FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

ASSIGNMENT OF BACHELOR'S THESIS

Title:	Storage Network Command Simulator - server and storage part
Student:	Dmitrii Vekshin
Supervisor:	Ing. Jiří Kašpar
Study Programme:	Informatics
Study Branch:	Computer Security and Information technology
Department:	Department of Computer Systems
Validity:	Until the end of summer semester 2019/20

Instructions

Research command line interface of a typical Fibre Channel (FC) based storage and Linux commands related to a storage connection.Design and implement server and storage part of the storage network command simulator.
Use the CDU-CLI framework for command line parsing and UDP message passing for communication with the FC switch part of the simulator.
Simulate the data operations by a direct filesystem link between the mount point and a directory representing a LUN on the storage side.

References

Will be provided by the supervisor.

prof. Ing. Pavel Tvrdík, CSc.
Head of Department

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
Dean

Prague January 5, 2019

FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Bachelor's thesis

Storage Network Command Simulator – server and storage part

Dmitrii Vekshin

Katedra počítačových systémů
Supervisor: Ing. Jiří Kašpar

May 17, 2019

Acknowledgements

I would like to express gratitude to my supervisor Ing. Jiří Kašpar for his help, advises and opportunity to work on very project. Also I would like to thank to my family and friends.

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended. In accordance with Article 46(6) of the Act, I hereby grant a nonexclusive authorization (license) to utilize this thesis, including any and all computer programs incorporated therein or attached thereto and all corresponding documentation (hereinafter collectively referred to as the "Work"), to any and all persons that wish to utilize the Work. Such persons are entitled to use the Work in any way (including for-profit purposes) that does not detract from its value. This authorization is not limited in terms of time, location and quantity. However, all persons that makes use of the above license shall be obliged to grant a license at least in the same scope as defined above with respect to each and every work that is created (wholly or in part) based on the Work, by modifying the Work, by combining the Work with another work, by including the Work in a collection of works or by adapting the Work (including translation), and at the same time make available the source code of such work at least in a way and scope that are comparable to the way and scope in which the source code of the Work is made available.

In Prague on May 17, 2019 .....

Czech Technical University in Prague
Faculty of Information Technology

Citation of this thesis

Vekshin, Dmitrii. *Storage Network Command Simulator – server and storage part.* Bachelor's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2019.

Abstrakt

Primárním cílem této bakalářské práce je vytvoření storage a serverové části v rámci projektu Simulace konfigurace storage sítě. V první části práce je provedena analýza vybrané množiny datových sítí, datových úložišť, jejich příkazových rozhraní a simulátorů daných technologií. Další částí je návrh, implementace a testování simulátorů úložiště a serverů s podporou modelování datových operací.

Klíčová slova datová síť, Fibre Channel, úložiště, World Wide Name, příkazové rozhraní

Abstract

The primary goal of this bachelor thesis is to create storage and server parts within the project Storage Network Command Simulator. The first part of the thesis presents an analysis of existing storage networks, data storages, their command interfaces and data storages simulators. The next part is the development, implementation and testing simulators of storage and server components, with supporting data operations modeling.

Keywords Storage Area Network, Fibre Channel, storage, World Wide Name, command line interface

Contents

List of Figures

List of Tables

Introduction

At the beginning of a programmer's career, a situation may arise when a practical experience is as important as theoretical knowledge. This situation is a typical case in a data storage management industry where graduate students do not have enough experience in a technological field to be employed and fulfill the requirement of the company.

Today the faculty of Information Technology provides excellent textbooks, high-class lectures, and has professional teachers for students who want to gain knowledge about the world of Information Technology. Unfortunately, not in all cases faculty equipment can provide practical skills that would be sufficient in the usage of some technologies in this infrastructure. The example of one of these technologies is a Storage Area Network (the term shall from hereon be referred to as the "SAN"). Today, SAN is a relevant topic in modern Information Infrastructure. SAN is a relevant topic because SAN is intended to process a high volume of information, which constantly growing and has to be saved and processed.
Another reason is that the help of SAN the requirement of segregation of the storage space between servers and the requirements of high performance, reliability, and support-ability will be fulfilled.

The project, in which I was lucky enough to participate, is aimed to design and create a system of modeling the Storage Area Network Environment, that consists of host servers, switches, storages and their interaction.
The goal of this project is to provide opportunities for monitoring process, configuring network and managing overall data network. The project is dedicated to students of the Faculty of Information Technology of the Czech Technical University in Prague and give opportunities to get knowledge and skills, that can be applied in actual work with data networking. The implementation of this system promotes an expansion of a content of The Storage and File-systems (code:BI-STO).

The thesis consists of five chapters, each of them describes the phase of the project.
The first chapter presents the project objectives, overall information about The Storage Area Network and lists of existing SAN storages and SAN simulators.
Next chapter describes the project structure, enumeration of simulated technologies and explanation of the tools. The terms are also explained here.
The third chapter shows the approach used in the stage design.
The Process of development is demonstrated in the fourth chapter.
The last chapter is about testing the use cases. The chapter contains the examples of usage and functions.

Chapter 1

Thesis description

The chapter describes the goal of the project. It also introduces the basic theory and issues in a data network and provides analysis of state-of-the-art data network technology.

1.1 Thesis' goal

- Research command line interface of a typical Fibre Channel (FC) based storage and Linux commands related to a storage connection. Design and implement server and storage part of the storage network command simulator.
- Use the CDU-CLI framework for command line parsing and UDP message passing for communication with the FC switch part of the simulator.
- Simulate the data operations by a direct filesystem link between the mount point and a directory representing a LUN on the storage side.

1.2 Keyword and Shortcut definition

This section will be presented by different definitions of terms, that are used in this project.

- **Fibre channel** (hereinafter FC) is a high-speed network technology, that intended for block-level data transferring, transfer of the data in a data-centers between servers, switches and storages. FC is a reliable technology, that prevents the loss of frames and it guarantees delivery of frames in the correct sequence. The standard technology description involves the use of optical fibre. [1]
- **Fibre Channel Protocol** is a transport protocol, that distributes Small Computer System Interface (hereinafter SCSI) commands in the

network. Before sending, the commands are encapsulated in Fibre Channel frames. The Protocol is designed for usage in FC network.[1]

- **Server-host** is a server with access to SAN and SAN storage.
- **Storage** is a data storage in SAN.
- **Fibre Channel switch** is a network device used in Fibre Channel networks. Fibre Channel switches are highly compatible with SANs. They provide high performance on the networks. In SAN, FC switches provide name server service. [2]
- **Fibre Channel Switch fabric** is the network topology, that consist of one or more switches. In switched fabric, all node ports are interconnected. [3]. Described topology in figure 1.1.

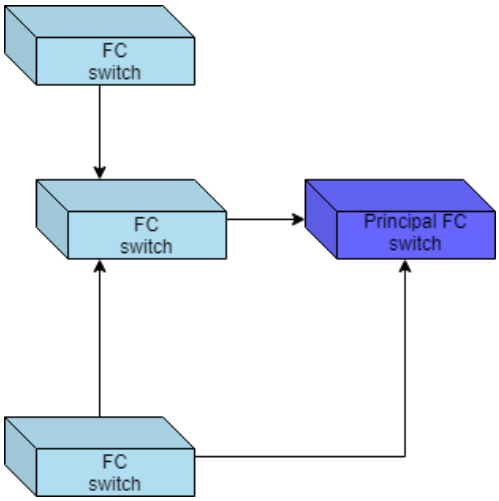

Figure 1.1: Fibre channel switch factory

The switch provides a connection between the compute node and the storage node during a data transfer operation.[4]

1.3 Storage area network and environment

A storage network is a data network that provides servers with access to shared storage resources. Storage Area Network is designed to manage and store large amounts of data.
SAN consists of the storages, switches and server-hosts. SAN architecture is demonstrated in figure 1.2.

