FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

ZADÁNÍ DIPLOMOVÉ PRÁCE

Název:	Simulátor konfigurace storage sítě
Student:	Bc. Karel Gudera
Vedoucí:	Ing. Jiří Kašpar
Studijní program:	Informatika
Studijní obor:	Počítačové systémy a sítě
Katedra:	Katedra počítačových systémů
Platnost zadání:	Do konce letního semestru 2019/20

Pokyny pro vypracování

Nastudujte funkce FC přepínače Brocade, zaměřte se na konfiguraci linkových rozhraní, kaskádování a zónování. Navrhněte a implementujte simulátor FC přepínače pro nácvik praktických dovedností s oblasti konfigurace storage prostředí středních a větších firem.
Požadované funkčnosti:
1. konfigurační příkazy pro konfiguraci linkových rozhraní, kaskádování a zónování.
2. simulace propojení a komunikace přepínačů, serverů a storage pomocí UDP datagramů,
3. simulace základních služeb FC sítě potřebných pro komunikaci simulátorů serverů a storage.

Pro implementaci příkazového rozhraní použijte frameworku cli.
Funkčnost simulátoru podrobte testům použitelnosti podle pokynů vedoucího práce.

Seznam odborné literatury

Dodá vedoucí práce.

prof. Ing. Pavel Tvrdík, CSc.
vedoucí katedry

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

V Praze dne 4. prosince 2018

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Diplomová práce

Simulátor konfigurace storage sítě

Bc. Karel Gudera

Katedra počítačových systémů
Vedoucí práce: Ing. Jiří Kašpar

8. května 2019

Poděkování

Chtěl bych poděkovat svému vedoucímu práce Ing. Jiřímu Kašparovi za vedení této práce, plnou podporu a nadšení při tvorbě. Dále chci poděkovat všem mým blízkým, kteří mě podporovali během celého studia.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo"), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 8. května 2019 .....

Odkaz na tuto práci

Gudera, Karel. *Simulátor konfigurace storage sítě*. Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2019.

Abstrakt

Tato práce se zabývá návrhem a implementací simulátoru přepínače, který je založen na protokolu Fibre Channel. Přepínač je možné spustit ve více instancích. Je implementováno tzv. kaskádování, tzn. že dané přepínače je možné propojovat pomocí virtuálních spojů. Takto propojené přepínače tvoří danou storage síť (fabriku). Každý přepínač má vlastní příkazové rozhraní, pomocí kterého je možné ho různě konfigurovat, např. konfigurace portů, aliasů či konfigurace zón. Dále přepínače umí simulovat základní služby, které slouží pro přípojení a komunikaci s ostatními zařízeními jako jsou storage, servery a navíc koordinátor simulace, kterým je možné přepínače částečně ovládat. Všechna komunikace je realizována pomocí UDP datagramů. Důraz je kladen především na simulaci příkazového rozhraní, nikoli simulaci skutečných protokolů.

Klíčová slova Storage Area Network, Fibre Channel, přepínač, simulátor, konfigurace, příkazové rozhraní

Abstract

This thesis deals with design and implementation of switch simulator which is based on Fibre Channel protocol. It is possible to start multiple instances of switch. Cascading of switches is implemented, i.e. switches can be interconnected by virtual links. These switches form a storage area network (fabric). Each switch has its own command line interface with which it can be configured, e.g. port configuration, alias configuration or zoning configuration. Furthermore, switches are able to simulate basic services that are used to connect and communicate with other devices such as storage, servers and moreover, a simulation coordinator which can partially control switches. All communication is realized by UDP datagrams. Emphasis is placed on the simulation of the command line interface rather than simulation of actual protocols.

Keywords Storage Area Network, Fibre Channel, switch, simulator, configuration, command line interface

Obsah

Seznam obrázků

Úvod

Data jsou v dnešní době nedílnou součástí našeho světa. V posledních letech jejich objem rapidně narostl a nepochybně bude narůstat dál. Tím samozřejmě vzniká potřeba výstavby stále nových datových center. Za tím se skrývá celá řada problémů, které je potřeba řešit, jako např. jakou technologii zvolit, kde sehnat odborníky, kteří nám navrhnou potřebnou infrastrukturu, kde sehnat správce pro tuto infrastrukturu a technologii, atd.

Jedna z takových méně známých technologií je Fibre Channel. Tato technologie se používá především v SAN sítích (Storage Area Network) v komerčních datových centrech.

Celá Fibre Channel technologie se dá rozdělit do tří základních komponent.

- **Datové úložiště** – Většinou disková pole, na kterých jsou uložena data.
- **Server** – Počítač nebo počítače, které přistupují k datovému úložišti.
- **Síť přepínačů** – Přepínač nebo přepínače, které propojují úložiště a servery.

Nás v této práci budou zajímat právě přepínače, konkrétně jak pomocí softwaru nasimulovat chování skutečných fyzických zařízení. Předchozí výrok je třeba brát s rezervou, protože nasimulování všech fyzických vlastností a použitých protokolů není v mých silách. Ovšem simulace konfigurace přepínače by měla do jisté míry korespondovat s opravdovým zařízením, takže pokud přijde administrátor od skutečného přepínače k našemu „virtuálnímu“, měl by vidět jistou podobnost v ovládání.

Tato práce by tedy měla být prospěšná všem, kdo se chtějí naučit konfigurovat síť Fibre Channel přepínačů, aniž by měli nějaké k dispozici. Dále by se práce měla uplatnit v bakalářském předmětu BI-STO.

Kapitola 1

Technologie Fibre Channel, vymezení pojmů a cílů práce

1.1 Fibre Channel

Fibre Channel (FC) je technologie vysokorychlostního propojení, primárně používaná pro připojení úložišť k serverům [1]. FC se používá především v SAN sítích (Storage Area Network) v komerčních datových centrech. FC lze v OSI modelu přirovnat ke druhé vrstvě (linkové). Této vrstvě odpovídá také např. dobře známý Ethernet.

1.2 Fibre Channel Protocol

Fibre Channel Protocol (FCP) je transportní protokol, který přenáší převážně SCSI příkazy přes FC síť [1]. Nemusí však vždy nutně jít jen o SCSI příkazy, FCP lze použít i s jinou sadou příkazů. FCP pracuje nad FC, podobně jako TCP/IP nad Ethernetem.

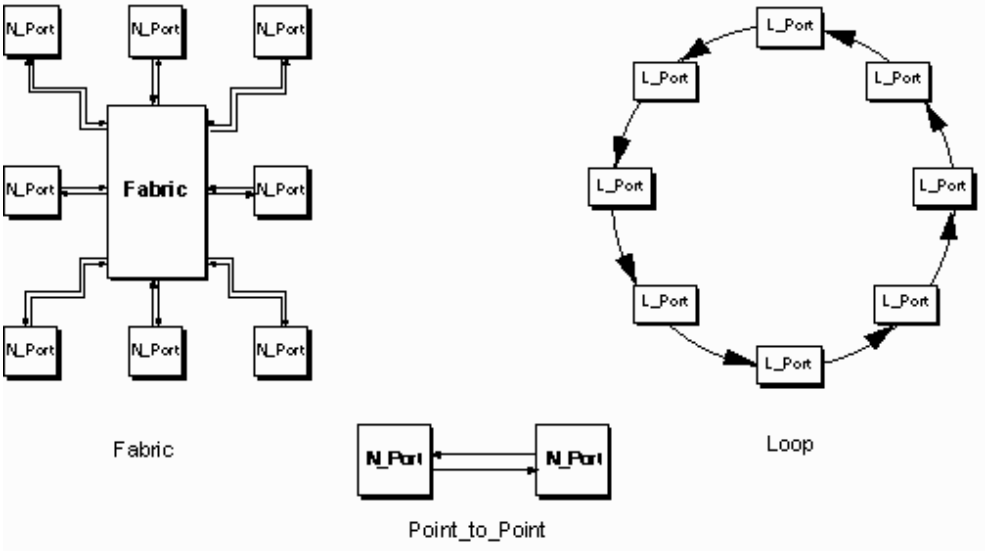
1.3 Topologie

Fibre Channel sítě používají tři základní topologie (viz obr. 1.1).

- **Point to Point** – Nejjednodušší topologie, kdy se dvě zařízení, např. server a storage, připojí přímo k sobě.
- **Loop** – Několik zařízení, která jsou navzájem spojená do logického kruhu. Komunikace je realizována pomocí tokenu, který si zařízení mezi sebou postupně předávají.

- **Fabric** – Dnes nejpoužívanější topologie, kdy je několik zařízení spolu propojeno pomocí jednoho nebo více přepínačů. Pokud je přepínačů více, tvoří spolu tzv. Fabriku, která se logicky tváří jako jeden velký přepínač.

Jelikož se dnes topologie typu Point to Point a Loop téměř nepoužívají, budeme dále uvažovat jen topologii typu Fabric.



Obrázek 1.1: Fibre Channel topologie [2]

1.4 Typy portů

Port na serveru

Server se do Fibre Channel sítě připojuje přes tzv. Host Bus Adapter (HBA), což je obdoba síťové karty. Tato karta může mít jeden či více portů.

Port na úložišti

Obdobně jako u serveru, úložiště také poskytují adaptéry, kterými se lze připojit do sítě. Opět, jeden adaptér může mít jeden nebo více portů pro připojení.

Port na přepínači

Klasický port na přepínači, který slouží k připojení zařízení nebo k připojeni dalšího přepínače do kaskády. Přepínače mívají typicky od 8 do 128 portů, ale mohou mít i více.

Rozdělení

Podle funkce rozlišujeme různé typy portů:

- **E_Port** – Port přepínače, který propojuje přepínače do kaskády.
- **F_Port** – Port přepínače, který propojuje přepínač se serverem či úložištěm.
- **FL_Port** – Port přepínače, který slouží pro připojení zařízení používající topologii typu smyčka.
- **FX_Port** – Port přepínače, ze kterého se stane F_Port nebo FL_Port, podle toho co připojíme.
- **N_Port** – Port zařízení, který připojuje zařízení k přepínači.
- **L_Port** – Port zařízení, který připojuje zařízení do smyčky.
- **NL_Port** – Port zařízení, který se připojuje k přepínači i smyčce.
- **U_Port** – Univerzální port používaný pro libovolný z portů FC.

Toto jsou základní typy portů pro Fibre Channel. Existuje jich ještě více, ale přišlo mi zbytečné je zde uvádět. Jelikož se nezabýváme topologií smyčky (loop), tak si vystačíme pouze s porty typu E_Port, F_Port, N_Port a U_Port.

1.5 Adresace

World Wide Name

World Wide Name (WWN) je unikátní identifikátor v sítích Fibre Channel. Jedná se o obdobu MAC adresy v Ethernetu. WWN má délku 8 bajtů. WWN se dále dělí na dvě podkategorie.

- **World Wide Node Name (WWNN)** – WWN koncového zařízení (HBA, storage, switch).
- **World Wide Port Name (WWPN)** – WWN konkrétního portu na zařízení.

Port ID adresace

Základní Port ID (PID, někdy také FCID) adresace spočívá v přidělení 3 bajtové adresy každému portu ve fabrice.

