Czech Technical University in Prague
Faculty of Electrical Engineering

Department of Computer Science and Engineering

BACHELOR PROJECT ASSIGNMENT

Student: **Lukáš Červenka**

Study programme: Open Informatics
Specialisation: Software Systems

Title of Bachelor Project: **Design of configuration model and integration of NETCONF interface in optical circuit switch**

Guidelines:

Familiarize yourself with NETCONF protocol and YANG data model. Design a configuration model in YANG for a given CESNET optical circuit switch. Conduct an analysis of tools available for integration of a configuration interface using NETCONF. Analyze features of those tools and incorporate the most feasible one to the switch. Implement the designed YANG model and validate its functionality. Evaluate the NETCONF configuration interface and YANG model under conditions that can arise during network setup; especially test transactional processing, request atomicity and state verification. Demonstrate the created implementation under real traffic load using the switch and an external management software.

Bibliography/Sources:

Cui, Jianqun, et al. "The Design of the Network Configuration Management Based on NETCONF Protocol." Applied Informatics and Communication. Springer Berlin Heidelberg, 2011. 705-713.
Enns, R., et al. RFC 6241 - NETCONF configuration protocol. Internet Engineering Task Force, 2011.
Scott, M., et al. RFC 6022 - YANG Module for NETCONF Monitoring. Internet Engineering Task Force, 2010.

Bachelor Project Supervisor: Ing. Tomáš Hégr

Valid until the end of the summer semester of academic year 2016/2017

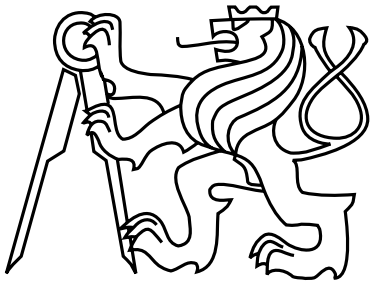
L.S.

prof. Ing. Filip Železný, Ph.D.
Head of Department

prof. Ing. Pavel Ripka, CSc.
Dean

Prague, December 22, 2015

České vysoké učení technické v Praze
Fakulta elektrotechnická
Katedra počítačů



Bakalářská práce

Návrh konfiguračního modelu a integrace NETCONF rozhraní do optického přepínače

Lukáš Červenka

Vedoucí práce: Ing. Tomáš Hégr

Studijní program: Otevřená informatika, Bakalářský

Obor: Softwarové systémy

10. ledna 2017

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne 30. 12. 2016 .....

Abstract

NETCONF protocol is a communication protocol designed by Internet Engineering Task Force (IETF). Its main purpose is to remotely configure devices and check their state. The goal of this bachelor project is to design a NETCONF interface that can be deployed to the optical circuit switch Czech Light developed by CESNET. As a result, a configuration model of the device and its implementation were created. Necessary software components were described and configured.

Abstrakt

NETCONF je komunikační protokol vytvořený skupinou Internet Engineering Task Force (IETF), který slouží ke vzdálené konfiguraci a monitorování zařízení. Úkolem této práce bylo navrhnout a nasadit NETCONF rozhraní do optického přepínače řady Czech Light vyvinutého sdružením CESNET. Výsledkem práce je konfigurační model zařízení a jeho implementace. Byl popsán výběr, instalace a konfigurace potřebných softwarových komponent.

Obsah

Seznam obrázků

Kapitola 1

Úvod

S rozvojem počítačových sítí přišla nutnost monitorovat a vzdáleně spravovat aktivní síťové prvky a servery. Dnes se k monitorování zařízení v běžných sítích používá stále především protokol SNMP, který má kořeny v roce 1988 a v mnoha ohledech již nevyhovuje současným potřebám. Jedním z pomyslných nástupců je protokol NETCONF vytvořený pracovní skupinou IETF. NETCONF přináší především standardizaci přístupu k informacím o aktuálním stavu zařízení a nastavení nezávisle na výrobci. Vyžaduje rovněž zabezpečenou komunikaci a transakční zpracování. Výrobci dnes již pomalu nacházejí cestu k protokolu NETCONF a implementují jej do svých zařízení.

Další technologií, která přišla s rozvojem sítí a rostoucími požadavky na infrastrukturu, jsou softwarově definované sítě (SDN). Na rozdíl od tradičních sítí, kde je logika řízení provozu nastavena na každém aktivním prvku zvlášť, rozhoduje o řízení toku dat v SDN centrální kontrolér, který jednotlivé aktivní prvky (zařízení na datové vrstvě) ovládá. K tomuto ovládání je nutné používat protokoly, které jsou standardizované, uznávané a implementované výrobci. Vedle protokolu OpenFlow může k přenosu nastavení v sítích SDN sloužit i protokol NETCONF.