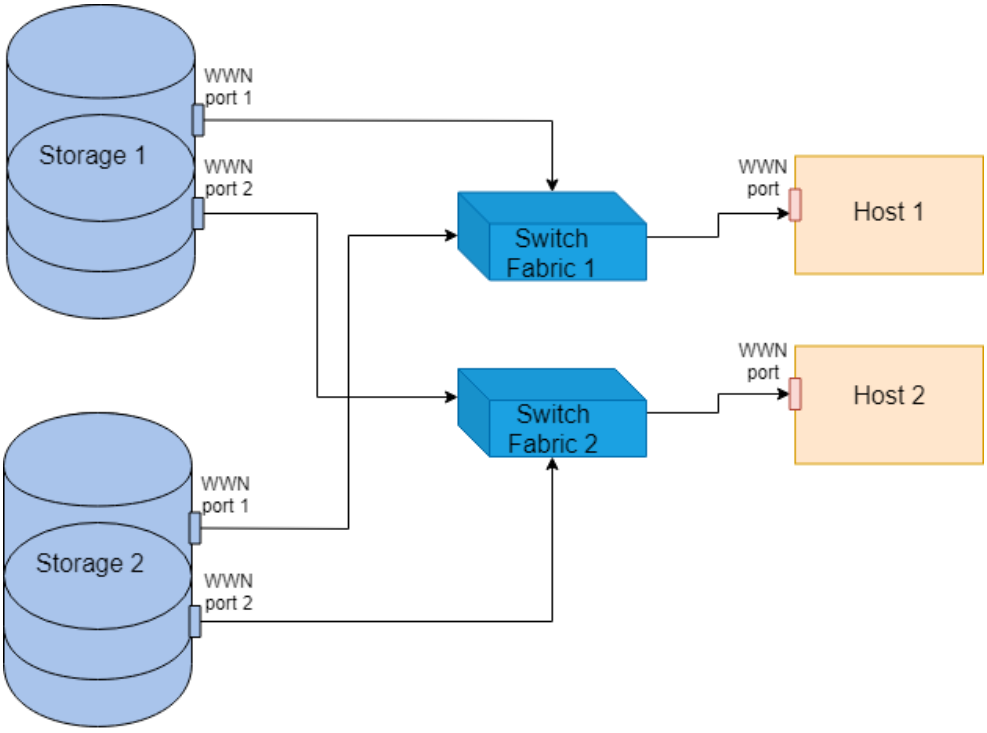

Figure 1.2: SAN architecture

In most cases, the Fibre Channel protocol is used to transfer data between SAN components. Also due to the protocol translation feature, SAN can include devices running on other protocols. That feature significantly increases system's flexibility. SAN topology is based on switched-architecture. [4]

It is worth noting how the presented storage volumes are presented to the operating system. The OS sees the volumes as directly related.
SAN is an isolated network, which means that SAN prevents access to resources for external network devices.

1.4 Execution in the real world

SAN is a great solution for the enterprise environment. The main advantages of using SAN in the enterprise environment are savings in operating costs, simple administration, reliability, and fault tolerance.

Furthermore, by using SAN it might bring the increase of:

- availability of business applications

- speed of backup
- centralization of monitoring processes

The price and complexity of using SAN dropped significantly at the beginning of the century. This made the technology available for deployment in the small and medium-sized enterprises.[5]

Today there is a wide range of SAN components on the data network market but there are no offers of ready-made solutions. The reason for this is that the approach to SAN implementation for each case must be unique and taken into account the specific order conditions. This caused by diversity of topology, different number of servers, storages and switches in the system, as well as different requirements of capacity, speed of data transmission, security level, and data protection.

Some major suppliers (Lenovo, Dell and QSAN) in online stores offer customers to set up an online SAN order - choose switches, storage, and hosts. This process is performed in such a way that the provider determines the types of components that are required in the SAN structure, and then the client selects the devices of each type. 1.3

1.4.1 SAN storages

The following analysis of the existence of SAN storages provide knowledge, that needed for modeling a storage part of the simulator.

Lenovo ThinkSystem DE2000H.
The supplier declares that the following product characteristics:
The ThinkSystem DE2000H (hereinafter Lenovo DE2000H) is low-cost and efficient, secure, and has a large capacity. It works great and has a high workload conditionals, which guarantees suitability for corporate needs. And also have the ability to solve the problem, assuming intense I / O operations such as Big-Date, data analysis, camera systems, backups. [7]

QSAN XCubeSAN XS1200 Series.
The supplier considers that the system effective and effective. The XCubeSAN XS1200 (QSAN XS1200) system provides data services such as data security, data protection, accessibility, scalability, and high consolidation. QSAN XS1200 is a storage designed for critical significant applications. [8]

Dell EMC PowerVault ME4.
Dell EMC PowerVault ME4. The supplier defines the product as: affordable, simple and fast, designed to manage the needs of physical and virtual applications. PowerVault ME4 (Dell ME4 only) is suitable for small businesses. It

Figure 1.3: Choosing SAN components

[6]

takes care of storage block consolidation requirements, intensive I / O operations, virtualization optimization, and data management. [9]

Systems are supported following host's OS,

Dell ME4:
Windows 2016,2012 R2
RHEL 6.9 and 7.4
SLES 12.3

VMware 6.7, 6.5 and 6.0

QSAN XS1200:
Windows 2008(R2),2012(R2),2016
SLES(SUSE Server)10,11,12
RHEL(Red Hat)5,6,7
CentOS 6,7
Solaris 10, 11
Free BSD 9, 10
Mac OS X 10.11 or later

Lenovo DE2000H:
Windows 2012 R2,2016
RHEL 6, 7
SLES(SUSE Server)11,12,15
vSphere 5.5, 6.0, 6.5, 6.7.

Comparing of products functionality is demonstrated in table 1.1.

Table 1.1: Storage features comparing

option	Dell ME4	QSAN XS1200	Lenovo DE2000H
Controllers	2	2	2
Controller cache	8GB	up to 32GB	8GB
Max drives	336	626	96
Max raw capacity	4PB	2.6PB	1.47PB
Host Connectivity	37	34	12
Host type	FC,iSCSI,SAS	FC,iSCSI,SAS	FC,iSCSI,SAS
RAIDs support	0,1,5,6,10,50	0,1,5,6,10,50	0,1,3,5,6,10
Data encryption	SED,FCE	SED,ISE	SED

Then the SAN management software was investigated. SAN control programs are not an open source. But the YouTube channel "Lenovo Data Center" demonstrates the providing of Lenovo customer support. The channel contents video tutorials. Tutorials demonstrate management of Lenovo S Series array using the Lenovo Storage Management console.
This course shows the basic software components and functions. Part of the demo of the workstation with the Lenovo Storage S3200, which mentions topics such as creating and presenting to hosts logical volumes (LVOL) also system configuration. The videos provide visualization of command syntax, structure and results of execution.

Another survey included two command interfaces - the IBM SAN Volume Controller and the HP Storage System Scripting Utility (hereinafter SSSU). Then SSSU was selected as CLI design model in the project for reasons of logical ordering and syntax clarity. Most suitability of the SSSU in the project is also ensured by the fact that the tool is provided by a supplier of another tool already selected for use in the project. The SSSU functionality is presented by figure 1.4. Most commands are presented by figure come with a variety of flags that are setting up command target or execution.