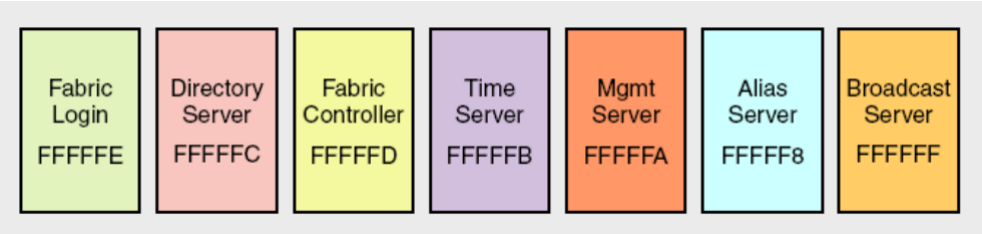
- **Domain ID** – Tento bajt adresy adresuje konkrétní přepínač. Jeden bajt nám dává 256 možných adres. Jelikož jsou ale některé adresy rezervované, máme k dispozici pouze rozsah od 1 do 239. To znamená, že můžeme mít maximálně 239 přepínačů v našem SAN prostředí.

- **Area ID** – Tímto bajtem adresujeme konkrétní port na přepínači. Zde nám specifikace dává rozsah od 1 do 255.
- **Node ID** – Posledním bajtem adresujeme port konkrétního zařízení. Tato část adresy má význam pouze pokud k přepínači připojujeme více zařízení ve smyčce, aby je bylo možné odlišit. Jak už bylo zmíněno dříve, my tento druh připojení uvažovat nebudeme, proto budou mít porty připojených zařízení adresy shodné s porty přepínače.

Existuje ještě několik adresovacích schémat, které dané 3 bajtové adresy rozdělují jinak. My si ovšem vystačíme pouze s tímto základním schématem.

Well-known adresy

Fibre Channel rezervuje několik adres, které se používají pro služby, které fabrika nabízí ostatním zařízením (viz obr. 1.2).



Obrázek 1.2: Fibre Channel služby [3]

- **Fabric Login** – Slouží k připojeni nového zařízení do fabriky. Přiřadí zařízení adresu, pomocí, které zařízení komunikuje s ostatními.
- **Directory Server** – Slouží k registraci zařízení a jejich atributů připojujících se do fabriky. Zařízení se poté mohou dotazovat na ostatní dostupná zařízení.
- **Fabric Controller** – Poskytuje informace o změnách registrovaným uzlům, pokud ve fabrice nastane nějaká změna.
- **Time Server** – Posílá informaci o čase všem přepínačům ve fabrice. Slouží k synchronizaci.
- **Management Server** – Poskytuje přístupový bod ke správě fabriky.
- **Alias Server** – Drží skupiny zařízení registrované pod jedním jménem. Slouží k multicastu.
- **Broadcast server** – Slouží k rozeslání rámců s touto adresou na všechny N_Porty a NL_Porty.

1.6 Aliasing

Jelikož jsou WWN zařízení a portů dlouhá čísla, která se špatně pamatují, tak máme možnost tzv. aliasingu. Nejedná se o nic jiného než jmennou službu, která dovoluje přiřadit jméno danému WWN.

1.7 Zónování

Zónování je mechanismus, který dovoluje logicky oddělit zařízení, která jsou spolu fyzicky propojena do jednotlivých oblastí (zón). Zařízení spolu mohou komunikovat pouze pokud náleží do stejné zóny. Zónování se týká pouze N_Portů a nastavuje se na přepínači.

1.8 Principal přepínač

Každá fabrika má svého „vůdce“, kterým je principal přepínač. Ten se volí na základě WWN přepínačů. Přepínač s nejmenším WWN je zvolen jako principal. Principal přepínač slouží k synchronizaci času pro celou fabriku a dále k přidělování doménového ID přepínači, který není součástí žádné fabriky. My ovšem z implementačních důvodů budeme mít pouze pevná doménová ID a budeme předpokládat, že samotný přepínač vždy tvoří fabriku o jednom přepínači. Může se to zdát trochu jako podvod, ale ve skutečnosti pokud se přepínač v iniciální fázi nepřipojí k žádné fabrice, tak opravdu zformuje fabriku o jednom přepínači.

1.9 Specifikace cílů práce

Nyní když známe základní pojmy, můžeme plně specifikovat cíle této práce.

1. **Simulace přepínače** – Přepínač bude simulován v samostatném procesu. Při spuštění procesu se vytvoří složka, kde se budou uchovávat různé konfigurační informace. Přepínač bude nabízet rozhraní příkazové řádky, které bude sloužit k jeho konfiguraci. Síťová komunikace přepínače s ostatními zařízeními nebo přepínači bude simulována pomocí protokolu UDP. K těmto účelům nám pomůže dodaný framework, který si popíšeme v dalších kapitolách.
2. **Kaskádování** – Bude implementováno propojení více přepínačů do kaskády. Dané přepínače poté budou tvořit danou FC fabriku.
3. **Základní příkazy** – Budou implementovány základní příkazy sloužící pro konfiguraci portů, aliasů a zónování.
4. **Základní služby** – Budou implementovány základní služby sloužící pro připojení serverů a úložišť.

5. **Platforma** – Celé řešení bude vyvíjené v jazyce C na Linuxové platformě.

Kapitola 2

Přehled současných řešení

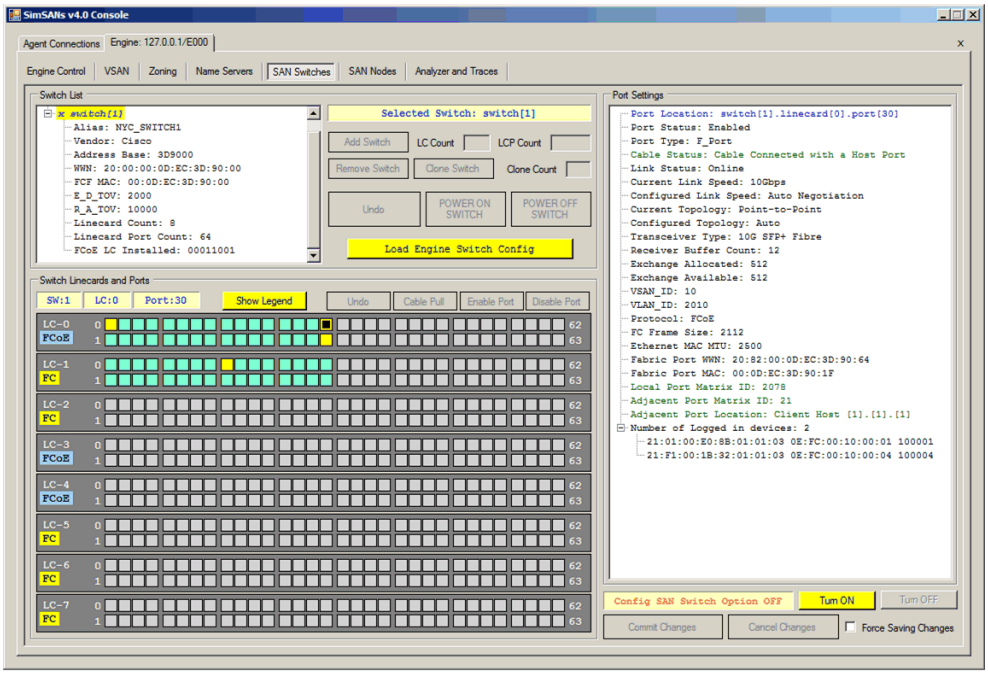
Podle [4] v současné době není žádný dostupný simulátor FC SAN sítě založený na přepínači od firmy Brocade. Jinými slovy, nepodařilo se mi dopátrat takový simulátor, který by nabízel příkazové rozhraní jako skutečné přepínače od firmy Brocade. Nicméně existují jisté simulátory, ale většinou se dají konfigurovat pouze přes grafické rozhraní. Dále existují simulátory, které jsou založené na klasickém Ethernetu.

2.1 SimSANs

SimSANs [5] je velice komplexní simulátor FC sítě. Jde asi o nejznámější (a nejspíše jediný) simulátor, který je dostupný zadarmo. Simulátor nabízí pro konfiguraci grafické rozhraní a je dostupný pouze pro platformu Windows. Mimo jiné implementuje skutečné chování protokolů jako je např. SCSI či FCoE. Jak simulátor vypadá si můžete prohlédnout na obrázku 2.1.

2.2 Cisco Packet Tracer

Packet Tracer [6] je velice populární simulátor od firmy Cisco. Simulátor je založený na Ethernetu a klasickém TCP/IP modelu. Nabízí grafické rozhraní kam je možno přidávat přepínače, směrovače, servery či koncová zařízení a různě je mezi sebou propojovat. Ke konfiguraci daných zařízení lze použít také grafické rozhraní nebo autentické rozhraní příkazové řádky. Jde o velice propracovaný nástroj, který se používá celosvětově pro školení síťových administrátorů.



Obrázek 2.1: SimSANs [7]

2.3 fake-switches

Fake-switches [8] je simulátor příkazového rozhraní různých přepínačů a směrovačů. Je implementováno několik modelů např. od firmy Cisco a Dell. Simulátor slouží hlavně k testování konfiguračních skriptů. K přepínačům se dá připojit i vzdáleně např. pomocí SSH. Celé řešení ja napsané v jazyce Python.

2.4 brocade-sim

Brocade-sim [9] je simulátor Ethernetového přepínače od firmy Brocade. Jde pouze o emulaci příkazového rozhraní, takže zde není implementováno propojení do kaskády či jiná funkcionalita, která by zajišťovala komunikaci mezi přepínači či jinými zařízeními.

2.5 sansimshell

Sansimshell [10] jsem po delším hledání našel na Githubu. Jde o velice prostou emulaci jednoho FC přepínače od firmy Brocade, který nabízí jen pár

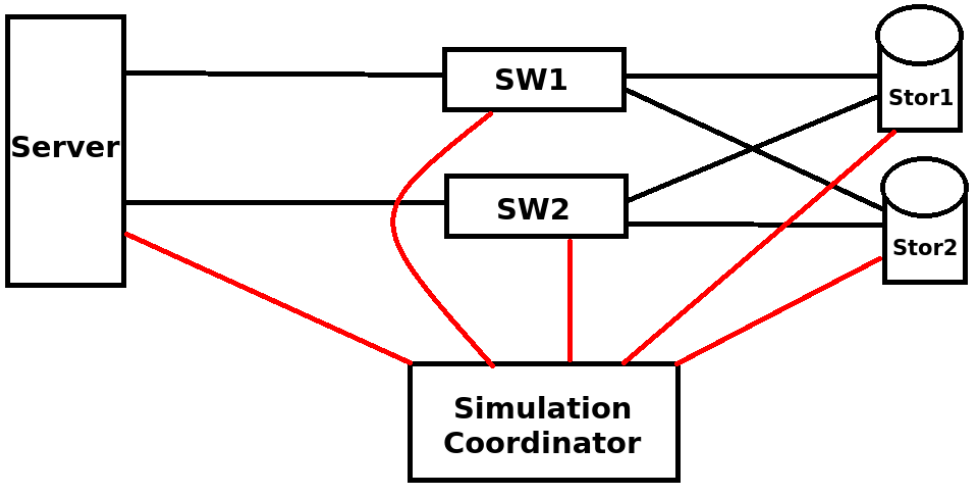
základních příkazů týkajících se zónování. Není zde implementováno kaskádování nebo služby pro připojení ostatních zařízení. Jak název napovídá, simulátor je celý napsaný v jazyce Shell.

Kapitola 3

Návrh řešení

3.1 Architektura řešení

Celková architektura by měla respektovat celky, tzn. každá komponenta (storage, server, switch a řídicí konzole simulace) v samostatném procesu s vlastním terminálovým oknem a odpovídajícím příkazovým rozhraním. S ostatními bude komunikovat pomocí UDP. Celkovou architekturu znázorňuje obrázek 3.1.



