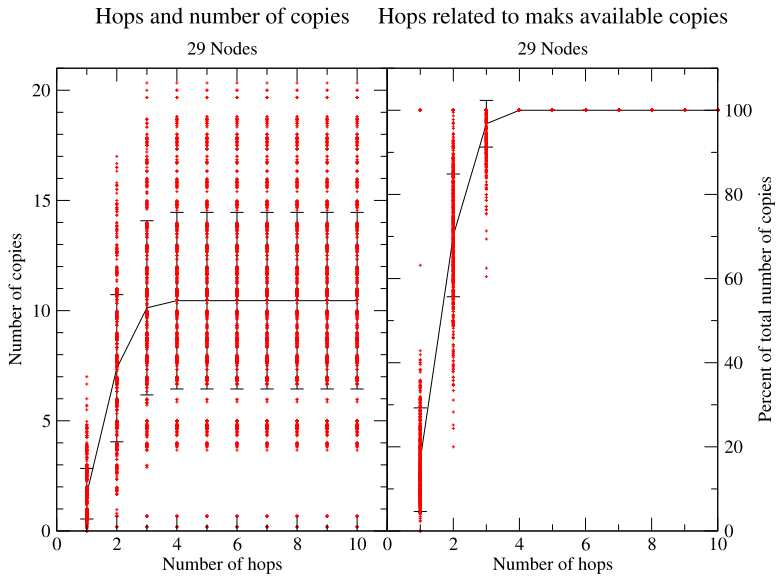

Figure B.1: Reachable files with 29 nodes

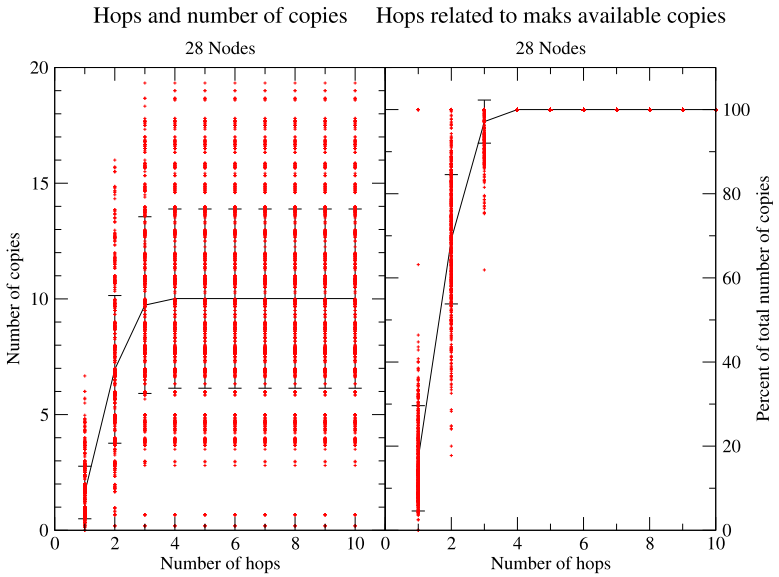

Figure B.2: Reachable files with 28 nodes

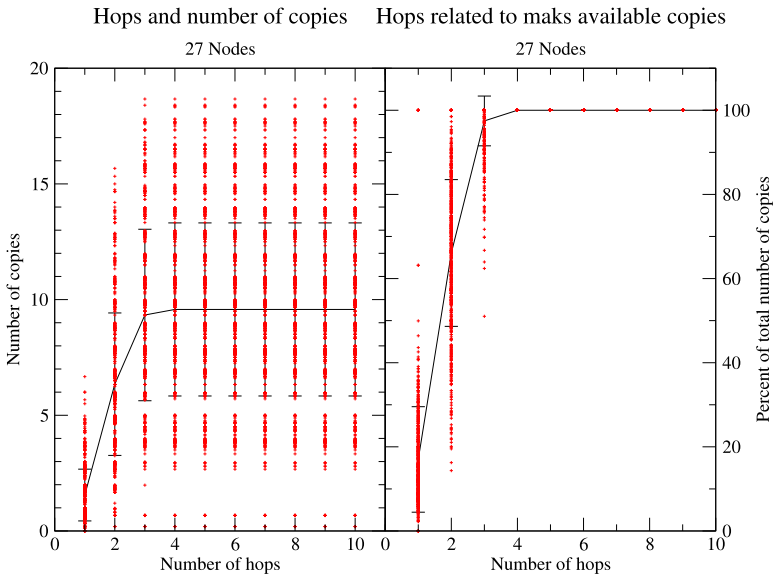

Figure B.3: Reachable files with 27 nodes

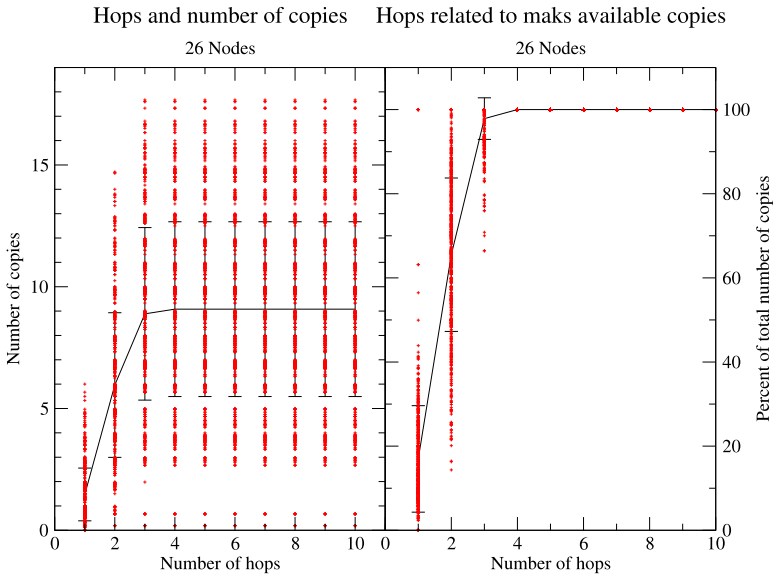

Figure B.4: Reachable files with 26 nodes

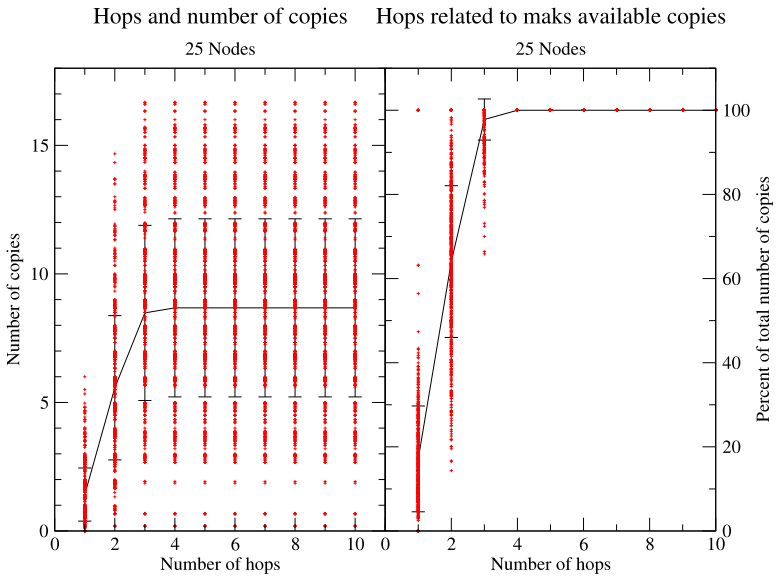

Figure B.5: Reachable files with 25 nodes

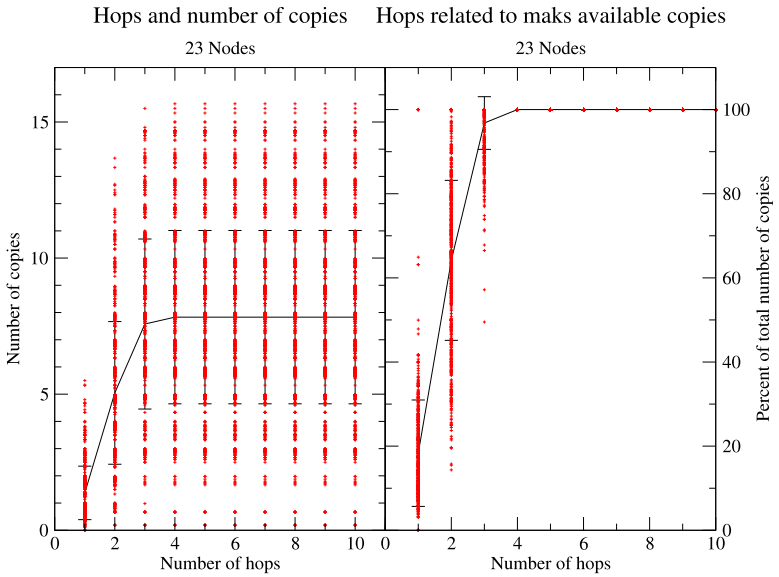