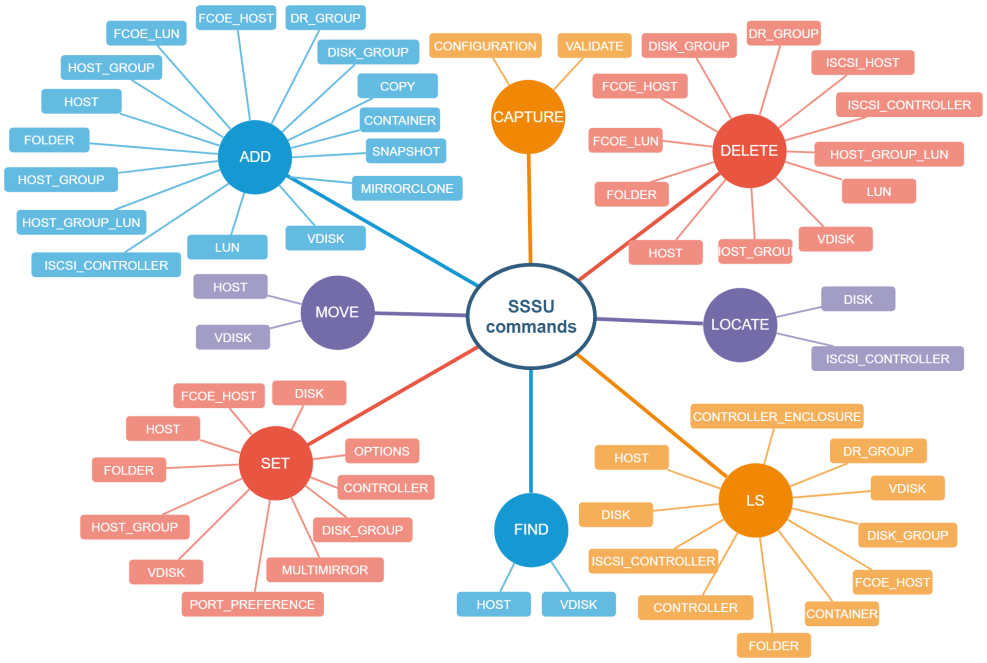

Figure 1.4: SSSU commands

[10]

Knowledge based on the analysis of functionality, architecture, graphical and

command interface structure of real software and hardware products are using on the design of components and commands interface in the planning phase of the project.

1.5 Existing storage area network simulators

Only SimSan simulator was found at the time of examining existing SAN simulators. In addition, storage simulators are analyzed to gain knowledge to form the right design approach.

1.5.1 SimSan

The program models the network behavior when running data transferring operations and shows the process of moving the SCSI command over FC/ FCoE. SimSan in detail demonstrates structure of frames in the network. The simulator allows users to test the role of the network administrator and get acquaintance with that role daily tickets. Examples of such tasks are work with logical unit numbers, port configurations, and name server queries.

The weak side of SimSan is not carrying out a simulation of working with volumes. This is the most important part of the SAN feature.

1.5.2 HP EVA P6000

It is a storage simulator with a graphical interface that very accurately simulates the behavior of product from HP Eva group. The program supports simulation work with hosts over Internet Small Computer System Interface (hereinafter iSCSI) and FCoE.
Simulator's command interface consists of two commands groups:

The first group is called "Provision" and contains a list of commands for simulating volume management operations.

- Create a logical volume group
- Add host
- Create logical volume
- Map logical volume to host

The second group is called Data Protection:

- Replication

- Snapshots
- Clone

An interesting feature of the system is the ability to set snapshot rules for each individual LVOL.

1.5.3 IBM System StorageDS Storage Manager

The program can simulate the behavior of several storages in data centre. Each of such storage is represented as SubSystem.
After installation, the program creates the default SubSystems (the description is presented in the table. 1.2

Table 1.2: List of simulated by IBM storages

Model	Controllers	Storage Array	Arrays count	Drives
DS5300	1	EXP5000	28	FDE, FC, SATA
DS5300	1	EXP5000	20	FDE, FC, SATA
DS5100	1	EXP5000	10	FC, SATA
DS4800	1	EXP810	8	FC, STA
DS4700	1	EXP810	6	FC, SATA

Each of that subsystems responds to a actual IBM product. Program simulates the creation of logical volumes, provision LVOLs to hosts and other volume management operations. In addition, the application simulates raid management and ports configuration. The graphical interface of the program is very clear and well demonstrates the structure of disk arrays.

1.5.4 HPE 3PAR StoreServ Simulator

The application imitates the behavior of the HPE 3PAR StoreServ Storage System. Software provides simulation of maintenance, administration and reporting processes.

Supported functionality and features:

- Disk array up to 48 disks HDDs
- Disk interfaces FC, SSD, NL
- 3PAR Management Console
- CLI and Remote CLI support

- Storage Provisioning
- Exporting Virtual Volumes
- Local Replication (Snapshot)

HP 3PAR Simulator does not provide server host simulations. The lack of hosts in the program makes it impossible to monitor the impact of data management operations.
However, the program allows to create a Fake-Host. Fake-Host has no properties, except the presence of WWID and Host-name.

Chapter 2

Analysis

The chapter describes the structure of the project. Moreover, It presents the tools used in the project. Information in this chapter is served as a starting point for further design.

2.1 Project architecture

The project architecture consists of four components:

- Storage
- Storage network
- Control console simulation
- Servers

Figure 2.1 shows the structure of the project.

The controlling component is Control Console Simulation FCSIM. FCSIM controls the start, pause, and the end of the simulation. Furthermore, FCSIM registers all transactions in the system.

Storage network is a network of switching factories that provides communication between the hosts and the storages. The component of the network performs zoning.

Storage is a data storage. The primary responsibility of storage is to provide storage space to the clients. For this reason, the storage also provides the transformation of physical space into logical storage space, splitting this logical space into logical volumes, setting up access control and mounting these volumes to hosts.

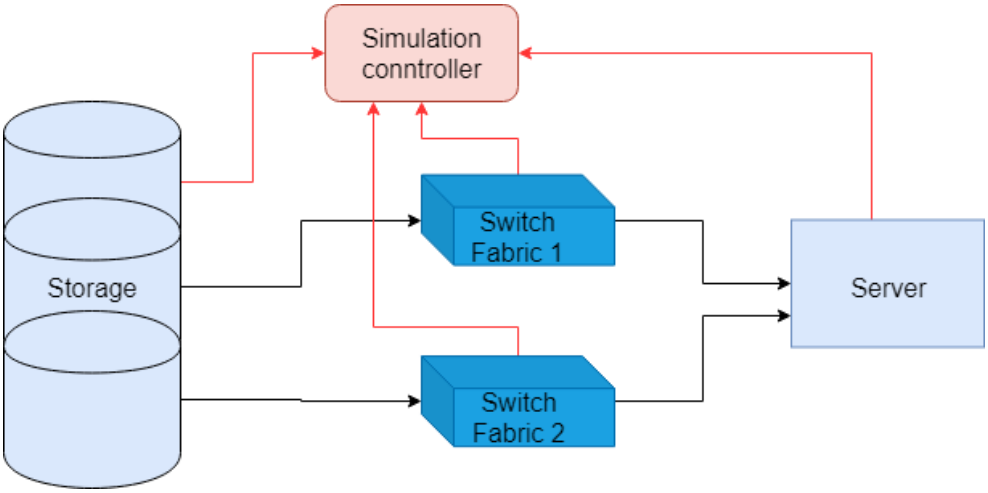

Figure 2.1: Project architecture

Server represents server-host connected to SAN. Host is a storage client, that consume storage resources. Host also performs management operations with the provided storage space, examples of such operations are representing as a local disk, then mounting file-systems and then write and riding from disk.

2.2 Simulated technologies

All technologies introduced in this section are the core of the functionality of the simulator.

2.2.1 Storage side

Logical Unit Number - a logical entity, that represents a logical storage space converted on storage side from a physical disk space. Such representation is accessible and usable to hosts.