Jedním ze zařízení, které by mohlo být použito v síti SDN, je i optický přepínač Czech Light vyvinutý sdružením CESNET. Rodina zařízení Czech Light neobsahuje jen optické přepínače, ale i optické zesilovače, konfigurovatelné DWDM multiplexery, selektivní přepínače vlnových délek či dokonce přepínače podporující optický multicast.

Pro implementaci NETCONF rozhraní do optického přepínače bylo nutné navrhnout model zařízení v jazyce YANG, naprogramovat v něm zachycenou funkcionalitu a správně ji integrovat do již existujícího projektu NETCONF serveru Netopeer.

Kapitola 2

Přehled použitých technologií

2.1 Technologie optických sítí

Experimenty s přenosem světla přes optická vlákna probíhaly již v 60. letech 20. století[13]. Postupným vývojem se neustále zlepšovaly parametry vláken, takže již v 80. letech bylo možné přenášet signály na několik desítek kilometrů. Díky tomuto došlo k masivnímu nasazení technologie optického přenosu dat do sítí elektronických komunikací.

Dnes se optická vlákna používají v mnoha oblastech. Velmi důležité je použití v sítích WAN, kde slouží k propojení vzdálených lokalit. Rovněž v sítích LAN a SAN jsou použity pro vysokokapacitní přenos dat. Běžně dosahované rychlosti jsou mezi 1 a 100 Gbps na jednom vlákně. Pro velmi krátké trasy v zastavěné oblasti se používají také zařízení pro přenos optického signálů volným prostorem bez použití kabelu - Free-space optic (FSO). Optická vlákna se používají také pro vysoce kvalitní přenos zvuku.

Obrázek 2.1: Zařízení pro přenos optického signálu volným prostorem

Výhody optických sítí nad metalickými jsou především v dosahu a kapacitě dosažitelné i na velké vzdálenosti. Důležitou vlastností je také vyšší bezpečnost přenosu dat - na metalickém vedení se snáze provádí odposlech. Proti metalickým sítím je velkou výhodou též galvanické oddělení zařízení na koncích kabelu.

2.1.1 Druhy optických vláken

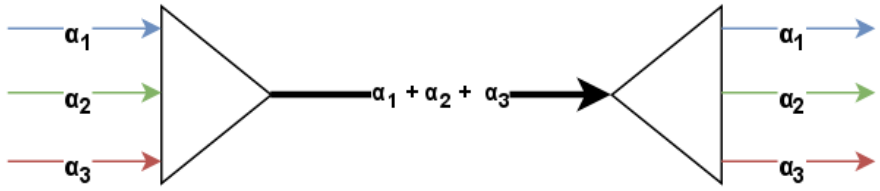
Běžně používaná vlákna můžeme rozdělit na dva druhy - jednovidová a mnohovidová, častěji používaná je anglická terminologicke single mode (SM) a mutli mode (MM).

Oba druhy se vzájemně liší způsobem šíření paprsku vláknem. Jejich rozdílné fyzikální vlastnosti tak ovlivňují jejich použití.

Mnohovidová vlákna se používají pro krátké a méně kapacitní propoje, jsou levnější na výstavbu. Jednovidová vlákna se používají na větší vzdálenosti, často v kombinaci se systémy Wavelength-Division Multiplexing (WDM).

2.1.2 Vlnový multiplex - WDM

Technologie WDM využívá přenosu více optických signálů po jednom vlákně, přičemž jednotlivé signály se liší vlnovou délkou (barvou).



Obrázek 2.2: Princip WDM

Tento přístup umožňuje oboustranou komunikaci po jednom vlákně. Zároveň je možné mnohonásobně zvýšit kapacitu optické trasy tím, že začneme komunikovat najednou pomocí více vlnových délek.

2.1.3 Aktivní optické sítě

Aktivní optické sítě (AON) pro svou funkci vyžadují aktivní síťové prvky. Těmito zařízeními jsou často switche a routery, které provádí logiku směrování provozu. Optická síť tak slouží jen jako médium přenosu dat, které by mohly být přenášeny i jinak.