Figure B.6: Reachable files with 23 nodes

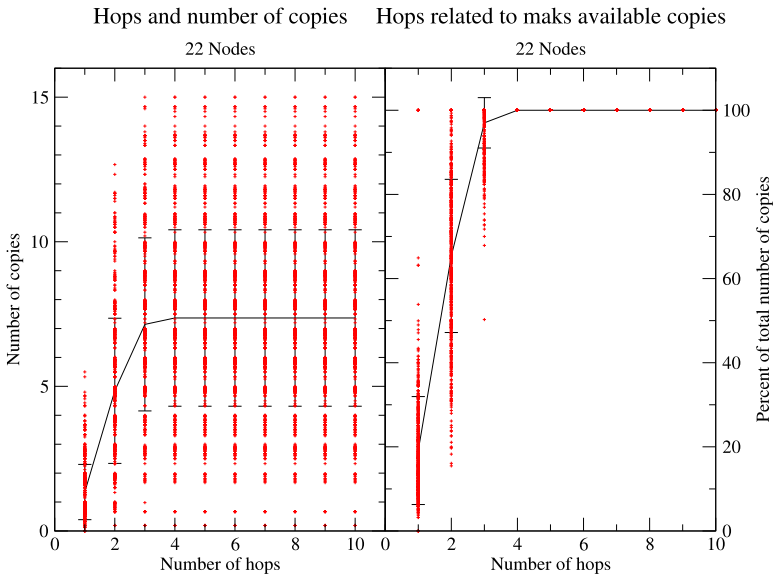

Figure B.7: Reachable files with 22 nodes

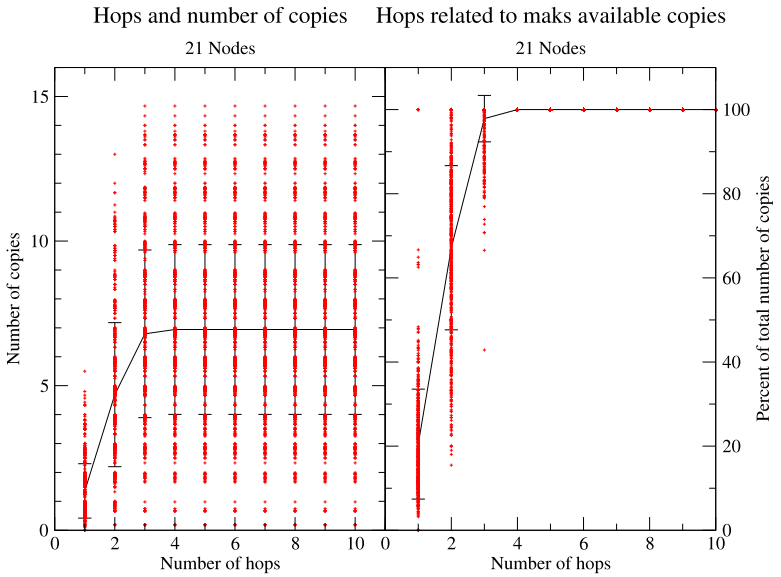

Figure B.8: Reachable files with 21 nodes

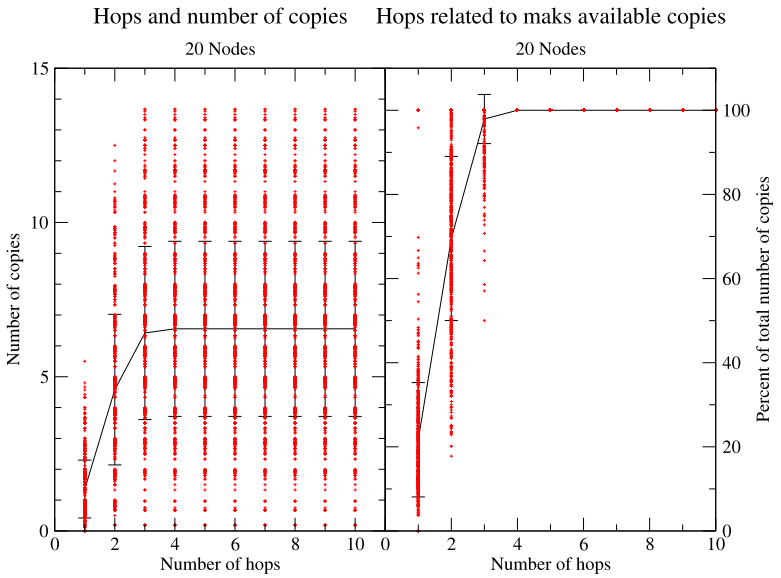

Figure B.9: Reachable files with 20 nodes

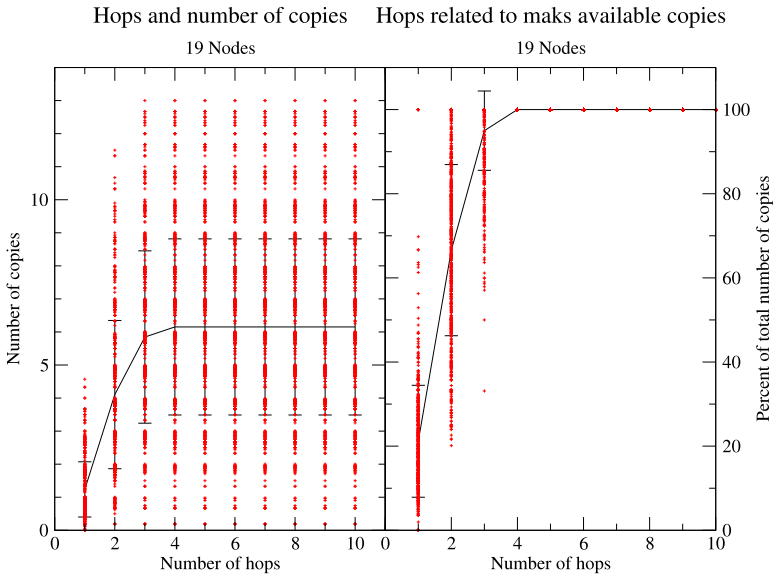

Figure B.10: Reachable files with 19 nodes

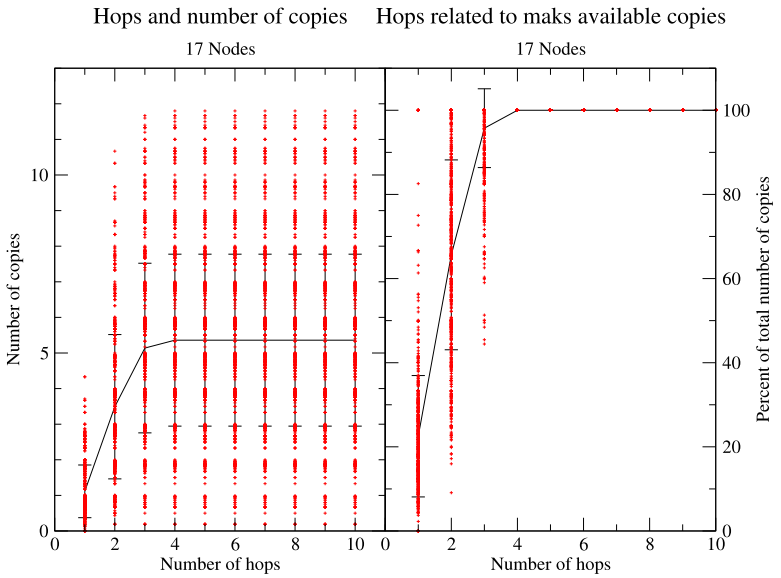

Figure B.11: Reachable files with 17 nodes