Mapping volumes - the operation provides LVOL assignment for the host LUN. The operation makes LVOL visible to the host.

Mounting volumes - The operation mounts the logical volume to the host. After the process is complete, the host can use a LVOL, and the host OS identifies the LVOL as a local disk.

Pool - logical space that is created by aggregate capacity from different physical sources.

Zoning - a service that allows to set rules for access to storage resources for each host.

World Wide Name - eight-byte unique device identifier. In Fibre Channel networks, WWN is the equivalent of MAC address.[1]

RAID - A redundant array of independent disks is a physical disk array management technology that improves performance or reliability, or both.[4]

Logical volume - is disk-based logical address space.

Physical Volume - (hereinafter PVOL) physical disk.

Physical Volume Group - (hereinafter PVG) is a set of physical disks. Mostly, disks in the group have similar properties, or placement, or both.

2.2.2 Server side

Multipath - technique multipath allows to define several physical paths between server and storage. The use of the multipath provides dynamic load sharing and automatically emergency paths changing.

File-system management - the process of creating and mounting a file system on an unallocated disk.

2.3 Tools and technologies used in project

The section presents several external components of the project. The techniques shown in the section are used in the development of the project.

2.3.1 Framework ttdrv

The framework simulates the command interface, loads, and then processes the text from the standard input.
The utility sends and receives sockets through the UDP port. The resulting sockets are processed by the framework, which includes the launch of the subroutine. The choice of a specific procedure occurs in accordance with the contents of the socket. The maximum number of predefined procedures is 256. The first byte of the socket header is the encoded name of the communication protocol. For each type of communication protocol, the infrastructure allows you to temporarily register a routine. After the timeout, registration will be canceled, provided that the other protocol socket has not arrived.
The second byte of the socket is used in the request-response services. This type of communication is defined in such a way that it is required the request to contain a letter in the second byte in a large register (A-Z), while the answer

must contain the same character in a small register (in addition, the request and the Answer must be on the same protocol).
Each supported service has a unique letter for socket headers.

2.3.2 Command Definition Utility

The Command Definition Unit is used to define the command syntax.
The CDU also generates sources for use in parsing and validating incoming commands. In addition, the framework prepares the methods of collecting data in incoming commands, and then presents them to routines. Another feature of the CDU is the preparation of routines templates.

CDU ensures that the unification of the command line interfaces for project components.

Working with CDU involves writing files with command definitions in the command definition language, after which files with command definitions are processed by the CDU. The output of this operation is four other files:

- unit_table.c
- unit_run.c
- unit_cli..c
- unit_cli.h

Unit Table contents information about the syntax of the command. The file is used in parsing and checking incoming commands.

Unit_ Run file presents headers of routines. Routines are used for processing incoming commands.

Files UNIT_CLI provide transferring data of incoming commands to routines. For each possible value of parameter or qualifier of incoming command, file UNIT_CLI.h contents three EXTERN variables:
i

- present
- value
- result

Present demonstrates the presence of parameter or qualifier value. This is a useful feature in case of an optional value.
Value contents parameter or qualifier value.
File UNIT_CLI.C provide setting up of that variables and then call of suitable routine. Figure 2.2 shows the CDU use.

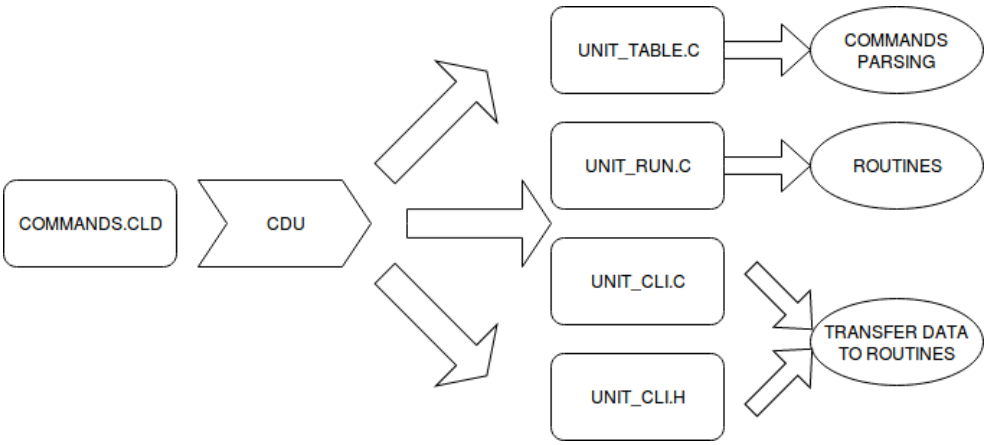

Figure 2.2: CDU use

2.3.3 Commands line definition

Creation of the command definitions requires the writing command definition files that describe commands. Command definitions consist of several parts: verbs, parameters, qualifiers, keywords, and routines.

command definition file - a command-defining file which contains a record of an image or routine that is called by a given command

Verb - specification of command to the following processing. On figure 1.4, Verbs are presented in circles adjacent to the central one (called SSSU commands).

Parameter - command object specification. For a parameter, it is possible to specify the type of its value.

Qualifier - description action performed by the command. The type of the qualifier can also be specified. Examples of such specifications are integer values, keywords, and character strings.

Keyword – a predefined string. Keywords can be used as a value of a parameter, qualifier, or another keyword.

2.3.4 UDP

User Datagram Protocol - protocol of transport layer (OSI model). USD provides a simple messaging service regardless of reliability, which means that It does not perform flow control and error correction functions. UDP is the

interface between IP and higher layer protocols. Also, the protocol requires a relatively small size of the message head, causing a lower network load. [11]

UDP is useful if the reliability of the transport protocol is negligible. For example, in systems where errors and flow are checked in higher layers. As in our case, where the control and parsing of the front commands are performed by the application.

2.3.5 Links

Link - is an object in a file system that is not a real file or folder, but link is a reference to folder or file.

Chapter 3

Design

This chapter describes in detail the design of the server and storage components as well as their functions and interaction.

3.1 Design foundations

Each component is represented by an independent process with own command prompt window and own command line interface, which makes the viewing of simulation processing simple and clear.

Communication between the system's components is provided through UDP protocol. Each component is able to interact with another independently.

Data operations will be provided directly in file-system.

Mapping volumes is simulated via creating of links in file-system.

Inserting and removing of physical disks is simulated by creating resp. removing files with information about disk.

The creation of resp. removing of logical volumes occurs in the system by creating the directories and removing them. For servers, this approach provides abilities to create their own file-systems on supplied memory sources.

3.2 Server

The Server component is a server host with a UNIX-like OS. In the project, the server has the role of initiator, the server begins to communicate by sending requests to the storage. The server provides the following functions:

- Preparing for communication by connection to SAN

- Port Configuration.
- Initiation of disk mounting.
- Mounting file-systems.
- Ensure simulation of multipath technique.
- Evidence of mapped and mounted disks.
- Showing information about mapped disks, disk drives, system and ports. The server and its ports are identified by a unique WWID.

3.3 Storage

Storage - a component that represents a data storage and manages storage resources, protects data, and also provides resources to clients.

The repository component models the elements that represent the physical aspects of the internal structure of the storage, such as PVOL, PDG, PDG and controllers. Also, the Storage represents the modeling of logical units, such as LVOL, pools over groups, RAID and LUN.