2.1.4 Pasivní optické sítě

Pasivní optická síť (PON) naopak pro svou funkci nevyžaduje žádná aktivně napájená zařízení. Pomocí vlnových multiplexerů a demultiplexerů je možné oddělit síťový provoz již na fyzické vrstvě - využívá se zde WDM technologie.

Pasivní sítě jsou důležitý směr vývoje optických sítí, jelikož přinášejí řadu výhod proti sítím aktivním. Výhody jsou ekonomického i technického charakteru. Ve velké míře se dnes stavějí Fiber to the x (FTTx) sítě založené na PON.

2.2 Zařízení Czech Light

Czech Light je název řady fotonických zařízení vyvíjených sdružením CESNET. Sdružení CESNET v České republice plní roli National Research and Education Network (NREN). Jeho rolí je tak mimo jiné zajišťení síťové konektivity pro univerzity a další vědecká pracoviště. Taková síť má určitá specifika, které byly motivací při vývoji zařízení Czech Light. Jednou z těchto motivací byla i možnost stavět síť bez opticko-elektricko-optických (OEO) převodníků, za předpokladu, že síť bude stále řiditelná. Výhodou takového řešení je, že se mezi vysílačem a přijímačem nenacházejí žádné aktivní prvky.

Na začátku vývoje byly Erbium Doped Fibre Amplifier (EDFA) optické zesilovače, postupně se rodina zařízení Czech Light rozrostla o další zařízení jako variabilní Dense Wavelength-Division Multiplexing (DWDM) multiplexer nebo optické přepínače schopné přepínat optický signál z každého vstupu na každý jeden výstup. Některé přepínače umožňují také přepínáním jednoho vstupu na více výstupů zároveň. Jedná se tak o optický multicast, který je patentovanou technologií sdružení CESNET[6].

Obrázek 2.3: Logo Czech Light[6]

2.2.1 Modely zařízení Czech Light

Optické zesilovače - CLA

Prvním zařízením s označením Czech Light byl optický zesilovač CLA PB01, který byl poprvé nasazen v roce 2004 na 159 km dlouhou trasu. Tento zesilovač pracující na principu EDFA je použitelný pro zesílení DWDM signálů[6].

Dnes je k dispozici řada dalších modelů, které se liší výkonem zesílení, počtem portů a dalšími parametry.

Variabilní multiplexery - CL-VMUX

Tato zařízení jsou schopna na jednom konci spojit jednotlivé optické kanály do DWDM signálu a na druhém konci je zase rozdělit. Výhodou těchto zařízení je možnost vyrovnání úrovně jednotlivých kanálů na výstupu ze zařízení, což je předpokladem pro správnou funkčnost následujících optických zesilovačů.

Přepínače a multicastové přepínače - CLS a CLM

Optické přepínače jsou schopny přepínat optický signál z každého vstupu na libovolný výstup. Jednotlivé modely se liší počtem portů a typem použité optické přepínací matice[1], která určuje životnost a fyzikální vlastnosti přepínaných tras. Do této skupiny patří i zařízení,

na kterém probíhal vývoj NETCONF rozhraní, konkrétně se jednalo o model CLS-16x16. Zařízení řady CLM navíc proti CLS podporuje přepnutí jednoho vstupu na více výstupu (optický multicast).