Obrázek 3.1: Celková architektura

3.2 Použitý framework

Při implementaci se využívá frameworku cli, který mi dodal vedoucí práce Ing. Jiří Kašpar. Tento framework se skládá ze dvou stěžejních částí. První část slouží k emulací příkazového rozhraní a komunikaci pomocí UDP datagramů. Druhá část se používá k parsování příkazů získaných z daného příkazového rozhraní a předání k dalšímu zpracování.

3.2.1 Framework ttdrv

Tento framework slouží k emulaci příkazového rozhraní našeho přepínače. Umí tedy načítat řádky ze standardního vstupu, které se mohou předat k dalšímu zpracování. Mimo toho navíc v pozadí poslouchá na zvoleném UDP portu a pokud dorazí nějaký paket, vyvolá se přerušení a zavolá se odpovídající obslužná rutina. Obslužných rutin může být více, prakticky až 256. Framework nabízí funkci pro zaregistrování či odregistrování rutiny pro specifický protokol. Protokol je určen prvním bajtem posílaných UDP paketů. Danou rutinu můžeme zaregistrovat jen dočasně a pokud do určité doby nedorazí paket daného protokolu uplatní se timeout, který si nastavíme. Tato vlastnost frameworku se dá využít i pro rutiny, které je nutné volat periodicky (např. posílání „HELLO zpráv“) tak, že po uplynutí timeoutu rutinu neodregistrujeme a nastavíme nový timeout. Je to velice příjemný způsob jak nasimulovat vlákno, aniž bychom museli řešit zamykání na sdílených prostředcích. Ještě stojí za zmínku, že framework pro svou činnost využívá i druhý bajt paketu. Tento bajt slouží pro komunikaci typu request-response, ke které slouží funkce UDPsend_recv. Aby komunikace fungovala request musí v druhém bajtu obsahovat znak (char) A–Z a response musí odpovědět odpovídajícím a–z na stejném protokolu.

Vrstva Buffer

Framework pro svou činnost používá mezivrstvu Buffer, což je dynamická struktura textového bufferu. Zasílané zprávy jsou tedy textového charakteru a k formátovanému zápisu a čtení zpráv se nabízí použití funkcí sprintf resp. sscanf.

3.2.2 Framework cdu

Tento framework slouží k parsování příkazů získaných z příkazové řádky. Jde o parser, který je převzat z operačního systému OpenVMS a portován do jazyka C. K definici příkazů se používají tzv. CLD soubory (Command Language Definition files). Z těchto souborů je poté vygenerován kód jazyka C, který je zodpovědný za parsování daných příkazů.

Takto vypadá začátek souboru, ve kterém jsou definovány příkazy přepínače.

```
define verb switchenable
    routine switchenable
    noparameters
    noqualifiers

define verb switchdisable
    routine switchdisable
    noparameters
    noqualifiers

define verb portenable
    routine portenable
    parameter P1, label=PORTNUM, prompt="Port Number"
        value(type=$NUMBER, required)
    noqualifiers

define verb portdisable
    routine portdisable
    parameter P1, label=PORTNUM, prompt="Port Number"
        value(type=$NUMBER, required)
    noqualifiers

define verb aliadd
    routine aliadd
    parameter P1, label=ALINAME, prompt="Alias Name"
        value(type=$STRING, required)
    parameter P2, label=MEMBERS, prompt="Member(s)"
        value(type=$STRING, required)
    noqualifiers

define verb alicreate
    routine alicreate
    parameter P1, label=ALINAME, prompt="Alias Name"
        value(type=$STRING, required)
    parameter P2, label=MEMBERS, prompt="Member(s)"
        value(type=$STRING, required)
    noqualifiers
```

3.3 Návrh datových struktur

Datové struktury jsem se snažil navrhnout tak, aby dané komponenty byly co nejvíce nezávislé. Všechny dílčí části mají svůj „konstruktor“ a „destruktor“. Konstruktor vždy alokuje danou strukturu na haldě a vrátí na ni ukazatel. Všechny funkce pracující se strukturou mají jako první parametr ukazatel na tuto strukturu. Destruktor uvolní paměť alokovanou pro tuto strukturu.

3.3.1 Switch

Hlavní struktura, která obsahuje vše co je potřeba pro běh programu.

```
1  typedef struct Switch
2  {
3      IOhandle * tthandle;
4      IOhandle * udphandle;
5      Buffer * switchIPAddress;
6      Buffer * name;
7      Buffer * buffer;
8      Buffer * prompt;
9      Buffer * history;
10     PortArray * portArray;
11     ZoneConfig * zoneConfig;
12     ZoneConfig * transZoneConfig;
13     NameServer * nameServer[MAX_SWITCHES];
14     Buffer * otherNames[MAX_SWITCHES];
15     uint16_t switchUDPPort;
16     uint8_t topology[MAX_SWITCHES][MAX_SWITCHES];
17     uint8_t nextHops[MAX_SWITCHES];
18     uint64_t otherWwns[MAX_SWITCHES];
19     uint64_t lastHello[MAX_SWITCHES];
20     uint64_t lastTopologyChange[MAX_SWITCHES];
21     uint8_t state;
22     uint8_t switchId[3];
23     uint8_t switchWwn[8];
24     uint8_t principalId;
25     uint8_t principalLock;
26     int terminators[4];
27     char execDirectory[1024];
28     char switchDirectory[1024];
29 } Switch;
```

- **tthandle** – Handle terminálu frameworku ttdrv.
- **udphandle** – Handle UDP listeneru frameworku ttdrv.
- **switchIPAddress** – IP adresa přepínače.
- **switchUDPPort** – UDP port, na kterém přepínač poslouchá.
- **name** – Jméno přepínače.
- **buffer** – Buffer, do kterého se načítají příkazy z terminálu.
- **prompt** – Prompt příkazového rozhraní.
- **history** – Historie příkazového rozhraní.
- **portArray** – Struktura držící informace o „fyzických“ portech přepínače.

- **zoneConfig** – Struktura uchovávající aktivní konfiguraci aliasů a zón.
- **transZoneConfig** – Struktura uchovávající lokální změny v konfiguraci aliasů a zón, než dojde k jejich potvrzení.
- **nameServer** – Pole struktur uchovávající informace o připojených zařízeních v celé fabrice (servery, storage).
- **otherNames** – Jména ostatních přepínačů, které jsou součástí fabriky.
- **otherWwns** – WWN ostatních přepínačů, které jsou součástí fabriky.
- **topology** – Matice sousednosti přepínačů ve stejné fabrice (všechny cesty jsou délky 1).
- **nextHops** – Vypočítané nejkratší cesty k ostatním přepínačům.
- **lastHello** – Čas poslední HELLO zprávy ostatních přepínačů ve fabrice.
- **lastTopologyChange** – Čas poslední změny topologie ostatních přepínačů ve fabrice.
- **state** – Stav přepínače (ENABLED/DISABLED).
- **switchId** – FCID přepínače.
- **switchWwn** – WWN přepínače.
- **principalId** – První bajt FCID principal switche ve fabrice.
- **principalLock** – Zámek sloužící k synchronizaci pomocí principal přepínače (viz kapitola implementace).
- **terminators** – Speciální znaky ukončující čtení z terminálu.
- **execDirectory** – Cesta k adresáři se spustitelným souborem přepínače.
- **switchDirectory** – Cesta k adresáři obsahující konfigurační soubory přepínače.

3.3.2 Port

Struktura Port reprezentuje fyzický port přepínače. Tuto strukturu obaluje další struktura PortArray, které se při inicializaci předá požadovaný počet portů, které se mají alokovat. PortArray je poté využívána ve struktuře Switch (viz výše).

```
1  typedef struct Port
2  {
3      Buffer * dstIPAddress;
4      Buffer * name;
5      int dstUDPPort;
6      int dstPortIndex;
7      int srcPortIndex;
8      uint8_t physicalState;
9      uint8_t logicalState;
10     uint8_t status;
11     uint8_t type;
12     uint8_t portId[2];
13     uint8_t portIfId[4];
14     uint8_t portWwn[8];
15     uint8_t connectedSwitchId;
16     uint8_t connectedDeviceWwn[8];
17     uint8_t connectedPortWwn[8];
18     uint64_t lastHello;
19 } Port;
20
21 typedef struct PortArray
22 {
23     Port ** ports;
24     int numPorts;
25 } PortArray;
```

- **dstIPAddress** – Cílová IP adresa připojeného přepínače.
- **dstUDPPort** – Cílový UDP port připojeného přepínače.
- **name** – Jméno portu.
- **srcPortIndex** – Index lokálního portu.
- **dstPortIndex** – Index cílového portu.
- **physicalState** – Indikátor jestli je port někam „fyzicky“ připojen.
- **logicalState** – Indikátor jestli je port zapnut či vypnut.
- **status** – Status připojení (např. CONNECTION_ESTABLISHED).
- **type** – Typ portu. V našem případě buď E_Port nebo F_Port.
- **portId** – Spodní 2 bajty FCID portu.
- **portIfId** – 4 bajtové ID interfacu portu. Bohužel jsem nikde nenašel k čemu toto ID slouží.
- **portWwn** – WWN portu.

- **connectedSwitchId** – První bajt FCID připojeného přepínače (pokud jde o E_Port).
- **connectedDeviceWwn** – WWN připojeného zařízení.
- **connectedPortWwn** – WWN cílového portu.
- **lastHello** – Čas poslední HELLO zprávy, kterou port obdržel.

3.3.3 ZoneConfig

ZoneConfig je struktura uchovávající všechny informace o týkající se zón. Je tvořena třemi podstrukturami.

```
1  typedef struct Alias
2  {
3      char * aliName;
4      char ** members;
5      int numMembers;
6      int maxMembers;
7  } Alias;
8
9  typedef struct Zone
10 {
11     char * zoneName;
12     char ** members;
13     int numMembers;
14     int maxMembers;
15 } Zone;
16
17 typedef struct Config
18 {
19     char * cfgName;
20     char ** members;
21     int numMembers;
22     int maxMembers;
23 } Config;
24
25 typedef struct ZoneConfig
26 {
27     Alias ** aliases;
28     Zone ** zones;
29     Config ** configs;
30     char * defZone;
31     char * enable;
32     int numAliases;
33     int numZones;
34     int numConfigs;
35     int maxAliases;
36     int maxZones;
37     int maxConfigs;
38 } ZoneConfig;
```

- **Alias** – Alias má své jméno a lze do něj přidat libovolný počet členů, kterými jsou WWN nebo dvojice Domain ID a Port Index (D,I). Proměnná numMembers uchovává počet členů v aliasu a proměnná maxMembers skutečnou délku alokovaného pole. Proměnná maxMembers slouží především k dynamické realokaci paměti, stejně je tomu i v ostatních strukturách.
- **Zone** – Zóna podobně jako alias má taky své jméno a lze do ní přiřadit to samé co do aliasu, ale navíc může obsahovat právě jméno aliasu.
- **Config** – Do struktury Config lze přiřadit libovolný počet zón.

ZoneConfig obaluje tyto tři struktury a drží veškeré jejich definice. Dále obsahuje proměnou defZone, která určuje chování pokud je zónování vypnuté. Může nabývat hodnoty allaccess (každý vidí každého) nebo noaccess (nikdo nevidí nikoho). Proměnná enable určuje aktuální aktivní konfiguraci.

3.3.4 NameServer

NameServer je struktura, která drží všechny potřebné informace o lokálně připojených zařízeních (servery, storage).