Architecture is designed in such a way, that the ultimate product is a simulated storage which includes two controllers. Each controller manages a certain set of physical disks.
Total Storage capacity consists of aggregated capacity of connected and registered PVOL. Adding of PVOL require to specify the type and size of a disk and to assign the disk to a controller.
PVOL can be combined in a PDG. It is possible to set the type of RAID for each of such a group. By creation of a group, there is a creation of a new pool. The pool's capacity consists of the sum of the capacities of the individual elements of the group and consumption of space, that is reserved for data protection (depending on the type of raid and the group). There are two default groups with RAID 0, this groups contents controllers disks. By default, controller has its own group. Pool is a source for LVOL, which is then assigned to a hosts LUNs. Each host has a set of LUNs At the moment LVOL can be assigned to several LUN of different hosts. After that assignment, LVOL becomes visible and available to mounting to host. But LVOL cannot be mounted to more than one host at a time, exception is a cluster systems.
LVOL supports functions of the writing protection, which makes the disk read-only. Also for each LVOL is possible to select the type of cache. The storage architecture is presented oin the figure 3.1.

The following is a list of simulated elements also their properties:

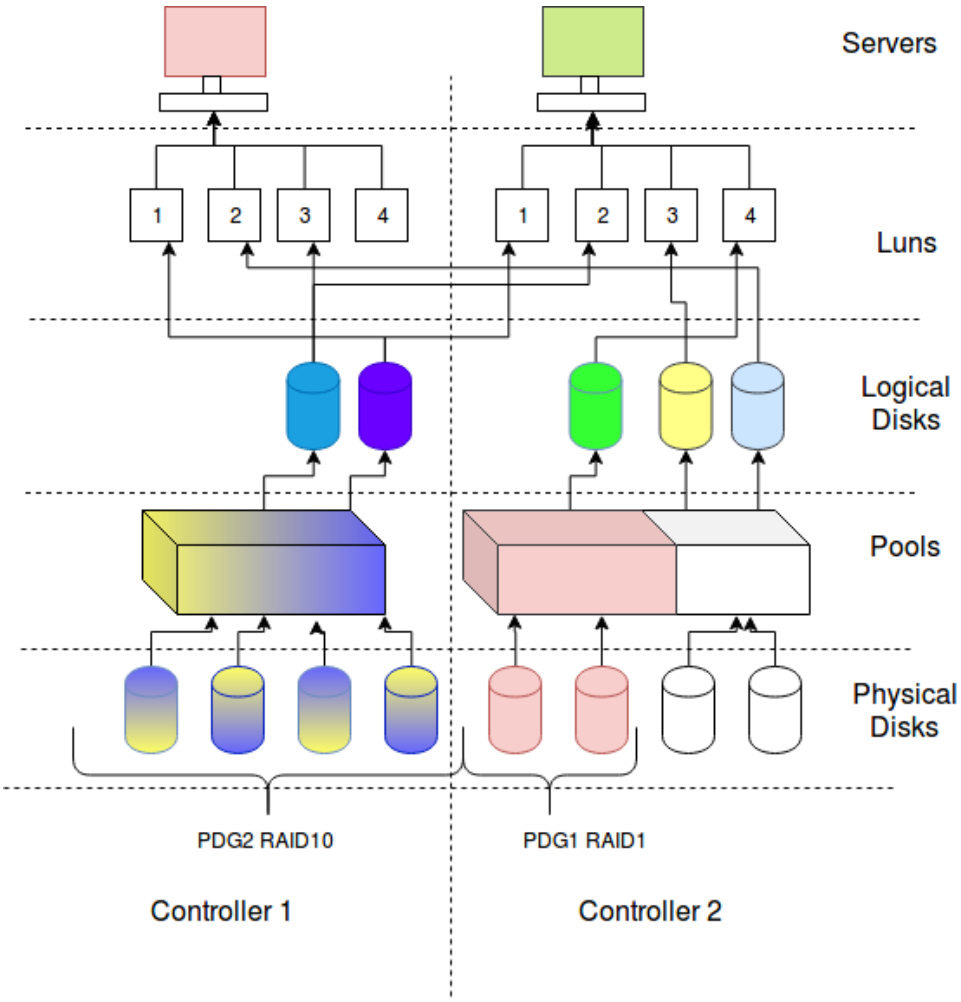

Figure 3.1: Storage architecture

- **Pool** - Source of logical memory, that is converted from storage, physical memory. Pool capacity is used to create logical volumes.
- **PVOL** - Source of physical memory. Each physical disk capacity extends the total storage capacity.
- **PDG** - a group of physical disks, performs RAID determination over disks containing a group.
- **Port** - Device FC port.
- **LVOL** - logical volume on storage.
- **LUN** - Logical unit number, provide presents of LVOL to host.

- **Mapping LVOL** The mapping relationship between LVOL and Host. The operation assigns LVOL to the LUN host, which provides the host access to LVOL in the Storage. After mapping, the host can request the Storage to mount LVOL.
- **List of host** - list of hosts connected to the storage, includes host data such as name, WWID, FC port number, connection type, and host compatibility with multipath. Also, there is a list of its ports for each host.

When creating a Pool, PVOL, LVOL or LUN, the system generates and then assigns an an identification number to the created device.

3.4 Command line interface

The design of the CLI components is based on the need to fulfill the requirement for the implementation of the Operations of Volume Management in the system. Another driving factor was the knowledge from the analysis of existing solutions. Since the Server and Storage perform different activities in the project, the CLI design consists of two phases.

3.4.1 Storage command line interface

Storage CLI allows to control the following features and functionality of the system:

- Creating a group of physical disks, assigning disks to PDG, setting up the type of PDG raids.
- Creating LVOL and defining its features, like size, type of cache, writing protection, alias and changing all of these parameters
- Adding of PVOL with parameters - size, interface, and selection of disk controller.
- Creating a PDG of physical disks, assigning disks to PDG, setting up the type of raid in PDG .
- Registration of hosts and its ports, assigning alias to hosts.
- Mapping LVOL to hosts with specification ports that are used for mapping.
- Detection of active devices in the network.

- Present to user information about elements of storage and relations with hosts.

3.4.2 Server command line interface

The CLI server allows the user to customize the server element that affects the simulation run and manage volumes storage space provided by the storage. The following operations are supported:

- Initiate mounting of an available volumes.
- Initiate mapping of volumes.
- Canceling of mapping relation.
- Creating and mounting of a file-system.
- Present to user information about host LUN on storage.
- Present to user information about server elements.
- Request information about available logical volumes on storage.
- Enabling or disabling of multipath technology.
- Control of mounted or mapped volume state.

3.5 Interaction of components

Communication between components is provided by sending UDP Datagram. To implement this type of communication, the program use functions: UPDset, UPDput and UDPsend_recv from io.h library, functions are presented in figure 3.2.

UPDput provides a sending UDP Datagram to target port, that specify by function parameters. UDPsend_recv ensures the sending and receiving of a UDP Datagram. The response Datagram is processed by the preset handler function. Also, the function requires to set a response timeout. UPDset provides a sending UDP Datagram handler function to certain UDP port. That mean all input UDP Datagram are processed by that handler function.

Control console simulation manages the life cycle of each particular component.
The beginning of the participation of each component in the simulation also implies to the beginning of listening on UDP port. Number of UDP port and

```
void UDPprot_set(IOhandle *this, Buffer *address, int port,
    int (* handler)(IOhandle *that, Buffer *data,
    Buffer *address, int port), uint8_t protocol, int timeout);

int UDPput(IOhandle *this, Buffer *data, Buffer *address,
    int port);

void UDPsend_recv(IOhandle *this, Buffer *data, Buffer *address,
    int port,   int (* handler)(IOhandle *that, Buffer *data,
    Buffer *address, int port), int timeout);
```

Figure 3.2: UDP_function

IP address are provided by Control console simulation.
There are two commands to start Server or Storage:

create(directory, IP address, UDP port, number of FC ports) - command serves for creation of a new component. That command creates the folder directory and opens the UDP port, which creates internal representation of a new component.

run(directory, IP address, UDP port) - command serves to restore existing component and provide the start of listening on UDP port.