Konfigurovatelný add-drop multiplexer - CL-ROADM

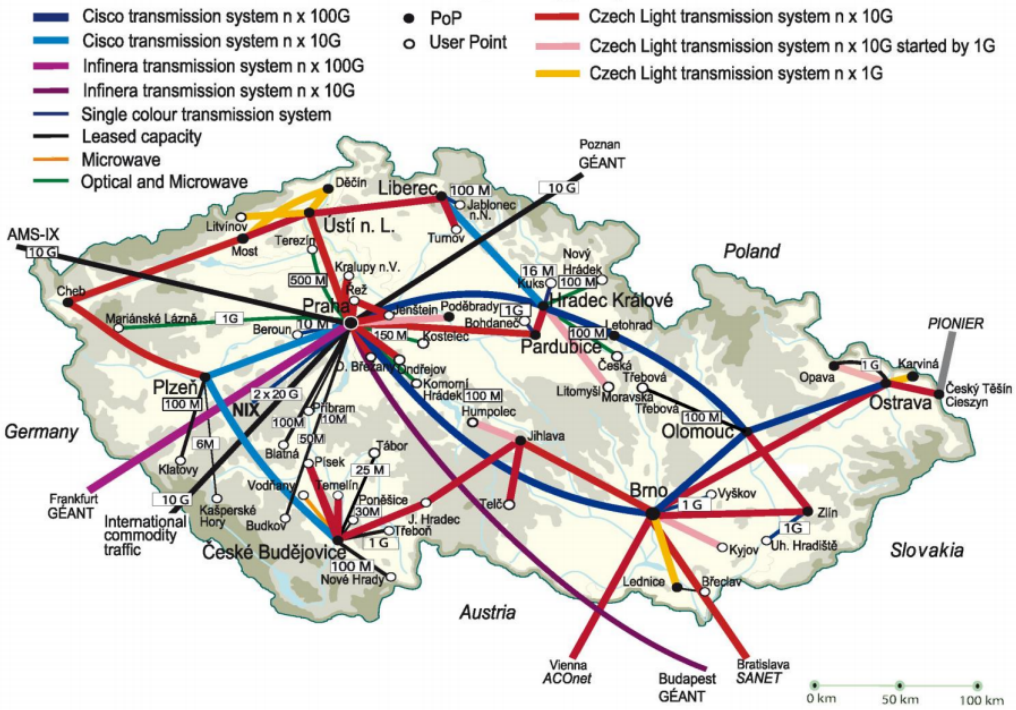
Toto zařízení dokáže dynamicky měnit složení jednotlivých vlnových délek na trase. Umí tak vydělit nebo naopak začlenit kanály do vlákna.

Selektivní přepínač vlnových délek - CL-WSS

Zařízení řady CL-WSS je schopno směrovat vybrané kanály z jednoho vlákna do jiného.

2.2.2 Nasazení zařízení Czech Light

Zařízení Czech Light kompletně vykrývají potřeby optického přenosového systému. Na páteřní síti je tak dnes produkčně nasazeno přes 80 zařízení a to nejen v síti CESNET.



Obrázek 2.4: Optická síť CESNET2[15]

Další místa, kde jsou použita zařízení Czech Light[6]:

- ROWANet - páteřní optická síť kraje Vysočina
- PilsFree - komunitní síť působící v Plzni v okolí
- Nasazení mimo ČR - Slovensko, Anglie, Dánsko, Ukrajina, Srbsko, Jordánsko, Egypt

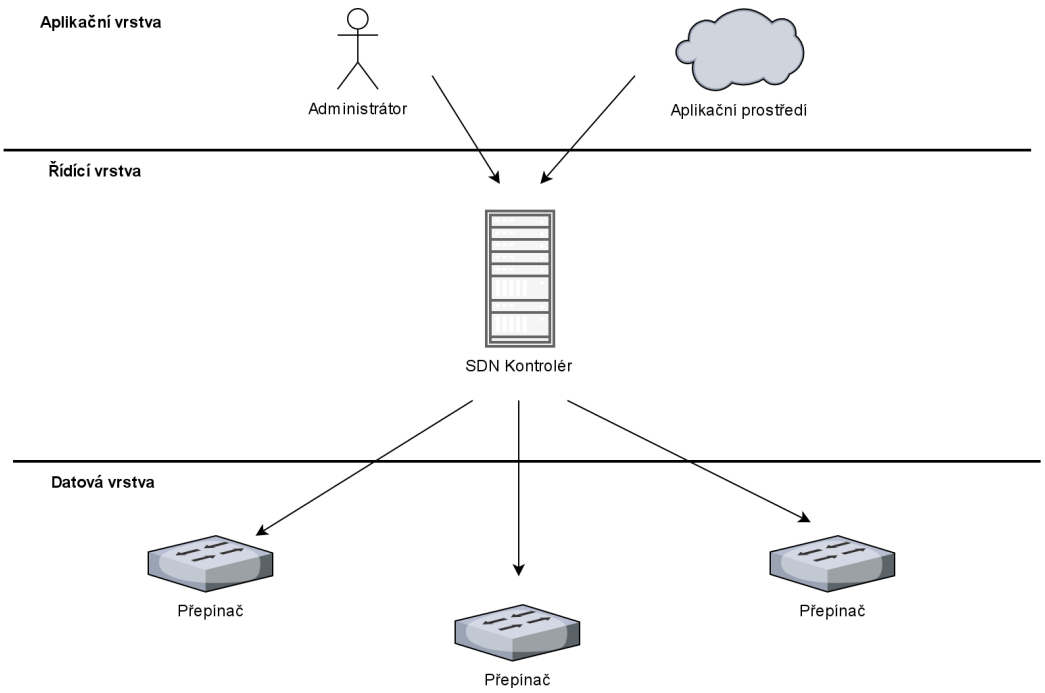
2.3 Softwarově definovaná síť

Technologie SDN vznikla jako reakce na potřebu pružnější konfigurace sítí s ohledem na trendy jako masivní nasazení virtualizace, výrazný růst datových toků a potřeby aplikací zasahovat do síťové infrastruktury, třeba z důvodu bezpečnosti nebo load-balancingu.