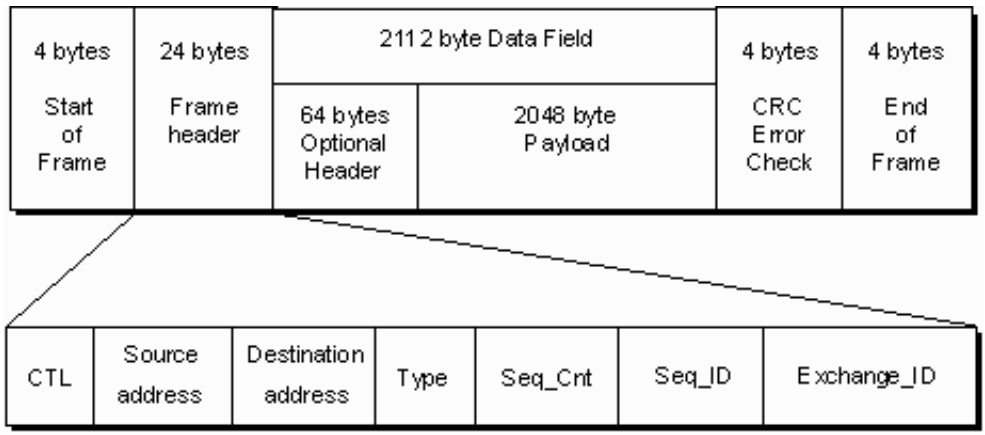
```
1  typedef struct NSEntry
2  {
3      int switchId;
4      int portIndex;
5      int devicePortIndex;
6      int UDPPort;
7      char IPAddr[32];
8      uint64_t FPortWwn;
9      uint64_t NPortWwn;
10     uint64_t deviceWwn;
11     char deviceName[256];
12 } NSEntry;
13
14 typedef struct NameServer
15 {
16     NSEntry ** entries;
17     int numEntries;
18     int maxEntries;
19 } NameServer;
```

- **switchId** – Domain ID přepínače.
- **portIndex** – Index portu přepínače, na kterém je zařízení připojeno.
- **devicePortIndex** – Index portu připojeného zařízení.
- **IPAddr** – IP adresa připojeného zařízení.

- **UDPPort** – UDP port připojeného zařízení.
- **FPortWwn** – WWN portu na přepínači.
- **NPortWwn** – WWN portu na zařízení.
- **deviceWwn** – WWN připojeného zařízení.
- **deviceName** – Jméno připojeného zařízení.

3.3.5 FCFrame

FC rámce lze rozdělit v zásadě do dvou skupin. První skupina rámců slouží ke konfiguraci fabriky a druhá skupina jsou datové rámce, které jsou směrovány napříč fabrikou. Jak FC rámec vypadá znázorňuje obrázek 3.2. Je jasné, že pole jako start of frame či end of frame nepotřebujeme, vlastně si vystačíme pouze s hlavičkou rámce a payloadem. Mimo těchto dvou skupin navíc potřebujeme rámce zajišťující komunikaci s koordinátorem simulace. Všechny typy rámců jsou simulovány pomocí UDP datagramů.



Obrázek 3.2: Fibre Channel rámec [11]

- **CTL** – Slouží k odlišení kontrolních rámců od datových.
- **Source Address** – 3 bajtová adresa zdroje.
- **Destination Address** – 3 bajtová adresa cíle.
- **Type** – Spolu s CTL slouží pro další odlišení rámců.
- **Seq_Cnt** – Slouží pro číslování rámců, pokud vysíláme nějakou sekvenci. Podobně jako sequence number v TCP.

- **Seq_ID** – Odlišuje sekvence.
- **Exchange_ID** – Spojuje více sekvencí dohromady a vytváří obousměrnou komunikaci.
- **Payload** – Obsahuje samotná data.

Budu se tedy držet jednoduché konvence. Každý rámec bude obsahovat danou hlavičku a payload. Pomocí CTL budu rozlišovat jednotlivé rámce, např. jestli se jedná o konfiguraci fabriky, datový rámec nebo rámec od koordinátora simulace. CTL bude zároveň odpovídat prvnímu bajtu posílaného paketu, kterým framework ttdrv odlišuje různé protokoly. Zdrojovou a cílovou adresu asi netřeba vysvětlovat, jen stojí za zmínku, že pokud v některých rámcích nemají adresy smysl, tak budou nastavené na nulu (0x000000). Proměnná type bude sloužit k dalšímu dělení rámců, např. rámce se stejným CTL, které plní stejnou funkci, ale mají malinko jiný význam. Zbylé položky hlavičky mají smysl v případě, že posíláme vetší množství dat a je třeba ho rozdělit do více rámců a poslat jako sekvenci. UDP paket může mít velikost až 64 KiB, což je pro naše účely dostatečná velikost na to abychom nemuseli tento problém fragmentace řešit. Samotný payload už bude obsahovat data nějakého vyššího protokolu, který budeme implementovat.

Struktura FC rámce se snaží přiblížit skutečnému rámci. Navíc obsahuje proměnné ttdrvProt1 a ttdrvProt2, které odpovídají prvním dvěma bajtům používaných frameworkem a srcPortIndex a dstPortIndex, abychom věděli ze kterého portu na který port rámec posíláme. Dále jsou zde dvě funkce, které zajišťují konverzi ze struktury FCFrame na strukturu Buffer a naopak, protože framework pro posílání UDP zpráv pracuje pouze se strukturou Buffer.

```
1  typedef struct FCFrame
2  {
3      uint8_t ttdrvProt1;
4      uint8_t ttdrvProt2;
5      uint8_t srcPortIndex;
6      uint8_t dstPortIndex;
7      uint8_t CTL;
8      uint8_t type;
9      uint32_t srcAddr;
10     uint32_t dstAddr;
11     uint32_t seqCnt;
12     uint32_t seqId;
13     uint32_t exchId;
14     char payload[MAX_PAYLOAD_SIZE];
15 } FCFrame;
```

Další zajímavá položka je indikátor, který určuje zda jsme navštívili přepínač s daným Domain ID. Tento indikátor neobsahují všechny rámce, ale používá se

u rámců, které slouží k záplavovému algoritmu. Předcházíme tak tzv. broadcastové bouři, která je dobře známá z Ethernetových sítí. Přišlo mi jednoduší použít takovýto způsob ochrany před cykly než implementovat např. Spanning Tree Protocol.

Kapitola 4

Implementace řešení

4.1 Synchronizace topologie

Každý přepínač má globální pohled na topologii fabriky. Danou topologii reprezentujeme jako matici rozměru 240x240. Ve fabrice může být totiž maximálně 239 přepínačů indexovaných od 1 do 239. Nultý řádek a sloupec je zde navíc kvůli jednoduchosti indexace.

4.1.1 Link State Update

Link State Update zprávy (LSU) jsou zprávy, které se posílají při změně na nějaké lince (přidání/odebrání linky). K šíření těchto zpráv slouží záplavové rámce.

Typy LSU zpráv jsou následující:

- **LSU Request** – LSU Request je zpráva, která říká všem přepínačům ve fabrice, že je potřeba aby do fabriky poslali informace o svých sousedech.
- **LSU Response** – Odpověď na LSU Request.
- **LSU Link Up** – Zpráva, která informuje o přidání linky.
- **LSU Link Down** – Zpráva, která informuje o ztrátě linky.

Mimo LSU Request mají zprávy takovýto obsah:

- **Domain ID** – Domain ID přepínače, který zprávu posílá.
- **WWN** – WWN přepínače, který zprávu posílá.
- **Name** – Jméno přepínače, který zprávu posílá.
- **Timestamp** – Časová značka poslední změny na přepínači.

- **Neighbours** – Sousedé přepínače (řádka matice sousednosti na indexu Domain ID).
- **Nameserver Info** – Informace lokálního Name Serveru přepínače.

4.1.2 Výpočet cest

Pokud na přepínač dorazí zpráva LSU Response nebo LSU Link Up provedou se následující akce:

1. Pokud je časová značka zprávy starší než poslední časová značka, kterou máme uloženou pro daný přepínač tak nic neděláme.
2. V opačném případě aktualizujeme časovou značku, řádek matice sousednosti na indexu obdrženého Domain ID a ostatní informace obdržené od daného přepínače.
3. Pomocí Dijkstrova algoritmu [12] spočítáme Next Hop pro každý přepínač ve fabrice.

Pokud na přepínač dorazí zpráva LSU Link Down provedou se následující akce:

1. Stejně jako v předchozím případě zprávu ignorujeme, pokud má starší časovou známku.
2. Jinak aktualizujeme seznam sousedů pro daný přepínač a všechny potřebné informace.
3. Jelikož jde o ztrátu linky, tak pomocí algoritmu Depth First Search [13] nalezneme všechny nedosažitelné uzly a nastavíme všechny jejich atributy na výchozí hodnoty.
4. Pomocí Dijkstrova algoritmu spočítáme Next Hop pro všechny zbývající přepínače ve fabrice.

4.2 Volba principal přepínače

Principal switch se volí na základě WWN. Switch se nejnižším WWN je zvolen jako principal. Existují tři případy, kdy je nutné zvolit nový principal switch.

1. **Propojení dvou samostatných přepínačů** – Zde je to velice přímočaré, přepínače si vymění svá WWN a oba zvolí stejný principal switch. Od této chvíle, pokud se do fabriky připojuje další samostatný přepínač, tak je mu přiřazen principal switch fabriky.

2. **Spojení dvou fabrik** – Pokud se spojují dvě fabriky, aktivuje se algoritmus pro volbu nového principal přepínače.

3. **Ztráta principal přepínače** – Poslední případ kdy je potřeba zvolit principal přepínač je, když se stane nedosažitelným. Tato situace může nastat pouze v případě obdržení LSU Link Down. Stejně jako v předchozím případě se aktivuje algoritmus pro volbu nového principal přepínače.

Algoritmus volby principal přepínače

Pro volbu principal přepínače jsem se rozhodl implementovat tzv. Bully algoritmus [14], což je distribuovaný algoritmus pro volbu synchronizátora systému.

Algoritmus používá tři typy zpráv:

- **Election** – Oznamuje zahájení volby.
- **Alive** – Odpověď na Election zprávu.
- **Victory** – Oznamuje zvolení nového synchronizátora.

Postup volby je následující (předpokládejme, že přepínač P inicializuje volbu):

1. Pokud má P nejmenší WWN, pošle Victory zprávu všem ostatním přepínačům, jinak pošle Election zprávu všem přepínačům s menším WWN.
2. Pokud P nedostane žádnou Alive odpověď do určitého času, pošle Victory všem ostatním přepínačům.
3. Pokud P dostane Alive odpověď od přepínače s menším WWN, neposílá žádné další zprávy a čeká na Victory zprávu. Pokud ji do určitého času nedostane, restartuje proces volby.
4. Pokud P dostane Election zprávu od přepínače s vetším WWN, pošle mu Alive odpověď a restartuje proces volby tím že pošle Election zprávu přepínačům s nižším WWN.
5. Pokud P dostane Victory zprávu, nastaví odesílatele jako nového synchronizátora.

4.3 E_Port Login

E_Port Login, neboli propojení do kaskády je proces kdy se dva přepínače snaží navázat spojení a nějakým způsobem zformovat fabriku. Tento proces není úplně triviální, jelikož je potřeba, aby v danou chvíli probíhal pouze jeden login ve fabrice. Důvodem je, že zónovací konfigurace je sdílená pro celou fabriku, a

pokud by nebyl zajištěn výlučný přístup mohlo by dojít k nekonzistenci. Pro tento účel se nabízí využít právě principal přepínače, které budou sloužit jako synchronizátoři systému.