After creating or running the components are available to listen to other commands from Control console simulation:

connect(local FC port, remote FC port, remote IP, remote UDP port) - command provide connecting of device FC port to switch FC port. Command model connecting fibre optic cable .

disconnect(local FC port) - disconnect port of fibre. If the command is received by the server without the multipath technologies or if after the disconnection there would be no path to the storage, that server provides unmounting of all disks.

destroy(directory) - deleting of components from program, command also remove folder directory of component.

start() - starting simulation.

exit() - exit of program. Servers provides unmounting of all disks.

stop() - stop simulation.

Registration of ports on switch is required to connect to the network and start

communication. For that reason the service FabricLogin was designed. Each registered component must send its WWID, IP address, UDP port and FC port number.

After registering ports, the server and storage update information about online device and their FC ports in the network. In order to provide this operation, a service called DirectoryServer exists, that returns a record of all FC ports involved in the network and belong to the set of zones of the requesting FC port.
For each device found, the response includes WWID of device, WWID of port, FC port, IP address and UDP port.

Server and storage handle service response differently.
Storage saves the response as a list of online devices.
Server requests to each FC port a list of available volumes. Operation is provided by the service VolumeInfo. The following step is to check the stored list of mounted disks and then provide mounting of all available volumes from that list.

All volume mapping and mounting processes are provided by storage,
the operation can begin as requested by the host or at the command of the storage.
If running a Volume mapping for a host is initiated by storage, the operation require entering the volume name and server name.
If the mapping is initiated by the server, request is handled by VolumeFactory service. Processing the request consists of creating a new LVOL in the storage, and assign that LVOL to the host LUN. In this case, the operation requires to specify the size of LVOL, name alias and type of disk cache.
When volume mapping is processed, storage updates list of mapped volumes, then run VolumeInfo service and synchronize information with storage.
Both the storage and the server can cancel mapping session. When canceled, the second side is notified, and after that both parties delete the relation record. Canceling does not take place if volume mounted to host.
Each server stores a list of volumes attached to it. When the server starts, the VolumeMount service mounts all the volumes in this list, then the contents of the list are updated.
Information on mounting and mapping volumes can be represented by the corresponding command, as well as information that the information is presented in the file system in the form of folders and their permissions.

The server may request to mount the volume that is assigned to its LUN. When storage receives a request to mount a volume for a server, storage check if the volume is occupied by another host. If the volume is available to mounted, a directory is created on the host with no read and write permissions. Directory represents mounted, unallocated disk. If successful, the server and

the repository record the relevant information about the changes.

3.6 WWID generator

The requirement to generate a WWID says that WWID for each device must be unique and that the length of WWID must match the actual length of WWID - 8-bytes. Algorithm djb2 was chosen for hashing In view of this requirement, the following principle of generating WWID is proposed:
Due to the guaranteed uniqueness of the names in the project, the storage or server WWID is generated as the 8 byte hash of device name. WWID in PVOL consists of four digits which represents the serial number of the added disk and WWID storage without first four digits. The WWID port consists of WWID storage or server where the last four digits are replaced by the port ordinal number given to be created into the system. The current ordinal numbers of the disk and port are kept in the system info on the storage or server.

chapter Implementation

This chapter presents the process of implementing the planned system features.

Chapter 4

Implementation

This chapter presents the process of implementing the planned system features.

4.1 Server

Simultaneously, several servers can participate in the simulation. Each server instance works in a separate terminal window, and also has its own starting directory. In addition, each server has its own directory tree.
The core of instance is a Server Data Object,that contains information about all temporary and permanent elements of the server. Server component is developed according to the Singleton pattern, its mean one instance cannot have more than one Server Data Object.

The following is a description of all server directories:

- **/dev** directory content two folders. /dev/dsk a folder contains mounted disks. Each object is a link to the corresponding LVOL.
  /dev/available - a list of available storage volumes. Each entry is represented as a link to the volume information file in the repository.
- **port** a folder containing files representing FC ports.
- **sys** - directory content a system information.
- **var** - directory provides information of mounted disks of storage. Each disk represented as a file, that contents storage of a disk information.

Each available volume is represented by a link to the file in the storage, that Each mounted disk is represented as a link to the volume folder in the storage.
Mounted disks without a file system do not have read and write permissions. After the file system is created, the disk gets read and write permissions.

4.2 Server commands definition

A set of special commands for the Server ensures that all planned features are supported. The following commands are defined:

- **Show** - command to display system element information. The only parameter is required and specifies the command syntax. The following syntaxes are supported:
  - **PORT** - presents port information.
  - **SYS_INFO** - presents system information.
  - **MOUNTED_LVOL** - shows a list of mounted volumes.
  - **MAPPED_LVOL** - shows a list of mapped volumes.
- **Mount** - command initiates a mounting of volume.
- **Umount** - command cancels a mounting.
- **Map** - command requests a storage to create LVOL and assign to LUN.
- **Unmap** - erases of host LUN of assigned volume. LVOL name is required.
- **CreateFS** - creating mounting of file system.
- **Get_volumes** - command returns a list of LUN volumes assigned to the host.
- **Discovery** - command updates all information about mounted and mapped volumes.
- **Set_multipath** - command allows multipath enabling. The parameter specifies the feature state: ON / OFF.

4.3 Storage

Storage core is Storage Data Object. Each storage instance, cannot have more then one Storage Data Objects.
. Each storage instance runs into separate terminal window and has own catalog tree. In one simulation there is a participation of several storage instances.
Core Entity contents a information about all physical and logical elements of Storage. All changes of the system elements are immediately written to the file representation of the changed storage element.

Each physical or logical element of the Storage is introduced in the form of a file which name corresponds to the system identifier of the element. Each such file contains information about the item.
Logical volumes presented in the form of files with information about the disk, and directories. Figure 4.1 describes the location of LVOL and its presence in the pool information file.