Je náročné tyto úkoly plnit v infrastruktuře, kde je každý síťový prvek ovládán samostatně. SDN přináší v tomto případě velké výhody, a to v několika směrech. Jednou z nich je komunikace pomocí otevřených protokolů, které jsou nezávislé na výrobci hardwaru. Důležitá je také abstrakce sítě, díky níž pro změny nastavení nepotřebujeme znát celou topologii. SDN síť je ovládána centrálně, což přináší výhody v podobě snazší správy zařízení, nebo lepšího řízení toku dat v síti.

2.3.1 Architektura SDN

Síť SDN můžeme rozdělit na tři vrstvy: Aplikační, řídící a datovou.



Obrázek 2.5: Architektura SDN sítě

Datovou vrstvu tvoří fyzická infrastruktura. Řídící vrstva znamená centrální kontrolér, který s datovou vrstvou komunikuje nejčastěji pomocí protokolu OpenFlow. Aplikační vrstva, tj. aplikace či administrátor, se baví s řídící vrstvou přes definované API.

2.3.2 Standardy používané v SDN

OpenFlow

Tento protokol slouží ke komunikaci mezi řídící a datovou vrstvou sítě. Jedná se dnes o pravděpodobně nejpoužívanější protokol v sítích SDN.

Ke svému fungování používá na zařízeních datové vrstvy tabulky, které určují, jak se má přepínač k jednotlivému provozu chovat[10]. Tyto tabulky si přepínače plní na základě rozhodnutí kontroléru. Přijde-li na přepínáč provoz, který neodpovídá tabulce, je přeposlán kontroléru, který určí, co s ním má přepínač udělat. Provoz (tok) je zpravidla určen pěticí zdrojová IP adresa, zdrojový port, cílová IP adresa, cílový port a transportní protokol. Klasifikace toku ale může být určena mnohem více parametry, jako je MAC adresa, VLAN ID a podobně.

Extensible Messaging and Presence Protocol - XMPP

Protokol XMPP byl původně vyvinut pro instant messaging. Postupně se začal pro svou robustnost používat i v jiných oblastech.

Protokol XMPP používá například společnost Juniper ve své SDN platformě Contrail[16].

Open Networking Operating System - ONOS

ONOS je open source projekt, který si klade za cíl vytvořit operační systém určený pro podporu SDN sítí.

NETCONF

Protokol NETCONF může být pro svou míru abstrakce nad zařízením také použit pro distribuci nastavení v sítích SDN. Je zde rozdíl v přístupu ke konfiguraci proti protokolu OpenFlow. NETCONF je primárně protokol pro správu zařízení, přičemž může podporovat opravdu širokou škálu funkcionality. Proti tomu OpenFlow v zásadě specifikuje, jak má být zařízení navrženo, od přepínačů vyžaduje minimum vlastní funkcionality.

2.4 Protokoly pro vzdálenou správu

2.4.1 Telnet

Připojení přes protokol Telnet na spravované zařízení je dnes používáno již jen okrajově. Jedná se o nezabezpečené Transmission Control Protocol (TCP) spojení pracující standardně na portu 23. Každý výrobce implementuje vlastní způsob ovládání zařízení, a to buď pomocí Command Line Interface (CLI), nebo Text User Interface (TUI). Protokol Telnet je popsán v RFC 854[11].

2.4.2 SSH

Protokol SSH, dnes používán především ve verzi 2, je jeden z nejčastějších způsobů správy vzdáleného zařízení založeného na systému GNU/Linux či BSD. Běžně pracuje na TCP portu 22 a veškerou komunikaci mezi klientem a serverem šifruje. Po přihlášení na zařízení pracuje uživatel s CLI nebo TUI, které opět není standardizované. Protokol SSH je popsán v RFC 4251[17].