Jak daný proces vypadá si můžete prohlédnout na obrázku 4.1. V podstatě jde jen o to, že pro úspěšný login je potřeba, aby přepínače, které ho provádějí oba vlastnili zámek. Na obrázku je znázorněna situace, kdy dochází ke spojení dvou fabrik. Pokud se ovšem přidává redundantní linka je potřeba tento případ ošetřit, jinak by došlo k livelocku. Dále je také potřeba dodat, že zvolení nového principal přepínače uvolní alokovaný zámek.

4.3.1 Doménový konflikt

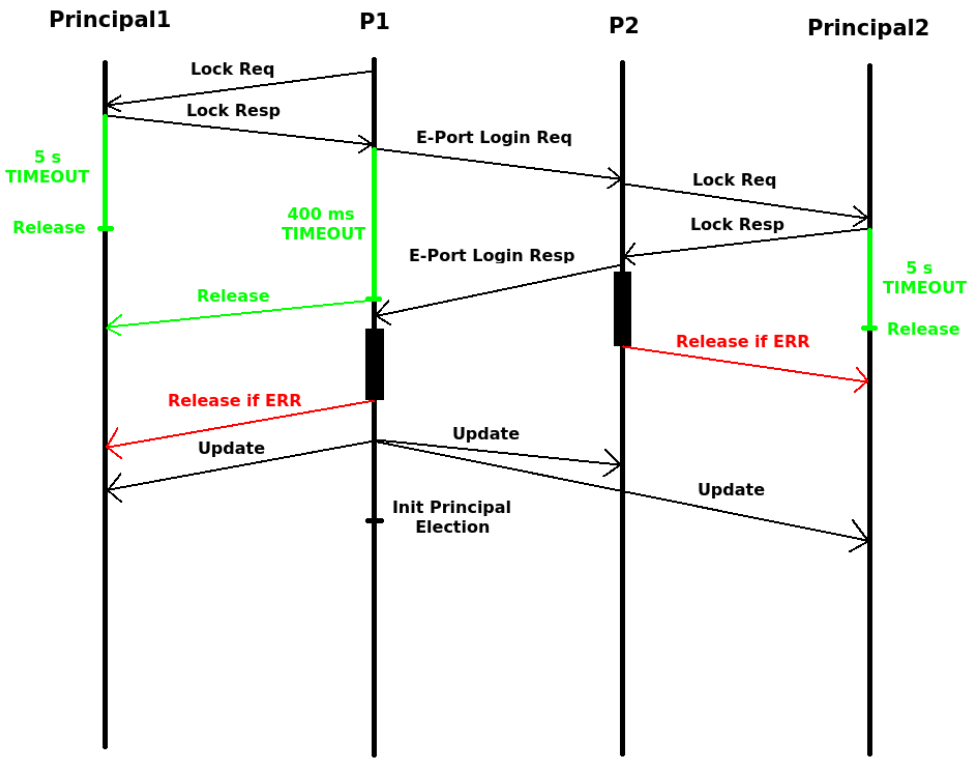
Pro úspěšný E_Port Login je potřeba, aby spojované přepínače či fabriky měli unikátní doménové identifikátory, které se nepřekrývají. Pokud tato podmínka není splněna, tak port na kterém se snaží navázat spojení přejde do chybového stavu a nesnaží se dále navazovat nové spojení. Aby port začal opět navazovat spojení je potřeba ho vypnout a znovu zapnout.

4.3.2 Zónovací konflikt

Další nutnou podmínkou navázání spojení je kompatibilní zónovací konfigurace obou přepínačů (fabrik). V zásadě existují tři případy.

1. **Clean Merge** – Zónovací konfigurace jsou totožné. V tomto případě není třeba nic řešit.
2. **Propagation** – První přepínač má definovanou konfiguraci a druhý ne. V tomto případě se konfigurace prvního přepínače propaguje do fabriky druhého přepínače.
3. **Merge** – Oba přepínače mají definované konfigurace a obě konfigurace obsahují jedinečná jména aliasů, zón a konfigurací. V tomto případě dojde ke sloučení konfigurací a propagují se všem přepínačům v nově vzniklé fabrice.

Jakýkoli jiný případ bude mít za následek přechod portu do chybového stavu jako u konfliktu doménových identifikátorů.



Obrázek 4.1: E_Port Login

4.4 E_Port Logout

E_Port Logout je proces opačný. Oproti E_Port Login je značně jednodušší, protože zde stačí poslat logout zprávu jen jedním směrem. Přepínač, který logout zprávu poslal, rozešle do fabriky zprávu o ztrátě linky (LSU Link Down). To samé udělá přepínač, který logout zprávu obdržel. Logout zpráva se posílá např. při vypnutí portu pomocí příkazu portdisable. Může tak dojít k rozpadu jedné fabriky do dvou samostatných fabrik.

4.5 Neočekávaný Logout

Krom korektního odhlášení ze systému je implementován mechanismus, který detekuje neočekávanou ztrátu přepínače či linky. Jde o jednoduché posílání HELLO zpráv mezi porty sousedních přepínačů. Pokud rozdíl aktuálního času a času poslední poslední HELLO zprávy na daném portu přesáhne určitou mez, je spojení mezi porty považováno za ukončené a do fabriky se pošle zpráva o ztrátě linky.

4.6 Rozhraní pro koordinátora simulace

Každý přepínač při svém spuštění zaregistruje handler, který slouží k přijímání příkazů od koordinátora simulace. Příkazy slouží k inicializaci či „fyzické“ konfiguraci. Tento handler reaguje pouze na UDP pakety od koordinátora simulace, který pomocí nich definuje fyzickou topologii sítě.

4.6.1 Create

Vytvoří potřebné konfigurační soubory a inicializuje přepínač.

Parametry

1. Jméno přepínače
2. IP adresa přepínače
3. UDP port přepínače
4. Počet fyzických portů přepínače

4.6.2 Run

Pouze inicializuje přepínač. Konfigurační soubory musí existovat.

Parametry

1. IP adresa přepínače
2. UDP port přepínače

4.6.3 Connect

Na lokálním FC portu nastaví fyzický stav na enabled a ostatní potřebné atributy.

Parametry

1. Číslo lokálního FC portu
2. Číslo vzdáleného FC portu
3. IP adresa vzdáleného zařízení
4. UDP port vzdáleného zařízení

4.6.4 Disconnect

Na lokálním FC portu nastaví fyzický stav na disabled a nastaví potřebné atributy na výchozí hodnoty.

Parametry

1. Číslo lokálního FC portu

4.6.5 Start

Zapne přepínač, takže se začne navazovat spojení na portech, které mají zapojené fyzické vlákno.

4.6.6 Stop

Vypne přepínač. Součástí vypnutí přepínače je poslaní logout zprávy na všechny aktivní porty.

4.6.7 Exit

Ukončí program. Zde se logout zpráva neposílá.

4.6.8 Destroy

Vymaže všechny vytvořené konfigurační soubory.

4.7 Implementace základních služeb

Budeme implementovat dvě základní služby, které slouží pro připojení serveru či storage.

4.7.1 Fabric Login

Fabric Login je ta úplně nejzákladnější služba. Každý přepínač poslouchá na protokolu, který slouží pro tuto službu.

Přepínač očekává, že mu zařízení pošle následující informace:

- Zdrojový index portu, ze kterého zařízení komunikuje.
- Cílový index portu, na který zařízení posílá informace.
- Své WWNN.
- Své WWPN.
- Své jméno.

Pokud přepínač vyhodnotí Fabric Login jako úspěšný, pošle zpět informace, které si naopak žádá zařízení a FCID adresu, která je zařízení přiřazena. Dále přepínač přidá informace o novém zařízení do svého lokálního Name Serveru a rozešle do fabriky informace o nově přidaném zařízení.

4.7.2 Directory Server

Jakmile se provede Fabric Login, tak se zařízení může přepínače dotazovat na Directory Server (Name Server). Odpověď obsahuje seznam viditelných zařízení s ohledem na nastavení zónovací konfigurace. Mimo informací týkajících se FC, vracíme zařízení i IP adresu a UDP port dostupných zařízení. Komunikace mezi serverem a storage tedy neprobíhá přes FC fabriku, ale na přímo.

Algoritmus vyhodnocení dotazu

1. Zjistíme D,I zařízení, které se dotazuje.
2. Expandujeme celou zónovací konfiguraci tak, aby nám zůstali pouze jména zón a dvojice D,I.
3. Zjistíme v jaké zóně, nebo zónách se nachází zařízení, které se dotazuje.
4. Vrátíme informace o všech zařízeních, které se nacházejí ve stejných zónách.

4.8 Implementované příkazy

Ze 165 příkazů, které jsem obdržel od vedoucího práce jsem se jich rozhodl implementovat 45. Vybral jsem ty nejzákladnější příkazy, které se používají ve většině případů při konfiguraci FC fabriky.

Dodaný framework nepočítá se všemi možnostmi, které nabízí opravdové rozhraní přepínače. Proto se bude naše syntaxe od originální malinko lišit. Celkově mi připadá, že díky tomu příkazové rozhraní působí konzistentněji. Na podstatě příkazů to ovšem nic nemění.

4.8.1 switchenable

Zapne přepínač. V důsledku tohoto příkazu se provede portenable na všech portech přepínače.

Originální syntax

```
switchenable
```

Reálný syntax

```
switchenable
```

Parametry

Žádné.

4.8.2 switchdisable

Vypne přepínač. V důsledku tohoto příkazu se provede portdisable na všech portech přepínače.

Originální syntax

```
switchdisable
```

Reálný syntax

```
switchdisable
```

Parametry

Žádné.

4.8.3 portenable

Zapne port na přepínači. Na tomto portu se poté začne navazovat spojení, pokud je připojeno fyzické vlákno. Tento příkaz nelze použít pokud je přepínač vypnutý.

Originální syntax

```
portenable portnumber
```

Reálný syntax

```
portenable portnumber
```

Parametry

- **portnumber** – Číslo FC portu. Porty jsou indexovány od 0.

4.8.4 portdisable

Vypne port na přepínači. Tento příkaz způsobí odeslání logout zprávy na daném portu, na kterou poté připojený přepínač nebo zařízení adekvátně reaguje.

Originální syntax

```
portdisable portnumber
```

Reálný syntax

```
portdisable portnumber
```

Parametry

- **portnumber** – Číslo FC portu. Stejně jako u portenable.

4.8.5 aliadd

Přidá člena nebo členy do aliasu. Alias musí existovat a redundantní členové se ignorují.

Originální syntax

```
aliadd "aliName","member[; member...]"
```

Reálný syntax

```
aliadd "aliName" "member[; member...]"
```

Parametry

- **aliName** – Jméno aliasu.
- **member(s)** – Člen nebo seznam členů oddělených středníkem a uzavřených v dvojitých uvozovkách. Členové mohou být specifikování dvěma způsoby.
  - **D,I** – Kde D je doménové ID přepínače a I je index portu na přepínači.
  - **WWN** – WWN zařízení.

4.8.6 alicreate

Vytvoří alias s jedním nebo více členy. Alias nesmí existovat.

Originální syntax

```
alicreate "aliName","member[; member...]"
```

Reálný syntax

```
alicreate "aliName" "member[; member...]"
```

Parametry

Stejné jako u aliadd.

4.8.7 aliremove

Odstraní člena nebo členy z aliasu. Neexistující členové se ignorují.