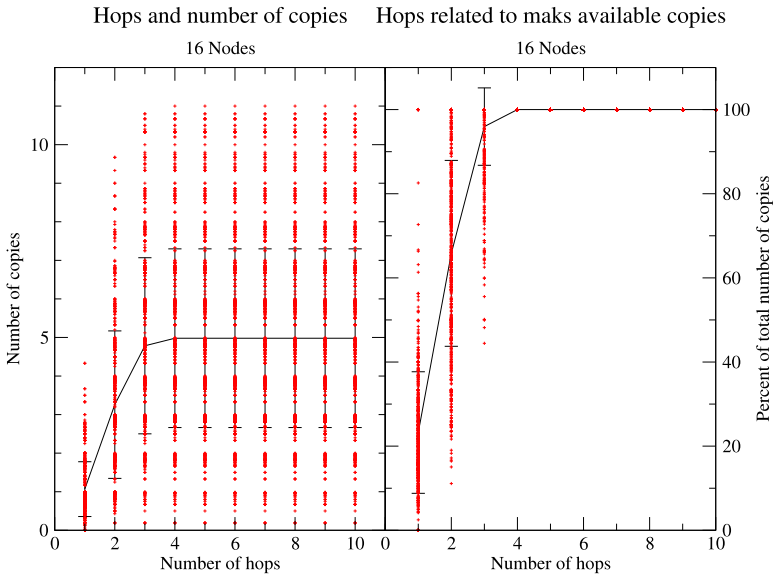

Figure B.12: Reachable files with 16 nodes

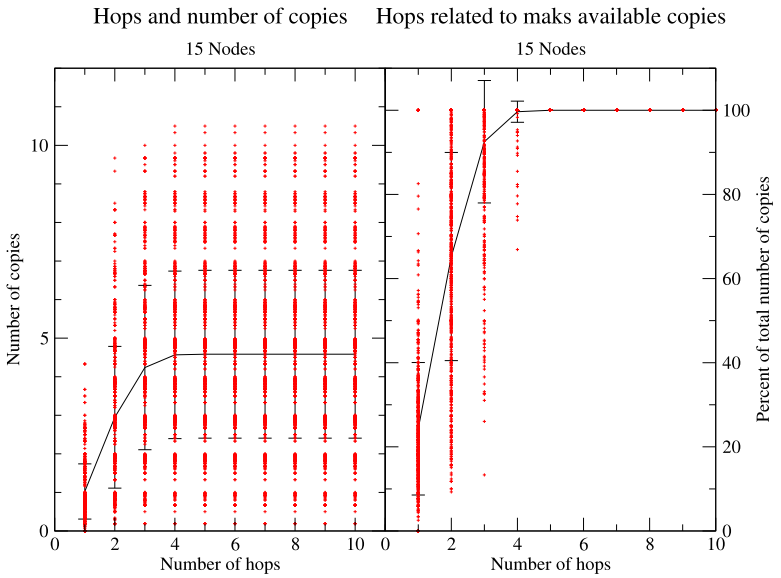

Figure B.13: Reachable files with 15 nodes

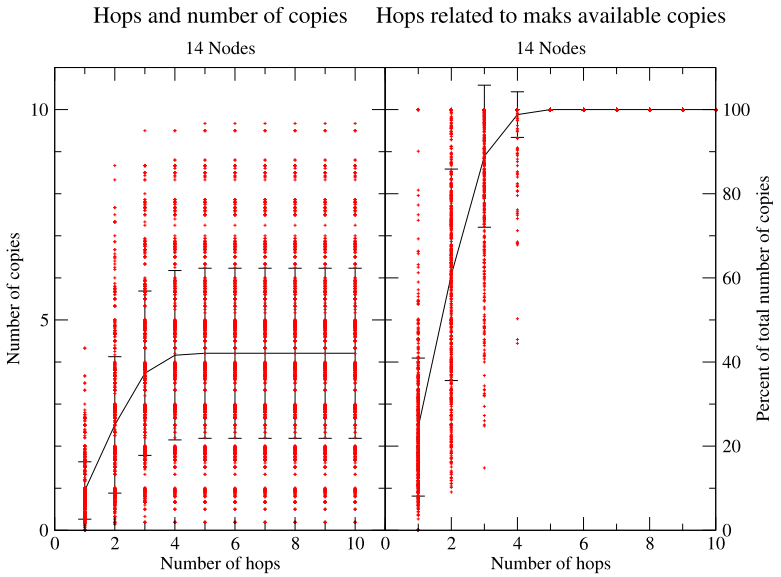

Figure B.14: Reachable files with 14 nodes

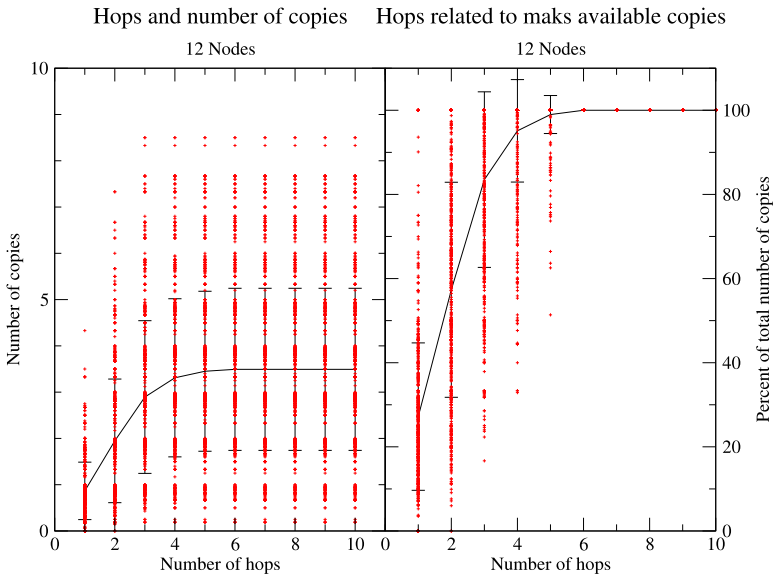

Figure B.15: Reachable files with 12 nodes

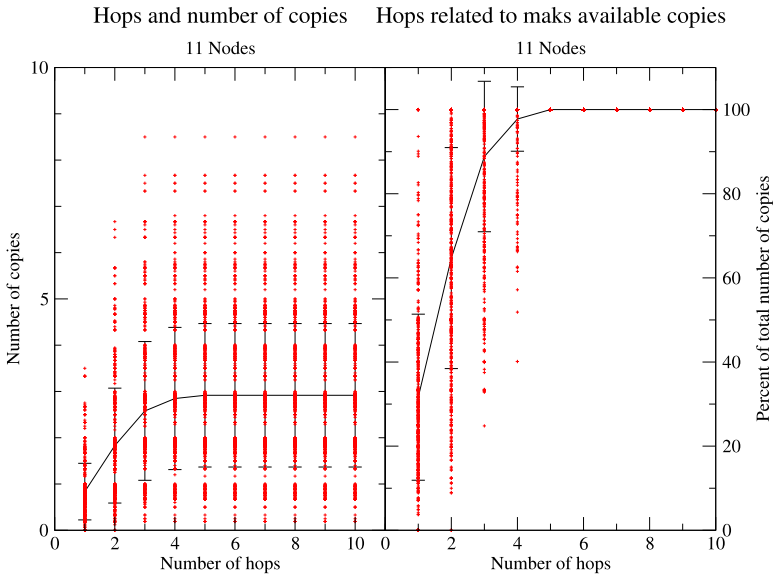

Figure B.16: Reachable files with 11 nodes

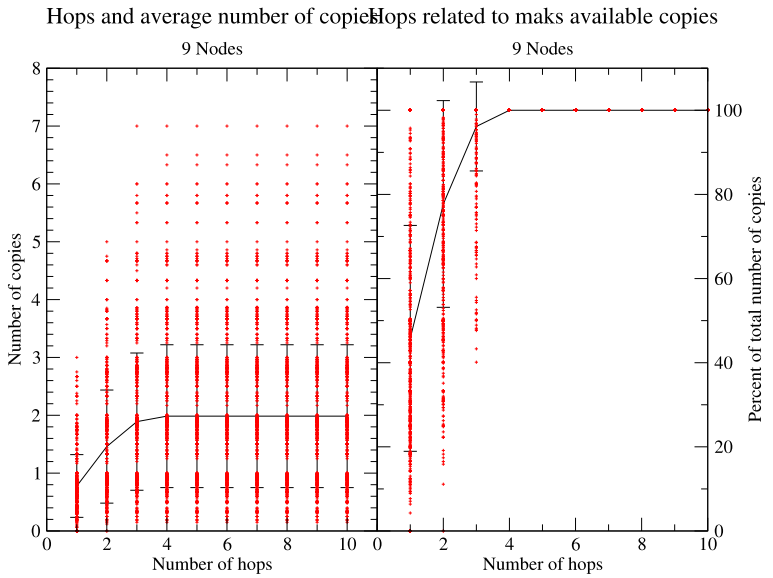