2.4.3 Web UI

Správa zařízení přes webový prohlížeč, respektive přes protokol HTTP/HTTPS, je velmi častá. U mnohých SOHO zařízení je to jediný způsob administrace. Protokol HTTP[3] komunikaci nešifruje a standardně běží na portu 80, HTTPS[14] již komunikaci šifruje a poslouchá na portu 443. Webové rozhraní není určeno pro automatizovanou hromadnou správu zařízení. Implementace rozhraní se u jednotlivých výrobců velmi liší - rozdílný přístup k přihlašování, použití modulů v jazyce Java apod.

2.4.4 SNMP

Protokol SNMP slouží ke zjišťování stavu i k nastavování zařízení. Můžeme jej nalézt ve třech verzích - 1, 2c a 3. Verze 1 a 2c běžně pracuje na UDP portu 161, je nešifrovaná a za přístupové heslo k údajům můžeme považovat název komunity. Verze 3 přichází s volitelnou možností šifrování přístupových údajů (nyní již jméno a heslo) a také přenášených dat. Protokol SNMP je binární, získaná data mají stromovou strukturu. Popis struktury dat zařízení specifikují MIB soubory, které k zařízením obvykle dodává výrobce.

Příklad získání IP adresy zařízení pomocí SNMP:

```
$: snmpwalk -v2c -ccommunity  172.23.92.246  iso.3.6.1.2.1.4.20.1
iso.3.6.1.2.1.4.20.1.1.127.0.0.1 = IpAddress: 127.0.0.1
iso.3.6.1.2.1.4.20.1.1.172.23.92.246 = IpAddress: 172.23.92.246
iso.3.6.1.2.1.4.20.1.2.127.0.0.1 = INTEGER: 1
iso.3.6.1.2.1.4.20.1.2.172.23.92.246 = INTEGER: 10
iso.3.6.1.2.1.4.20.1.3.127.0.0.1 = IpAddress: 255.0.0.0
iso.3.6.1.2.1.4.20.1.3.172.23.92.246 = IpAddress: 255.255.255.252
iso.3.6.1.2.1.4.20.1.4.127.0.0.1 = INTEGER: 0
iso.3.6.1.2.1.4.20.1.4.172.23.92.246 = INTEGER: 1
```

Protokol SNMP je popsán v řadě RFC.

2.4.5 API

Další možností přístupu k zařízení je přes výrobcem dodávané API. Za příklad mohu vybrat RouterOS od společnosti MikroTik, kde je většina funkcionality zpřístupněna i přes API[8], které může komunikovat i šifrovaně. Existují knihovny pro různé programovací jazyky, výrobce tak počítá například s automatickým exportem nastavení přímo z informačního systému.

Nevýhodou těchto API je situace, kdy chceme síť řídit z centrálního prvku. Každý výrobce implementuje vlastní způsob ovládání. Chceme-li automatizovat nastavení zařízení od různých výrobců, musíme abstrahovat a následně vytvářet konkrétní implementace pro jednotlivá API. Jednotlivá zařízení navíc ani nemusejí mít stejný rozsah schopností API.

2.4.6 NETCONF

Protokol NETCONF je definován v RFC 6241[12]. Jedná se o protokol, který je určen ke kompletní správě zařízení. Základní funkcionalitou je získávání stavových dat a konfigurace.

Data přenášená a ukládaná protokolem NETCONF jsou obvykle ve formátu XML. Ke komunikaci je používano metody Remote Procedure Call (RPC), přenos probíhá přes zabezpečené spojení TLS nebo SSH. Server běžně naslouchá na TCP portu 830.

Zařízení může obsahovat několik verzí konfigurace. Standardně jsou to `running`, `startup` a `candidate`. Tyto konfigurace jsou uloženy v tzv. datastoru ve formátu XML. Každý modul si udržuje vlastní datastore s různými verzemi nastavení.

Další vlastností protokolu je možnost zámků konfigurace, která zabraňuje neintegritě nastavení při konfiguraci z více míst současně.

NETCONF protokol obsahuje řadu příkazů, uvedu některé z nich.

Příkaz	Popis
get	Vyžádá `running` konfiguraci zařízení a stavová data
get-config	Vyžádá konfiguraci zařízení, uživatel musí specifikovat datastore
edit-config	Upraví konfiguraci specifikovaného datastoru
copy-config	Zkopíruje obsah jednoho datastoru do druhého
lock	Zamkne přístup ke konfiguraci
unlock	Odemkne přístup ke konfiguraci
commit	Zkopíruje obsah `candidate` datastoru do `running`

Tabulka 2.1: Vybrané příkazy protokolu NETCONF

Konfigurace zařízení je rozdělena do modulů podle funkcí. Jeden modul tak může popisovat nastavení síťových rozhraní, jiný například nastavení databáze nebo SSH serveru. Modely konfigurací jsou popisovány jazykem YANG. Více o jazyku YANG je uvedeno v sekci návrhu modelu.