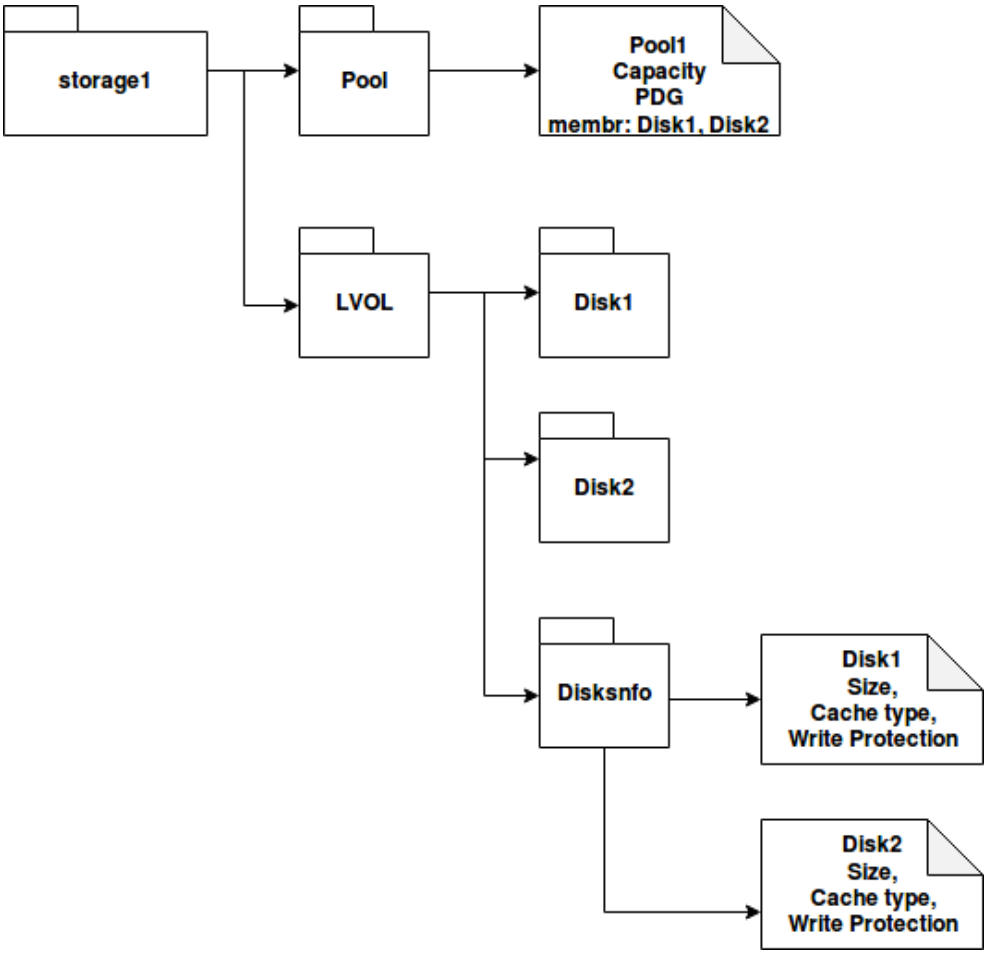

Figure 4.1: LVOL location

The following is a description of all storage directories:

- **sys** - directory content system information.
  This folder contains the name of the storage, WWID of the storage and the number of physical disks, ports, pools, registered hosts, volumes, groups.This folder contains the name of the storage, WWID of the storage and the number of physical disks, ports, pools, registered hosts, volumes, groups.

- **pool** - directory contents pool information.
- **pdg** - physical groups information.
- **pvol** - information about physical volumes.
- **lvol** - information about logical volumes.
- **port** - information about system FC ports.
- **host** - records of storage clients. Folder contents information about every particular server in form of file. Also servers LUN are stored in subfolder.

4.4 Storage commands definition

To implement the projected storage functionality, the following commands are implemented.

4.4.1 Add

The command is used to add a new element in the system. The command requires specification of the type of target and is used to add the following element types: PVOL, LVOL, LUN, PDG, Host. The unit type is specified as a mandatory command parameter, which will specify the command syntax. For each command syntax, a corresponding subroutine is defined.

The following are the options for specification of command syntax:

- **PDG** - creation of physical disks group. Command requires two additional parameters - group name and type of raid.
- **PVOL** - registration of physical disk. Command requires three additional parameters, that specify disk size in GB, ID of controller which drives the disk and disk interface (SSD, HDD, SATA, SAS or PATA).
- **LVOL** - creation of logical volume. The command requires specification of name alias for disk, disk size, cash type, state write protection function for that volume and identifier of pool, where the volume is created.
- **HOST** - host registration. The command requires entry of name alias, multipath support state and host WWID.

The following code listing shows the contents of the definition command for adding a physical group: 4.2

```
define verb ADD
    parameter P1, label=OPTION, prompt="What",
        value (required,type=ADD_OPTIONS)

define type ADD_OPTIONS
    KEYWORD PDG, syntax=ADD_PDG
    KEYWORD PVOL, syntax=ADD_PVOL
    KEYWORD HOST, syntax=ADD_HOST
    KEYWORD LVOL, syntax=ADD_LVOL

define syntax ADD_PDG
routine ADD_PDG
    parameter P1, LABEL=OPTION,
        VALUE(REQUIRED)
    parameter P2, label=PDG_NAME, prompt="Name"
        value(required,type=$IDENT)
    parameter P3, label=RAID, prompt="RAID"
        value(required,type=$RAID_KEYWORDS)


DEFINE TYPE RAID_KEYWORDS
    KEYWORD RAID_0, DEFAULT
    KEYWORD RAID_1,
    KEYWORD RAID_01,
    KEYWORD RAID_10,
    KEYWORD RAID_5,
    KEYWORD RAID_6
```

Figure 4.2: Add physical disk group, command definition

4.4.2 Remove

The command deletes an item from the system. For all possible purposes of adding commands, a delete command is supported that performs the deletion of an element of the same type. It's also defined at a type of syntax for deleting a member of a group. Each version of the command syntax runs a special procedure. The following syntaxes are supported:

- **PDG** - removal of physical disk group.

  command example: REMOVE PDG testGroup

- **PVOL** - removal of physical disk. Command requires specification of a disk ID.
- **LVOL** - removal of logical volume. Volume name is required.
- **HOST** - removal of host. Host name is required.
- **GROUP_MEMBER** - removal of disk from group. Disk id and group name are required.

4.4.3 Show

The command shows a list of all elements of the selected type. For PDG, pool, PVOL, LVOL, host, there is an optional parameter - device identifier, that allows to specify certain element to present. The following syntax options are supported.

- PDG
- POOL
- PVOL
- LVOL
- HOST
- HOST_LUN
- MOUNTED_LVOL
- STORAGE_PORT
- CLIENT_PORT
- SYS_INFO

command example: SHOW PDG testGroup

4.4.4 Set

The command performs a change to the system element. The following command syntaxes are defined.

- **LVOL** - command to change LVOL, syntax defines optional qualifiers to change name, capacity, cache, write protection, and one mandatory parameter to specify name of volume to change.

  command example:
  SET LVOL - -NEW_NAME=myVolume - -SIZE=1024
  —CACHE=WRITE_BACK myDisk

- **HOST** - command to change host properties, syntax consists of two optional qualifiers, which are used to change the alias and record of multipath support state. To specify a host to change, command requires specification of a host name.

4.4.5 Map

Command is used to assign a logical volume to host LUN. Host and volume names are required.

command example: MAP Prague MyVolume

4.4.6 Unmap

Command is used to cancel relation between logical volume and host LUN. Host and volume names are required.

command example: UNMAP Prague MyVolume

4.4.7 Mount

The command is used to mount lvol, that assigned to host lun, to host.

4.4.8 Umount

The command ensures the cancellation of mounting of volume, to host.

4.4.9 Extend_group

Command is used to cancel relation between logical volume to host LUN. Host name and volume name are required.

```
void ADD_PDG () {

  RAID_present = cli_present("RAID");
  RAID_result = cli_getvalue("RAID", &RAID_value);
  OPTION_present = cli_present("OPTION");
  OPTION_result = cli_getvalue("OPTION", &OPTION_value);
  PDG_NAME_present = cli_present("PDG_NAME");
  PDG_NAME_result = cli_getvalue("PDG_NAME", &PDG_NAME_value);
  ADD_PDG_run();
}

void ADD_PDG_run () {

}
```

Figure 4.3: ADD PDG routine

command example: EXTEND_GROUP MyGroup 1111,1112,1113,1114

4.5 Routines

Written definitions of the commands were passed to CDU processing to create the resulting command interface. After CDUs processing, functions were generated in the programming language C for all routines that were introduced with defining the command.Then all generated routines have been complemented to fulfill a purpose of the commands which have triggered them.

Figure 4.3 shows an example of generated by CDU functions.

Every service supported in program is introduced as a handler function. Enabling of service in program means to setting up appropriate handler function to component instance UDP port. To clarify the choice of a function handler for processing an input Datagram, each protocol has a protocol identifier. This protocol identifier is represented by a single letter and is added to each UDP Datagram. there are three services supported In Server and Storage parts of project:

- volumeFactory initiated by the Servers. Server sends FC frame to storage, request contents a required volume parameters. Storage provides creation of volume, then volume is assigned to server LUN.
- volumeInfo, initiated by Servers. Server sends the FC frame to storage, request contents absolute address of mount point and local volume

ID. After storage presents information about volumes, that assigned to server LUNs. Volumes information presents as a creation of a links on mount point on disk information files. Storage response contents number of volumes.