Figure B.17: Reachable files with 9 nodes

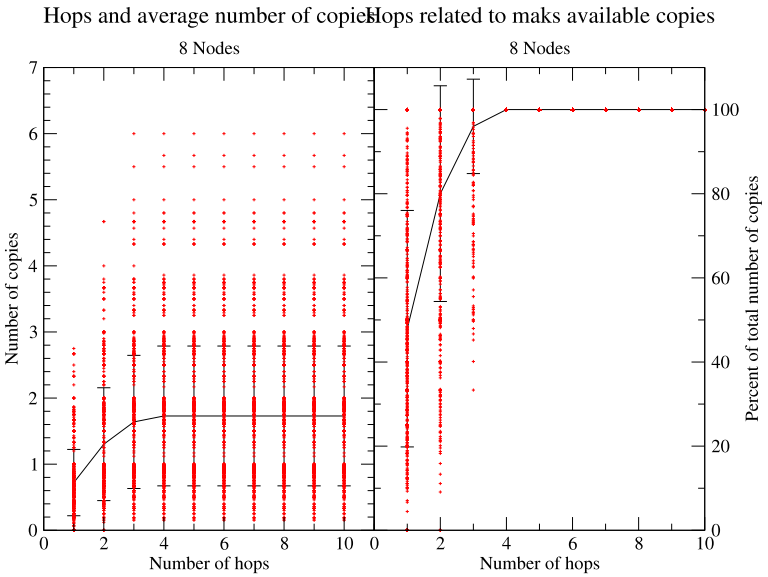

Figure B.18: Reachable files with 8 nodes

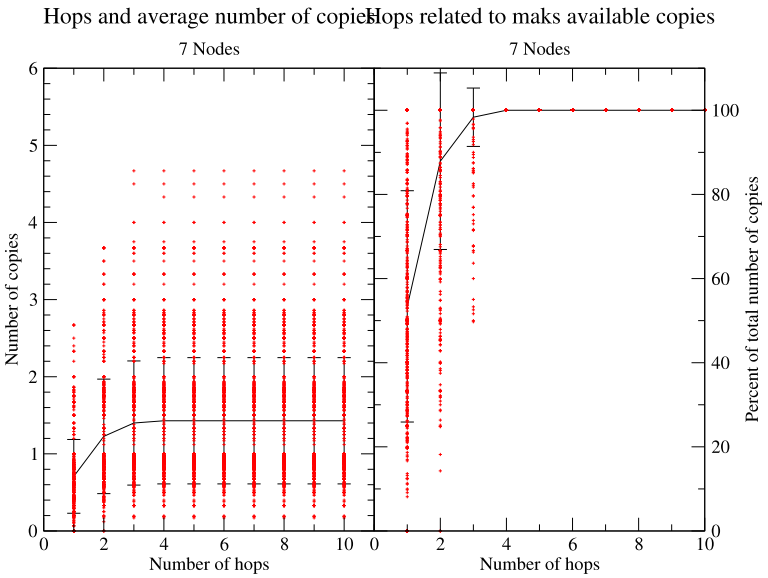

Figure B.19: Reachable files with 7 nodes

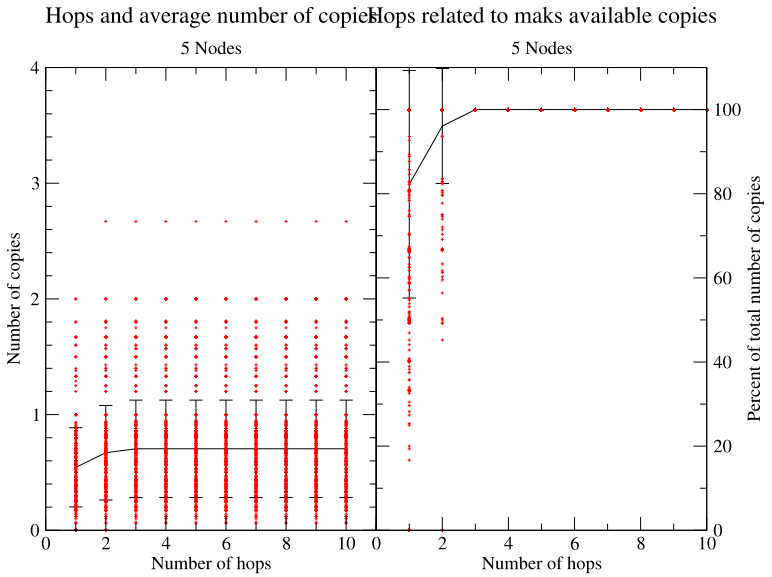

Figure B.20: Reachable files with 5 nodes

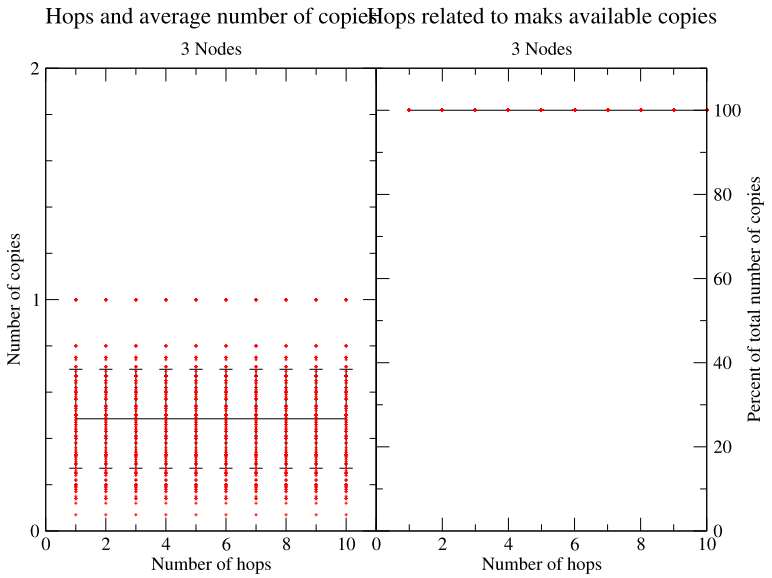

Figure B.21: Reachable files with 3 nodes

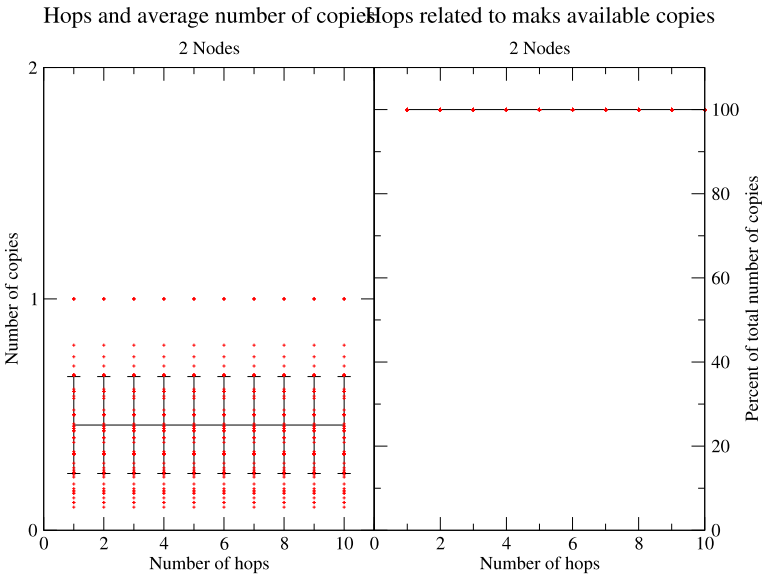

Figure B.22: Reachable files with 2 nodes