2.5 Porovnání protokolů vzdálené správy

2.5.1 Standardizovanost

Jednotlivé způsoby vzdálené správy můžeme podle míry standardizace rozdělit do tří kategorií:

Žádný popisující standard

Do této skupiny patří API - každý výrobce specifikuje své vlastní API, které je sice často detailně popsáno, ale týká se jen úzkého okruhu zařízení.

Standardizován přenos, nikoli obsah

V této skupině se nachází SSH, Telnet a webové UI. Protokoly pro připojení jsou popsány v RFC (SSH, Telnet i HTTP(S)), nicméně obsah komunikace se výrazně liší. Ovládání zařízení přes SSH a Telnet velmi ovlivňuje operační systém, webové UI je vždy specifické.

Standardizován přenos i obsah

Oba protokoly SNMP a NETCONF jsou detailně popsány v příslušných RFC.

2.5.2 Bezpečnost přenosu

Možný šifrovaný přenos

Zpravidla nešifrovaný přenos používá jen protokol Telnet, nicméně existují i zřídka používané implementace Telnetu s podporou SSL/TLS. V případě Web UI je přechod k šifrovanému přenosu snadný, stačí použít HTTPS. U protokolu SNMP je verze 1 a 2c vždy nešifrovaná, šifrovaný přenos a autentizace pomocí jména a hesla je možná až ve verzi 3. V případě API záleží na výrobci.

Šifrovaný přenos vždy

Protokol SSH používá šifrovanou komunikaci vždy, uživateli nabízí také řadu způsobů zabezpečení přístupu a přenosu. NETCONF používá k přenosu dat protokol SSH nebo data šifruje pomocí TLS.

2.5.3 Použití pro hromadnou správu

Ne všechny protokoly se hodí pro hromadnou správu zařízení. V případě Web UI můžeme takový přístup téměř zavrhnout. Pro protokol SSH je mnoho možností realizace hromadné správy, za zmínku stojí například projekt `Ansible`. Protokol SNMP slouží spíše pro dohled, SET operace často nejsou vůbec implementovány. API a NETCONF jsou pro hromadnou správu přímo určeny.

Kapitola 3

Návrh NETCONF rozhraní optického přepínače

3.1 Popis optického přepínače CLS-16x16

3.1.1 Hardware

Obrázek 3.1: Přepínač CLS-16x16, zadní pohled[1]

Přepínač CLS-16x16 je umístěn ve 2U case. V přední části se nacházejí SC konektory pro optické vstupy a výstupy. V zadní části jsou umístěny dva redundantní zdroje na 230 V a I/O porty vestavěného počítače architektury x86. Ke komunikaci s uživatelem slouží dvě gigabitová ethernetová rozhraní nebo RS-232 sériový port.

Obrázek 3.2: Přepínač CLS-16x16, přední pohled[1]

Uvnitř můžeme najít jednotku typu MEMS, která je schopna propojovat optické vstupy s výstupy. Jednotlivé modely switche používají různé optické jednotky, které se liší množstvím portů, útlumem, rychlostí přepínání, velikostí přeslechů a také životností.

3.1.2 Softwarové prostředí

Na přepínači je nainstalována distribuce vycházející z Debian GNU/Linuxu. V době psaní práce byla poslední verze 5.0RC1, která je založena na Debianu Jessie. Systém je nainstalován na CF kartě a je zkompilovaný pro architekturu x86.

Aby se karta nepoškozovala zbytečně častými zápisy, běží celý systém v operační paměti a každou požadovanou změnu v souborovém systému je třeba explicitně zapsat do perzistentního úložiště. Pro vzdálený přístup slouží především protokol SSH, pomocí kterého lze ovládat přepínací optickou matici programem `oswitch-dicon16x16`.

3.2 YANG model

Pro popis funkcionality přepínače je použit modelovací jazyk YANG[7]. Jazyk YANG je určen pro popis konfigurace a stavových informací zařízení při použití protokolu NETCONF. Data jsou uchována ve stromové struktuře, pro jejich reprezentaci může být použit i formát XML nebo JSON. Jazyk vytvořila pracovní skupina IETF a je definován v RFC 6020.