Originální syntax

```
aliremove "aliName", "member[; member...]"
```

Reálný syntax

```
aliremove "aliName" "member[; member...]"
```

Parametry

Stejné jako u aliadd.

4.8.8 alidelete

Smaže alias se všemi jeho členy. Alias musí existovat.

Originální syntax

```
alidelete "aliName"
```

Reálný syntax

```
alidelete "aliName"
```

Parametry

- **aliName** – Jméno aliasu.

4.8.9 alishow

Vypíše informace o definovaných aliasích, které odpovídají zadanému výrazu (pattern). Jinak vypíše informace o celé definované konfiguraci.

Originální syntax

```
alishow ["pattern"][,mode]
```

Reálný syntax

```
alishow ["pattern"] [mode]
```

Parametry

- **pattern** – Posixový výraz se standardníma wildcards. Vyraz může obsahovat speciální znaky.
  - **?** – Odpovídá jednomu libovolnému znaku.
  - ***** – Odpovídá libovolnému řetězci znaků.
  - [] – Odpovídá libovolnému znaku spadající do daného rozsahu (např. [a-z]).
- **mode** – Tento parametr může nabývat dvou hodnot. Výchozí hodnota je 0.
  - **0** – Pro výpis použije informace z transakčního bufferu.
  - **1** – Pro výpis použije aktuální konfiguraci, která je uložena v nevolatilní paměti.

4.8.10 zoneadd

Přidá člena nebo členy do zóny. Zóna musí existovat a redundantní členové se ignorují.

Originální syntax

```
zoneadd "zoneName", "member[; member...]"
```

Reálný syntax

```
zoneadd "zoneName" "member[; member...]"
```

Parametry

- **zoneName** – Jméno zóny.
- **member(s)** – Člen nebo seznam členů oddělených středníkem a uzavřených v dvojitých uvozovkách. Členové mohou být specifikování třemi způsoby.
  - **D,I** – Kde D je doménové ID přepínače a I je index portu na přepínači.
  - **WWN** – WWN zařízení.
  - **Alias** – Definovaný alias.

4.8.11 zonecreate

Vytvoří zónu s jedním nebo více členy. Zóna nesmí existovat.

Originální syntax

```
zonecreate "zoneName", "member[; member...]"
```

Reálný syntax

```
zonecreate "zoneName" "member[; member...]"
```

Parametry

Stejné jako u zoneadd.

4.8.12 zoneremove

Odstraní člena nebo členy ze zóny. Neexistující členové se ignorují.

Originální syntax

```
zoneremove "zoneName", "member[; member...]"
```

Reálný syntax

```
zoneremove "zoneName" "member[; member...]"
```

Parametry

Stejné jako u zoneadd.

4.8.13 zonedelete

Smaže zónu se všemi jejími členy. Zóna musí existovat.

Originální syntax

```
zonedelete "zoneName"
```

Reálný syntax

```
zonedelete "zoneName"
```

Parametry

- **zoneName** – Jméno zóny.

4.8.14 zoneshow

Vypíše informace o definovaných zónách, které odpovídají zadanému výrazu (pattern). Jinak vypíše informace o celé definované konfiguraci.

Originální syntax

```
zoneshow ["pattern"][,mode]
```

Reálný syntax

```
zoneshow ["pattern"] [mode]
```

Parametry

Stejné jako u alishow.

4.8.15 cfgadd

Přidá člena nebo členy do konfigurace. Konfigurace musí existovat, redundantní členové se ignorují a členové by měli být typu zóna. Nicméně kontrola správného typu se v tomto příkazu neprovádí.

Originální syntax

```
cfgadd "cfgName", "member[; member]"
```

Reálný syntax

```
cfgadd "cfgName" "member[; member]"
```

Parametry

- **cfgName** – Jméno konfigurace.
- **member(s)** – Člen nebo seznam členů oddělených středníkem a uzavřených v dvojitých uvozovkách. Členové mohou být specifikování jedním způsobem.
  - **Zóna** – Definovaná zóna.

4.8.16 cfgcreate

Vytvoří konfiguraci s jedním nebo více členy. Konfigurace nesmí existovat.

Originální syntax

```
cfgcreate "cfgName", "member[; member]"
```

Reálný syntax

```
cfgcreate "cfgName" "member[; member]"
```

Parametry

Stejné jako u cfgadd.

4.8.17 cfgremove

Odstraní člena nebo členy z konfigurace. Neexistující členové se ignorují.

Originální syntax

```
cfgremove "cfgName", "member[; member...]"
```

Reálný syntax

```
cfgremove "cfgName" "member[; member...]"
```

Parametry

Stejné jako u cfgadd.

4.8.18 cfgdelete

Smaže konfiguraci se všemi jejími členy. Konfigurace musí existovat.

Originální syntax

```
cfgdelete "cfgName"
```

Reálný syntax

```
cfgdelete "cfgName"
```

Parametry

- **cfgName** – Jméno konfigurace.

4.8.19 cfgshow

Vypíše informace o definovaných konfiguracích, které odpovídají zadanému výrazu (pattern). Jinak vypíše informace o celé definované konfiguraci.

Originální syntax

```
zoneshow ["pattern"][,mode]
```

Reálný syntax

```
zoneshow ["pattern"] [mode]
```

Parametry

Stejné jako u alishow.

4.8.20 cfgactvshow

Vypíše informace o aktivní konfiguraci. Aktivní konfigurace může být pouze jedna. Definované aliasy se expandují na skutečné hodnoty (WWN nebo D,I).

Originální syntax

```
cfgactvshow
```

Reálný syntax

```
cfgactvshow
```

Parametry

Žádné.

4.8.21 cfgsave

Uloží obsah zónovací konfigurace z transakčního bufferu do nevolatilní paměti a danou konfiguraci propaguje všem přepínačům ve fabrice. Je nutné kontaktovat principal přepínač a vynutit si výlučný přístup. Pokud nedostaneme odpověď, příkaz nelze provést a uživatel dostane chybovou hlášku. Dále tento příkaz provádí kontrolu zda-li je konfigurace validní.

Originální syntax

```
cfgsave
```

Reálný syntax

```
cfgsave
```

Parametry

Žádné.

4.8.22 cfgenable

Funguje stejně jako cfgsave, ale navíc aktivuje předanou konfiguraci.

Originální syntax

```
cfgenable "cfgName"
```

Reálný syntax

```
cfgenable "cfgName"
```

Parametry

- **cfgName** – Jméno konfigurace.

4.8.23 cfgdisable

Opak cfgenable. Všem přepínačům ve fabrice se pošle zpráva, že mají deaktivovat aktivní konfiguraci.

Originální syntax

```
cfgdisable
```

Reálný syntax

```
cfgdisable
```

Parametry

Žádné.

4.8.24 cfgclear

Vymaže všechny zónovací informace z transakčního bufferu.

Originální syntax

```
cfgclear
```

Reálný syntax

```
cfgclear
```

Parametry

Žádné.

4.8.25 defzone

Nastaví chování, pokud není aktivní žádná zónovací konfigurace. Tato změna se opět pouze zapíše do transakčního bufferu.

Originální syntax

```
defzone [--noaccess | --allaccess | --show]
```

Reálný syntax

```
defzone --noaccess | --allaccess | --show
```

Parametry

- **noaccess** – Nikdo nevidí nikoho.
- **allaccess** – Všichni vidí všechny.
- **show** – Pouze vypíše informace o výchozím chování.

Dané parametry se vzájemně vylučují.

4.8.26 cfgtransshow

Zobrazí informace o aktuální zónovací transakci.

Originální syntax

```
cfgtransshow
```

Reálný syntax

```
cfgtransshow
```

Parametry

Žádné.

4.8.27 cfgtransabort

Zruší zónovací transakci.

Originální syntax

```
cfgtransabort [token]
```

Reálný syntax

```
cfgtransabort
```

Parametry

- **token** – Nepovinný parametr, kterým je možné specifikovat transakci, kterou chceme zrušit. Nicméně jsem nikde nenašel, že by šlo na jednom přepínači mít více než jednu transakci, takže jsem se rozhodl daný parametr neimplementovat.

4.8.28 configupload

Jedná se o velice komplexní příkaz, který umí uložit celkovou konfiguraci přepínače do různých cílů, pomocí různých protokolů (např. scp, ftp). Celkovou konfigurací je myšlen soubor, pomocí kterého se přepínač inicializuje při spuštění. Je implementována velice strohá verze tohoto příkazu.

Originální syntax

```
configupload
configupload [-all] [-p ftp | -ftp] ["host","user" ,"path"[,"passwd"]]
configupload [-all] [-p scp | -scp] ["host","user" ,"path"]
configupload [-all] [-force] [-local|-USB|-U] ["file"]
configupload [-fid FID | -chassis | -all]
             [-p ftp | -ftp] ["host","user" ,"path"[,"passwd"]]
configupload [-fid FID | -chassis | -all]
             [-p scp | -scp] ["host","user" ,"path"]
configupload [-fid FID | -chassis | -all]
             [-force] [-local|-USB|-U] ["file"]
```

Reálný syntax

```
configupload [--force] --local "file"
```

Parametry

Jsou podporovány pouze následující parametry:

- **local** – V reálu to znamená, že konfigurace se nahraje do lokálního filesystému daného přepínače. Pro nás to znamená, že se konfigurace uloží do složky „switchDirectory/configs“.
- **file** – Jméno, pod kterým chceme konfiguraci uložit.
- **force** – Přepíše existující soubor bez varování.

4.8.29 configdownload

Jde o doplněk předchozího příkazu, který naopak danou konfiguraci stáhne a nahraje. Aby bylo možné konfiguraci stáhnout, je nejdříve potřeba přepínač vypnout pomocí switchdisable.

Originální syntax

```
configdownload
configdownload [-all] [-p ftp | -ftp] ["host","user","path"[,"passwd"]]
configdownload [-all] [-p scp | -scp] ["host","user","path"]
configdownload [-all] [-local|-USB|-U] ["file"]
configdownload [-fid FID [-sfid FID] | -chassis | -all]
               [-p ftp | -ftp] ["host","user","path"[,"passwd"]]
configdownload [-fid FID [-sfid FID] | -chassis | -all]
               [-p scp | -scp] ["host","user","path"]
configdownload [-fid FID [-sfid FID] | -chassis | -all]
               [-local|-USB|-U] ["file"]
```

Reálný syntax

```
configdownload --local "file"
```

Parametry

Jsou podporovány pouze následující parametry:

- **local** – Stejné jako v předchozím případě.
- **file** – Jméno konfigurace, kterou chceme stáhnout.

4.8.30 configlist

Vypíše seznam konfigurací uložených lokálně nebo na USB. Opět je implementována jen varianta pro lokálně uložené konfigurace.

Originální syntax

```
configlist -local|-USB|-U
```

Reálný syntax

```
configlist --local
```

Parametry

- **local** – Stejné jako v předchozím případě.

4.8.31 configremove

Odstraní konfiguraci uloženou lokálně nebo na USB.

Originální syntax

```
configremove -local|-USB|-U [file]
```

Reálný syntax

```
configremove --local "file"
```

Parametry

- **local** – Stejné jako v předchozím případě.
- **file** – Název konfiguračního souboru, který chceme odstranit.

4.8.32 configdefault

Obnoví konfiguraci přepínače na výchozí hodnoty. Přepínač musí být vypnutý pomocí switchdisable.

Originální syntax

```
configDefault [-all]
configDefault [-fid FID | -chassis |  -all ]
```

Reálný syntax

```
configDefault
```

Parametry

Žádný z uvedených parametrů není implementován.