Bibliography

[BBST01] Christopher Batten, Kenneth Barr, Arvind Saraf, and Stanley Trepetin. pstore: A secure peer-to-peer backup system. *http://www.comp.nus.edu.sg/ zhanghan/paper/MIT-Pstore.pdf*, 2001.

[BDET00] William J. Bolosky, John R. Douceur, David Ely, and Marvin Theimer. Feasibility of a serverless distributed file system deployed on an existing set of desktop pcs. In *SIGMETRICS '00: Proceedings of the 2000 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pages 34–43, New York, NY, USA, 2000. ACM Press.

[Bon87] P. Bonacich. Power and centrality: a family of measures. *American Journal of Sociology*, 92:1170–1182, 1987.

[BS05] Kyrre Begnum and John Sechrest. Mln home page. *http://mln.sourceforge.net/*, February 2005.

[bts05] Bit torrent specification. *http://wiki.theory.org/BitTorrentSpecification*, February 2005.

[Bur93] M. Burgess. Cfengine www site. *http://www.iu.hio.no/cfengine*, 1993.

[Bur04] M. Burgess. *Analytical Network and System Administration — Managing Human-Computer Systems*. J. Wiley & Sons, Chichester, 2004.

[CBC00] F. Cuomo, A. Baiocchi, and R. Cautelier. A mac protocol for a wireless lan based on ofdm-cdma. *Communications Magazine, IEEE*, 38(9):152–159, 2000. TY - JOUR.

[CMN02] Landon P. Cox, Christopher D. Murray, and Brian D. Noble. Pastiche: making backup cheap and easy. *SIGOPS Oper. Syst. Rev.*, 36(SI):285–298, 2002.