- volumeMount initiated by Servers. Server sends the FC frame to storage, request contents absolute address of mount point and storage volume id. Then Storage controls if volume is available to mount to this host. In success, Storage provides a mounting of volume.

the function sending an UDP Datagram, that contents the FC frame, with port parameters.

Control commands for manage simulation run are implemented as C language function.

Function “create” creates a directory tree of component. The function “create” like a function “run” opens UDP port and sets up the handler function to this port for other simulation control commands.
Connect function is setting up the port, with function input parameters, after it provides the FabricLogin service.
Destroy function cleans up everything by each component instance memory and remove all instance files and directories.
Function “start” provides the setting up handler function to component instance UDP port for all services that supported by the component.
Function “stop” supports the cancellation for all services.

Chapter 5

Testing

All Tests performed successfully. The planned functions of the components are tested and work as was expected.

5.1 Consistency

To Server and Storage components was several times sent a create and run commands with the different number of disks and ports. Was monitored catalog three of component, generation of WWID and storing of system information.

5.2 Components features

Storage instance was tested by entering commands to Storage terminal was entered a different commands, that not requires Storage to be connected.

Add, Remove, Set and Remove commands was monitoring by changing of system information, also result of command execution was followed in the catalog three.

Here was tested difference syntaxes types. Figure 5.1 shows examples of commands.

Show commands was tested and then was used for monitoring of another commands results.

5.3 Integration

The first was tested interaction with Control Command Simulation component.

```
ADD PDG "testGroup" RAID\_01
ADD PVOL 512 1 SSD
ADD LVOL MyDisk 1024 WRITE\_THROUGH OFF~1
ADD HOST Prague ON FC 5351879742643489~1
REMOVE "testGroup"
show HOST
```

Figure 5.1: Commands examples

To Server and Storage components after creating was sent a difference sequences of start, stop, exit and destroy commands.
Server and Storage components are presented ability to connect to Control component and furthermore ability to listen Control Component commands.

The second performed tested ability of Server and Storage components to communicate with Switch components.
Was tested command connect, that provides a registration of component FC port in Switch network. Result was controlled by entering to Switch terminal a command switchshow, after connection record of connected port had appeared in ports list, returned by switchshow command.

The next part of testing of Switch component interaction was command discovery, that provide information about devices, that ports are connected to network. After command had provided Server and Storage components extends a list of online devices in network and their ports.

5.4 Mapped and Mounting

The final testing group is mapping and mounting tests.
After mapping in server catolog three was appeared a links to volume information fails.
Fyrthermore was tested mountig command, and after command providing, in Server catalog three was appeared links to volumes, without right and write permissions. The last one, was tested create_FC operation, and after command was provide directory of target volumes became a readable and writable.

Conclusion

The main purpose of the thesis was the planning and realization of the server and storage parts of the Storage Network Command Simulator. Achievement of the goal of the thesis included several stages.
The wide analysis was carried out of existing simulators and real world solutions, also their physical and logical architecture, as well as work principles and command interfaces.
As part of the project, there was a study of the technology that are connected to the data storage and the storage network field.
Moreover, the study and the utilization of the programming instruments was mastered in this project. Those instruments are the frameworks, libraries, protocols and the command definition languages.

The data was obtained during the research was used to design of the command line interfaces, components structures and interaction of the components of the project.

The command definition language was used for definition of commands and the framework CDU-CLI was used for command line parsing during the creation of the command line interfaces.
Furthermore, the ttdrv framework and the io.h library was used for the realization of communication of components through the UDP Datagrams. Therefore, the function in the CI language was written for the command execution. During the working process there was the implementation of the possible connections and communication between the Switch component and Control component of the simulator. The implemented functionality provides representation of LUN and simulation of the data operations. Due to the creating of Links in filesystem, there was the realization of the mechanisms that allow to modeled the process of the mounting space storage for the Servers.

All objectives of the thesis have been achieved.

The project has the model structure which allows to add new elements and to change the old ones with the major changes of the project.
One of the direction of the expansion of the project is the distribution of the components of the project to the different computers and the creation of the local network between them.
Another one of the possible direction is the extension of the support of different types of connection and communication between the components.

Bibliography

[1] Petr Bouška: Fibre Channel SAN síť a konfigurace na Windows Server 2012. 2017, [cit. 20.04.2019]. Dostupné z: https://www.samuraj-cz.com/clanek/fibre-channel-san-sit-a-konfigurace-na-windows-server-2012/

[2] Carol Sliwa: Fibre Channel switch (FC switch). 2015, [cit. 20.04.2019]. Dostupné z: https://searchstorage.techtarget.com/definition/Fibre-Channel-switch-FC-switch

[3] IBM Knowledge Center: Switched fabric topology terminology. [cit. 20.04.2019]. Dostupné z: https://www.ibm.com/support/knowledgecenter/en/SSLTBW_2.1.0/com.ibm.zos.v2r1.ieag100/ieag1_diagswfabric_terms.htm

[4] Thomas C. Jepsen: *Distributed Storage Networks [Architecture, Protocols and Management]*. West Sussex: John Wiley and Sons Ltd, 2003, ISBN 0-470-85020-5, [cit. 20.04.2019].

[5] Earth Link for Abroad Networks: Storage. 2014, [cit. 20.04.2019]. Dostupné z: http://www.elan-eg.com/sys-storage.html

[6] THINKMATE: RAX-VM XT24-2262V4-VSANF. [cit. 20.04.2019]. Dostupné z: https://www.thinkmate.com/system/rax-vm-xt24-2262v4-vsanf

[7] Ilya Krutov: Lenovo ThinkSystem DE2000H Hybrid Storage Array [Product Guide]. 2019, [cit. 20.04.2019]. Dostupné z: https://lenovopress.com/lp0881-lenovo-thinksystem-de2000h-hybrid-storage-array

[8] QSAN Technology: XCubeSAN XS1200 (SMB). [cit. 20.04.2019]. Dostupné z: https://www.qsan.com/en/product.php?act=view&id=12

[9] Dell EMC: PowerVault ME4 Series SAN/DAS Storage Specification Sheet. 2018, [cit. 20.04.2019]. Dostupné z: https://www.dellemc.com/resources/en-us/asset/data-sheets/h17384-powervault-me4-series-ss.pdf

[10] Hewlett-Packard Development Company, L.P: *Hewlett-Packard Storage System Scripting Utility Reference*. Palo Alto, California, Hewlett-Packard Company, 2013, ISBN T5494-96594, [cit. 20.04.2019].

[11] Javvin Technologies Inc.: *Network Protocols Handbook*. 13485 Old Oak Road Saratoga CA 95070 USA, 2005, ISBN 408-872-3881, [cit. 20.04.2019].

Appendix A

Acronyms

FC Fibre channel

FCoE Fibre Channel over Ethernet

SAN Storage Area Network

LUN Logical unit number

WWN World Wide Name

LVOL Logical volume

SCSI Small Computer System Interface

iSCSI Internet Small Computer System Interface

Lenovo DE2000H Lenovo ThinkSystem DE2000H

QSAN XS1200 QSAN XCubeSAN XS1200 Series

Dell ME4 Dell EMC PowerVault ME4

CDU Command Definition Utility

CLI Command Language Interpreter

CT Command Table

DCL DIGITAL Command Language

PVOL Physical volume

PVG Physical Volume Group

LVG Logical Volume Group

SSSU Storage System Scripting Utility

Appendix B

Contents of enclosed CD

- `readme.txt` ..... Description of CD
- `exe` ..... exec file
- `src`
  - `impl` ..... application code
  - `thesis` ..... thesis text in LaTeX format
  - `thesis.pdf` ..... thesis text in PDF