4.8.33 configshow

Vypíše obsah konfiguračního souboru.

Originální syntax

```
configshow
configshow -pattern "pattern"
configshow [-all | -fid FID | -chassis] [-local|-USB|-U] [file]
           [-pattern "pattern"]
```

Reálný syntax

```
configshow --local "file" [--pattern "pattern"]
```

Parametry

- **local** – Stejné jako v předchozím případě.
- **file** – Název konfiguračního souboru, který chceme vypsat.
- **pattern** – Posixový výraz, kterým můžeme filtrovat výstup příkazu.

4.8.34 fabricshow

Zobrazí informace o fabrice. To znamená jaké přepínače se v ní nachází, jejich FCID, WWN, jména a kdo je principal přepínač.

Originální syntax

```
fabricshow
```

Reálný syntax

```
fabricshow
```

Parametry

Žádné.

4.8.35 switchshow

Zobrazí informace o přepínači a jeho portech.

Originální syntax

```
switchshow [-portcount | -iscsi]
```

Reálný syntax

```
switchshow [--portcount]
```

Parametry

- **portcount** – Zobrazí pouze počet portů přepínače.
- **iscsi** – Zobrazí informace o iscsi protokolu (není implementováno).

4.8.36 portshow

Zobrazí informace o konkrétním portu přepínače.

Originální syntax

```
portshow port
portshow [options] [slot/][ge]port [args] [optional_args]
```

Reálný syntax

```
portshow port
```

Parametry

- **port** – Index portu, který chceme zobrazit.

Další parametry nejsou implementovány. Jde o různé informace o protokolech či hardwarové statistiky portu, které pro nás nejsou důležité.

4.8.37 nsshow

Zobrazí informace lokálního Name Serveru, tzn. informace o připojených N_Portech (servery, storage).

Originální syntax

```
nsshow [ -r -t ]
```

Reálný syntax

```
nsshow
```

Parametry

- **r** – Vypisuje navíc informaci o jakém druhu změn má být zařízení informováno. Tuto funkcionalitu do detailu neimplementujeme, takže to pro nás není zajímavé.
- **t** – Vypisuje navíc jestli je připojené zařízení fyzické či virtuální. V našem případě jsou všechna zařízení fyzická, takže daný přepínač není implementován.

4.8.38 nscamshow

Zobrazí informace o vzdálených zařízeních, které jsou v cachi Name Serveru. Pro nás to znamená, informace o všech ostatních zařízeních, kromě lokálně připojených.

Originální syntax

```
nscamhow [ -t ]
```

Reálný syntax

```
nscamhow
```

Parametry

- **t** – Stejný případ jako u nsshow.

4.8.39 nsallshow

Zobrazí informace globálního Name Serveru, tzn. informace o všech zařízeních připojených do fabriky. Tento výpis ovšem není tak detailní jako u nsshow nebo nscamshow.

Originální syntax

```
nsallshow [type]
```

Reálný syntax

```
nsallshow [type]
```

Parametry

- **type** – Jediné validní hodnoty jsou:
  - **8** – Zobrazí zařízeni pracující nad FCP protokolem (výchozí hodnota).
  - **4, 5** – Zobrazí zařízeni pracující nad FC-IP protokolem. Tento typ zařízení ovšem neimplementujeme.

4.8.40 switchname

Zobrazí nebo nastaví jméno přepínače.

Originální syntax

```
switchname [ name ]
```

Reálný syntax

```
switchname [ name ]
```

Parametry

- **name** – Nové jméno přepínače.

4.8.41 portname

Zobrazí nebo nastaví jméno portu.

Originální syntax

```
portname portnumber [name]
```

Reálný syntax

```
portname portnumber [name]
```

Parametry

- **portnumber** – Index portu.
- **name** – Nové jméno portu.

4.8.42 wwn

Zobrazí WWN přepínače.

Originální syntax

```
wwn [-sn]
```

Reálný syntax

```
wwn
```

Parametry

- **sn** – Zobrazí navíc sériové číslo přepínače (není implementováno).

4.8.43 h

Zobrazí historii dvaceti posledních příkazů.

Originální syntax

```
h
history
```

Reálný syntax

```
h
history
```

Parametry

Žádné.

4.8.44 help

Zobrazí nápovědu k příkazu. Pokud není příkaz specifikován vypíše se seznam dostupných příkazů.

Originální syntax

```
help [ command ]
```

Reálný syntax

```
help [ command ]
```

Parametry

- **command** – Příkaz, ke kterému chceme nápovědu.

4.8.45 domain

Jde o nestandardní příkaz, kterým lze jednoduše měnit doménový identifikátor přepínače. Přepínač musí být vypnutý. Standardně se k tomu používá příkaz configure, ten je ovšem pro naše účely zbytečně složitý.

Originální syntax

```
configure
```

Reálný syntax

```
domain ID
```

Parametry

- **ID** – Doménový identifikátor, který chceme nastavit (1–239).

4.9 Logování

Každý přepínač má ve svém adresáři soubor log.txt, do kterého loguje nejrůznější události. Pokud simulaci provádíme lokálně na jednom počítači, můžeme log unifikovat např. pomocí příkazu `tail -qf swDir1/log.txt swDir2/log.txt...` Takto unifikovaný log vypadá následovně:

```
[2019-04-04 18:15:42] Switch #04 - Sending ADD ISL request to UDP port #5005, ports: #6 -> #6
[2019-04-04 18:15:42] Switch #05 - ADD ISL request received from UDP port #5004, ports: #6 -> #6
[2019-04-04 18:15:42] Switch #05 - Sending ADD ISL response to UDP port #5004, ports: #6 -> #6
[2019-04-04 18:15:42] Switch #04 - ADD ISL response received from UDP port #5005, ports: #6 -> #6
[2019-04-04 18:15:42] Switch #04 - Two fabrics needs to be merged
[2019-04-04 18:15:42] Switch #04 - Initializing Principal election
[2019-04-04 18:15:44] Switch #06 - New principal elected -> Id: 6
[2019-04-04 18:15:44] Switch #05 - New principal elected -> Id: 6
[2019-04-04 18:15:44] Switch #03 - New principal elected -> Id: 6
[2019-04-04 18:15:44] Switch #01 - New principal elected -> Id: 6
[2019-04-04 18:15:44] Switch #07 - New principal elected -> Id: 6
[2019-04-04 18:15:44] Switch #02 - New principal elected -> Id: 6
[2019-04-04 18:15:44] Switch #04 - New principal elected -> Id: 6
```

Kapitola 5

Testování

Testování proběhlo převážně ručně. V budoucnu má být ve frameworku implementován příkaz `@file`, který bude umět spouštět dávky příkazů ze souboru.

5.1 Testování konzistence

Tento test zjišťuje zdali při formování fabriky nedochází k nekonzistenci jednotlivých přepínačů. To znamená, že každý přepínač musí mít stejný pohled na fabriku.

Test proběhl následovně:

1. Vytvoříme sedm přepínačů a nedefinujeme např. následující topologii (dvě kružnice po třech uzlech spojené přes jeden uzel).

```
create switch sw1 8
create switch sw2 8
create switch sw3 8
create switch sw4 8
create switch sw5 8
create switch sw6 8
create switch sw7 8

connect sw1 0 sw2 0
connect sw2 1 sw3 1
connect sw1 2 sw3 2

connect sw5 0 sw6 0
connect sw6 1 sw7 1
connect sw5 2 sw7 2

connect sw3 4 sw4 4
connect sw5 5 sw4 5
```

2. Pomocí příkazu `domain` na přepínačích změníme doménová ID, aby nedošlo ke konfliktu.
3. Pomocí příkazu `start /all` v řídící konzoli spustíme všechny přepínače.

Tento test proběhl desetkrát. Fabrika se vždy zformovala malinko odlišným způsobem, ale vždy skončila v konzistentním stavu. Stav fabriky jsem kontroloval pomocí příkazu `fabricshow`.

```
sw1:admin> fabricshow
Switch ID   Worldwide Name          Enet IP Addr    FC IP Addr      Name
-------------------------------------------------------------------------
  1: 010000 76:aa:fd:d0:b2:68:23:c5 0.0.0.0         0.0.0.0          "sw1"
  2: 020000 6a:60:bb:b2:bf:eb:c2:c0 0.0.0.0         0.0.0.0         >"sw2"
  3: 030000 e9:54:63:17:0a:56:c2:a5 0.0.0.0         0.0.0.0          "sw3"
  4: 040000 fa:1b:45:63:ea:a8:f6:1c 0.0.0.0         0.0.0.0          "sw4"
  5: 050000 fa:fe:97:c7:de:16:27:77 0.0.0.0         0.0.0.0          "sw5"
  6: 060000 8d:53:2e:37:1e:1c:c9:d1 0.0.0.0         0.0.0.0          "sw6"
  7: 070000 d0:4f:d0:c5:e4:9c:87:a9 0.0.0.0         0.0.0.0          "sw7"

The Fabric has 7 switches
```

5.2 Testování příkazů

Testování příkazů volně navazuje na předchozí test. Po zformování fabriky o sedmi přepínačích postupně budeme testovat jednotlivé příkazy a sledovat zda jsou v souladu s očekávaným chováním. Příkazy jsou psané i s promptem, aby bylo jasné, na kterém přepínači se příkaz spouští.

- `sw4:admin> switchdisable`

  ```
  sw1:admin> fabricshow
  Switch ID   Worldwide Name          Enet IP Addr    FC IP Addr      Name
  -------------------------------------------------------------------------
    1: 010000 76:aa:fd:d0:b2:68:23:c5 0.0.0.0         0.0.0.0          "sw1"
    2: 020000 6a:60:bb:b2:bf:eb:c2:c0 0.0.0.0         0.0.0.0         >"sw2"
    3: 030000 e9:54:63:17:0a:56:c2:a5 0.0.0.0         0.0.0.0          "sw3"

  The Fabric has 3 switches

  sw5:admin> fabricshow
  Switch ID   Worldwide Name          Enet IP Addr    FC IP Addr      Name
  -------------------------------------------------------------------------
    5: 050000 fa:fe:97:c7:de:16:27:77 0.0.0.0         0.0.0.0          "sw5"
    6: 060000 8d:53:2e:37:1e:1c:c9:d1 0.0.0.0         0.0.0.0         >"sw6"
    7: 070000 d0:4f:d0:c5:e4:9c:87:a9 0.0.0.0         0.0.0.0          "sw7"

  The Fabric has 3 switches
  ```

  Způsobí rozpad na dvě samostatné fabriky tvořené přepínači sw1, sw2, sw3 a sw5, sw6, sw7.

- ```
  sw4:admin> alicreate "ali1" "00:11:22:33:44:55:66:77; 1,1"
  sw4:admin> cfgsave
  ```