[CVK98] Ann L. Chervenak, Vivekanand Vellanki, and Zachary Kurmas. Protecting file systems: A survey of backup techniques. In *Proceedings Joint NASA and IEEE Mass Storage Conference*, March 1998.

[DBF91] Y. Deswarte, L. Blain, and J.-C. Fabre. Intrusion tolerance in distributed computing systems. In *Research in Security and Privacy, 1991. Proceedings., 1991 IEEE Computer Society Symposium on*, pages 110–121, 1991. TY - CONF.

[Dik01a] Jeff Dike. User-mode linux. In *OLS 2001*, 2001.

[Dik01b] Jeff Dike. User-mode linux. In *5th Annual Linux Showcase & Conference 2001*, 2001.

[Dik02a] Jeff Dike. Blurring boundaries with user-mode linux. In *LCA 2002*, February 2002.

[Dik02b] Jeff Dike. User-mode linux. In *West Virginia University 2002*, 2002.

[Dik02c] Jeff Dike. User-mode linux and security. In *New England NetSec Professional's Roundtable 2002*, 2002.

[DR01] Peter Druschel and Antony Rowstron. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. *Lecture Notes in Computer Science*, 2218:329–350, November 2001.

[ea00] Brian Pawlowski et al. The nfs version 4 protocol. *Proceedings of the 2nd international system administration and networking conference (SANE2000)*, page 94, 2000.

[ELBZ02] Sameh Elnikety, Mark Lillibridge, Mike Burrows, and Willy Zwaenepoel. Cooperative backup system. In *FAST '02 - Conference on File and Storage Technologies*. Usenix Association, 2002.

[Fee99] Laura Marie Feeney. A taxonomy for routing protocols in mobile ad hoc networks. Technical Report T1999:07, SICS – Swedish Institute of Computer Science, October 1999.

[gnu05] Gnutella home page. *http://www.gnutella.com/*, February 2005.

[HK03] Taher H. Haveliwala and Sepandar D. Kamvar. The second eigenvalue of the google matrix. In *Stanford University Technical Report*, 2003.

[JMC+01] P. Jacquet, P. Muhlethaler, T. Clausen, A. Laouiti, A. Qayyum, and L. Viennot. Optimized link state routing protocol for ad hoc networks. In *Multi Topic Conference, 2001. IEEE INMIC 2001. Technology for the 21st Century. Proceedings. IEEE International*, pages 62–68, 2001. TY - CONF.

[KPB+04] Marc-Olivier Killijian, David Powell, Michel Bantre, Paul Couderc, and Yves Roudier. Collaborative backup for dependable mobile applications. In *Proceedings of the 2nd workshop on Middleware for pervasive and ad-hoc computing*, pages 146–149. ACM Press, 2004.

[KV98] Young-Bae Ko and Nitin H. Vaidya. Location-aided routing (lar) in mobile ad hoc networks. In *MobiCom '98: Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking*, pages 66–75, New York, NY, USA, 1998. ACM Press.

[Man94] Udi Manber. Finding similar files in a large file system. In *In proceedings of the USENIX Winter 1994 Technical Conference*, pages 1–10. Usenix Association, January 1994.

[MCM01] Athicha Muthitacharoen, Benjie Chen, and David Mazires. A low-bandwidth network file system. In *SOSP '01: Proceedings of the eighteenth ACM symposium on Operating systems principles*, pages 174–187. ACM Press, 2001.

[OW93] D.S. Ornstein and B. Weiss. Entropy and data compression schemes. *Information Theory, IEEE Transactions on*, 39(1):78–83, 1993. TY - JOUR.

[Per99] Charles E. Perkins. Mobile networking in the internet. *Mob. Netw. Appl.*, 3(4):319–334, 1999.

[PJS+94] Brian Pawlowski, Chet Juszczak, Peter Staubach, Carl Smith, Diane Lebel, and Dave Hitz. NFS version 3: Design and implementation. In *USENIX Summer*, pages 137–152, 1994.

[PR99] C.E. Perkins and E.M. Royer. Ad-hoc on-demand distance vector routing. In *Mobile Computing Systems and Applications, 1999. Proceedings. WMCSA '99. Second IEEE Workshop on*, pages 90–100, 1999. TY - CONF.

[SMLN+03] Ion Stoica, Robert Morris, David Liben-Nowell, David R. Karger, M. Frans Kaashoek, Frank Dabek, and Hari Balakrishnan. Chord: a scalable peer-to-peer lookup protocol for internet applications. *IEEE/ACM Trans. Netw.*, 11(1):17–32, 2003.

[uml05a] The user-mode linux kernel home page. *http://user-mode-linux.sourceforge.net/*, February 2005.

[uml05b] The user-mode linux on ppc. *http://www.nosreme.org/projects/umlppc/*, February 2005.

[Ver00] S. Verdu. Wireless bandwidth in the making. *Communications Magazine, IEEE*, 38(7):53–58, 2000. TY - JOUR.