  Vytvoří alias ali1 v transakčním bufferu, poté potvrdí transakci a uloží změny do konfiguračního souboru.

- ```
  sw4:admin> switchenable
  ```

  ```
  sw1:admin> fabricshow
  Switch ID   Worldwide Name          Enet IP Addr    FC IP Addr      Name
  -------------------------------------------------------------------------
    1: 010000 76:aa:fd:d0:b2:68:23:c5 0.0.0.0         0.0.0.0         "sw1"
    2: 020000 6a:60:bb:b2:bf:eb:c2:c0 0.0.0.0         0.0.0.0        >"sw2"
    3: 030000 e9:54:63:17:0a:56:c2:a5 0.0.0.0         0.0.0.0         "sw3"
    4: 040000 fa:1b:45:63:ea:a8:f6:1c 0.0.0.0         0.0.0.0         "sw4"
    5: 050000 fa:fe:97:c7:de:16:27:77 0.0.0.0         0.0.0.0         "sw5"
    6: 060000 8d:53:2e:37:1e:1c:c9:d1 0.0.0.0         0.0.0.0         "sw6"
    7: 070000 d0:4f:d0:c5:e4:9c:87:a9 0.0.0.0         0.0.0.0         "sw7"

  The Fabric has 7 switches

  sw1:admin> cfgshow
  Defined configuration:
   alias: ali1    00:11:22:33:44:55:66:77; 1,1

  Effective configuration:
  No Effective configuration: (No Access)

  sw7:admin> fabricshow
  Switch ID   Worldwide Name          Enet IP Addr    FC IP Addr      Name
  -------------------------------------------------------------------------
    1: 010000 76:aa:fd:d0:b2:68:23:c5 0.0.0.0         0.0.0.0         "sw1"
    2: 020000 6a:60:bb:b2:bf:eb:c2:c0 0.0.0.0         0.0.0.0        >"sw2"
    3: 030000 e9:54:63:17:0a:56:c2:a5 0.0.0.0         0.0.0.0         "sw3"
    4: 040000 fa:1b:45:63:ea:a8:f6:1c 0.0.0.0         0.0.0.0         "sw4"
    5: 050000 fa:fe:97:c7:de:16:27:77 0.0.0.0         0.0.0.0         "sw5"
    6: 060000 8d:53:2e:37:1e:1c:c9:d1 0.0.0.0         0.0.0.0         "sw6"
    7: 070000 d0:4f:d0:c5:e4:9c:87:a9 0.0.0.0         0.0.0.0         "sw7"

  The Fabric has 7 switches

  sw7:admin> cfgshow
  Defined configuration:
   alias: ali1    00:11:22:33:44:55:66:77; 1,1

  Effective configuration:
  No Effective configuration: (No Access)
  ```

  Způsobí zformování fabriky o sedmi přepínačích. Ostatní přepínače nyní mají v konfiguraci uložený alias ali1.

- ```
  sw4:admin> switchdisable
  sw4:admin> domain 1
  sw4:admin> switchenable

  sw3:admin> fabricshow
  Switch ID   Worldwide Name          Enet IP Addr    FC IP Addr      Name
  -------------------------------------------------------------------------
    1: 010000 76:aa:fd:d0:b2:68:23:c5 0.0.0.0         0.0.0.0         "sw1"
    2: 020000 6a:60:bb:b2:bf:eb:c2:c0 0.0.0.0         0.0.0.0        >"sw2"
    3: 030000 e9:54:63:17:0a:56:c2:a5 0.0.0.0         0.0.0.0         "sw3"

  The Fabric has 3 switches
  sw3:admin> switchshow

  switchName:     sw3
  switchType:     71.2
  switchState:    Online
  switchMode:     Native
  switchRole:     Subordinate
  switchDomain:   3
  switchId:       030000
  switchWwn:      e9:54:63:17:0a:56:c2:a5
  zoning:         OFF
  switchBeacon:   OFF
  HIF Mode:       OFF

  Index Port Address Media Speed       State   Proto
  ==================================================
     0   0   030100   id    N8       No_Light    FC
     1   1   030200   id    N8       Online      FC  E-Port  6a:60:bb:b2:bf:eb:c2:c0 "sw2"
     2   2   030300   id    N8       Online      FC  E-Port  76:aa:fd:d0:b2:68:23:c5 "sw1"
     3   3   030400   id    N8       No_Light    FC
     4   4   030500   id    N8       Online      FC  E-Port  segmented,(domain overlap)
     5   5   030600   id    N8       No_Light    FC
     6   6   030700   id    N8       No_Light    FC
     7   7   030800   id    N8       No_Light    FC

  sw7:admin> fabricshow
  Switch ID   Worldwide Name          Enet IP Addr    FC IP Addr      Name
  -------------------------------------------------------------------------
    1: 010000 fa:1b:45:63:ea:a8:f6:1c 0.0.0.0         0.0.0.0         "sw4"
    5: 050000 fa:fe:97:c7:de:16:27:77 0.0.0.0         0.0.0.0         "sw5"
    6: 060000 8d:53:2e:37:1e:1c:c9:d1 0.0.0.0         0.0.0.0        >"sw6"
    7: 070000 d0:4f:d0:c5:e4:9c:87:a9 0.0.0.0         0.0.0.0         "sw7"

  The Fabric has 4 switches
  ```

  Tato sekvence příkazů způsobí, že nedojde ke zformování fabriky s přepínači sw1, sw2, sw3 kvůli překryvu doménových identifikátorů.
  ```

- ```
  sw4:admin> switchdisable
  sw4:admin> domain 4
  sw4:admin> aliadd "ali1" "2,2"
  sw4:admin> cfgsave
  sw4:admin> switchenable

  sw4:admin> fabricshow
  Switch ID   Worldwide Name          Enet IP Addr    FC IP Addr      Name
  -------------------------------------------------------------------------
    4: 040000 fa:1b:45:63:ea:a8:f6:1c 0.0.0.0         0.0.0.0        >"sw4"

  The Fabric has 1 switches

  sw5:admin> switchshow
  switchName:     sw5
  switchType:     71.2
  switchState:    Online
  switchMode:     Native
  switchRole:     Subordinate
  switchDomain:   5
  switchId:       050000
  switchWwn:      fa:fe:97:c7:de:16:27:77
  zoning:         OFF
  switchBeacon:   OFF
  HIF Mode:       OFF

  Index Port Address Media Speed       State   Proto
  ==================================================
     0   0   050100   id    N8       Online      FC  E-Port  8d:53:2e:37:1e:1c:c9:d1 "sw6"
     1   1   050200   id    N8       No_Light    FC
     2   2   050300   id    N8       Online      FC  E-Port  d0:4f:d0:c5:e4:9c:87:a9 "sw7"
     3   3   050400   id    N8       No_Light    FC
     4   4   050500   id    N8       No_Light    FC
     5   5   050600   id    N8       Online      FC  E-Port  segmented,(zone conflict)
     6   6   050700   id    N8       No_Light    FC
     7   7   050800   id    N8       No_Light    FC
  ```

  Tato sekvence příkazů způsobí, že nedojde ke zformování fabriky s ostatními přepínači kvůli konfliktu zónovací konfigurace.

Všechny příkazy se chovají jak je očekáváno. Spousta podobných testů proběhla během vývoje programu. Další testování proběhne po dokončení serverové a storage části a také prakticky na předmětu BI-STO.
  ```

Závěr

Cílem této práce bylo vytvořit simulátor Fibre Channel přepínače, který bude odrážet jeho skutečné chování. Ke splnění tohoto cíle byl jako vzor vybrán přepínač od firmy Brocade, který je na fakultě dostupný. Přepínač měl nabízet ke konfiguraci rozhraní příkazové řádky a veškerá komunikace s ostatními komponentami měla být realizována pomocí protokolu UDP. K tomuto účelu posloužil framework cli dodaný vedoucím práce Ing. Jiřím Kašparem.

Důraz měl být kladen hlavně na následující funkcionalitu:

1. **Kaskádování** – Propojení více přepínačů mezi sebou. Takto propojené přepínače poté musí tvořit danou FC síť, neboli fabriku.
2. **Základní příkazy** – Implementace základních příkazů sloužících pro konfiguraci portů, aliasů a zónování.
3. **Základní služby** – Implementace základních služeb sloužících pro připojení serverů a úložišť.

Všechny tyto cíle byly splněny. Samozřejmě, že není implementováno vše do nejmenšího detailu, ale hlavní rysy chování přepínačů jsou podle mě dostatečně vystiženy. Bylo implementováno 45 příkazů, kterými je možné přepínače konfigurovat a 2 základní služby, které slouží pro připojení ostatních zařízení k dané FC síti. Na řešení této práce navazuje bakalářská práce Simulátor serveru a storage, jejíž autorem je pan Dmitrii Vekshin. Dané řešení mi přijde jako ideální způsob jak se obecně seznámit s technologií Fibre Channel i specifickým chováním přepínačů od firmy Brocade.

Literatura

[1] Rick Donato: Fibre Channel - SAN Protocols Explained. 2018, [cit. 8.2.2019]. Dostupné z: https://www.packetflow.co.uk/fibrechannel-sanprotocolsexplained/

[2] CERN: Fibre Channel topologies. [cit. 10.2.2019]. Dostupné z: http://hsi.web.cern.ch/HSI/fcs/spec/ov1.gif

[3] Brocade Communications Systems, Inc.: Fabric OS Administrator's Guide. 2016, [cit. 10.2.2019]. Dostupné z: https://www.atto.com/software/files/manuals/FabricOS_AdministratorGuide.pdf

[4] Brocade Communications Systems, Inc.: BROCADE SIMULATOR. [cit. 30.2.2019]. Dostupné z: https://community.broadcom.com/t5/Fibre-Channel-SAN-Forums/BROCADE-SIMULATOR-DOWNLOAD/td-p/93798

[5] SimSANs: Data Center Storage Networking Simulation. [cit. 30.2.2019]. Dostupné z: https://www.simsans.org/

[6] Cisco: Packet Tracer. [cit. 5.3.2019]. Dostupné z: https://www.netacad.com/courses/packet-tracer

[7] SimSANs: Switch and Fabric Port Configurations. [cit. 30.2.2019]. Dostupné z: https://www.simsans.org/images/VSD_SanSwPort.gif

[8] internap: fake-switches. [cit. 6.3.2019]. Dostupné z: https://github.com/internap/fake-switches

[9] brocade-sim: brocade-sim. [cit. 6.3.2019]. Dostupné z: https://code.google.com/archive/p/brocade-sim/

[10] dcastelob: sansimshell. [cit. 28.3.2019]. Dostupné z: https://github.com/dcastelob/sansimshell

[11] CERN: Fibre Channel Frame Structure. [cit. 11.3.2019]. Dostupné z: http://hsi.web.cern.ch/HSI/fcs/spec/ov4.gif

[12] Shlomi Elhaiani: Dijkstra's shortest path with minimum edges. [cit. 8.4.2019]. Dostupné z: https://www.geeksforgeeks.org/dijkstras-shortest-path-with-minimum-edges/

[13] GeeksforGeeks: Depth First Search or DFS for a Graph. [cit. 8.4.2019]. Dostupné z: https://www.geeksforgeeks.org/depth-first-search-or-dfs-for-a-graph/

[14] Nishu Aggarwal: Election algorithm and distributed processing. [cit. 12.4.2019]. Dostupné z: https://www.geeksforgeeks.org/election-algorithm-and-distributed-processing/

Příloha **A**

Seznam použitých zkratek

SAN Storage Area Network

FC Fibre Channel

FCP Fibre Channel Protocol

SCSI Small Computer System Interface

FCoE Fibre Channel over Ethernet

UDP User Datagram Protocol

TCP Transmission Control Protocol

IP Internet Protocol

HBA Host Bus Adapter

WWN World Wide Name

WWNN World Wide Node Name

WWPN World Wide Port Name

PID Port ID

FCID Fibre Channel ID

CLD Command Language Definition

LSU Link State Update

Příloha **B**

Obsah přiloženého média

- `readme.txt` ..... stručný popis obsahu média
- `bin` ..... adresář se spustitelnou formou implementace
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF
  - `thesis.ps` ..... text práce ve formátu PS