Popis modelu přepínače

Nejprve je nutné specifikovat název modulu, v tomto případě `czech-light-switch`.

```
module czech-light-switch {
```

Dále definuji použitý XML namespace, který dle specifikace musí být validní URI. Tento namespace je používán při NETCONF operacích pro jednoznačné určení modulu. Následně definuji prefix, který slouží k substituci namespace v referencích modelu.

```
namespace "urn:clswitch";
prefix "clsw";

organization "Faculty of Electrical Engineering,
 Czech Technical University in Prague";
description  "Module describing CESNET Czech Light optical switch";
contact      "cervelu5@fel.cvut.cz";
```

Výraz `container` je používán k definici funkcionality, je to zpravidla uzel datového stromu. Container nicméně může být v konfigurační části i prázdný, v takových případech pak obvykle zapíná danou funcionalitu s výchozími hodnotami vnitřních parametrů[7]. V tomto případě definuji container `czech-light-switch`, který obaluje všechna data.

```
container czech-light-switch {
```

Uvnitř se nachází list `switch-matrix-size`, který obsahuje počet vstupních (výstupních) portů přepínače. Tento list je jen stavová informace, která se nedá konfigurací změnit. Neměnnost určuje parametr `config "false"`. Zároveň je to informace, která nesmí být vynechána, to je zajištěno díky zápisu `mandatory "true"`.

```
leaf switch-matrix-size {
    config "false";
    mandatory "true";
    type uint16;
}
```

Dalším prvkem je container `crossconnects`, kterým jsou definována jednotlivá propojení. Propojení jsou definována v seznamu `crossconnect` (list `crossconnect`). Tento seznam (list) se dá chápat jako tabulka dvojic hodnot, jejímž unikátním klíčem je parametr `in-port` a hodnotou `out-port`, kdy oba představují číslo portu.

```
container crossconnects {

    list crossconnect {
        ordered-by system;

        key in-port;
        leaf in-port {
            config "false";
            type uint16;
            description "Input port number";
        }

        leaf out-port {
            config "true";
            type uint16;
            description "Output port number";
        }
    }
}}
```

Celý YANG model přepínače je k dispozici v příloze práce.

3.3 Architektura NETCONF serveru

Protokol NETCONF přináší řadu velmi užitečné funkcionality pro uživatele, která ovšem z pohledu vývojáře může být náročná na implementaci. Naštěstí dnes existuje několik možností, jak implementovat rozhraní NETCONF serveru, aniž bychom museli implementovat samotný protokol[9].

Pro implementaci NETCONF rozhraní do přepínače Czech Light jsem použil projekt `Netopeer`[5], který využívá funcionality knihovny `libnetconf`[4]. `Netopeer` i `libnetconf` jsou vyvíjeny, stejně jako Czech Light zařízení, sdružením CESNET.

3.3.1 Knihovna libnetconf

Knihovna `libnetconf` je napsána v jazyce `C` a je určena pro implementaci NETCONF serverových a klientských aplikací.

Poskytuje funkcionalitu jako připojení klienta na NETCONF server pomocí SSH (RFC 6242), odesílání a přijímání NETCONF zpráv (RFC 6241) a práci s konfigurací v datastoru.

3.3.2 Projekt Netopeer

Projekt Netopeer je soubor nástrojů pro práci s NETCONF protokolem[5].

- **netopeer-cli** je CLI klient pro protokol NETCONF, pomocí kterého lze získávat, nahrávat a manipulovat s konfigurací a daty zařízení
- **netopeer-server** je serverová aplikace implementující NETCONF protokol. Její funkcionalita je založena na TransAPI modulech.
- **netopeer-gui** je webová aplikace, která funguje jako NETCONF klient. Dokáže se připojovat na vzdálené servery a znázornit jejich konfugiraci. Dokáže získávat data, měnit konfiguraci, pracovat s datastory.
- **netopeer-configurator** je TUI nástroj pro základní nastavení Netopeer serveru. Umožňuje aktivovat a deaktivovat jednotlivé TransAPI moduly, importovat SSH klíče pro přihlášení k serveru a nastavovat bezpečností politiku.
- **netopeer-manager** je CLI nástroj pro práci s TransAPI moduly.

3.3.3 TransAPI moduly

TransAPI moduly jsou určeny pro Netopeer server, kde slouží pro implementaci funkcionality. Vycházejí z YANG modelu dané funkcionality, mají pevně danou strukturu kódu. Pro import slouží výsledná dynamická C knihovna a YANG (resp. YIN) model.

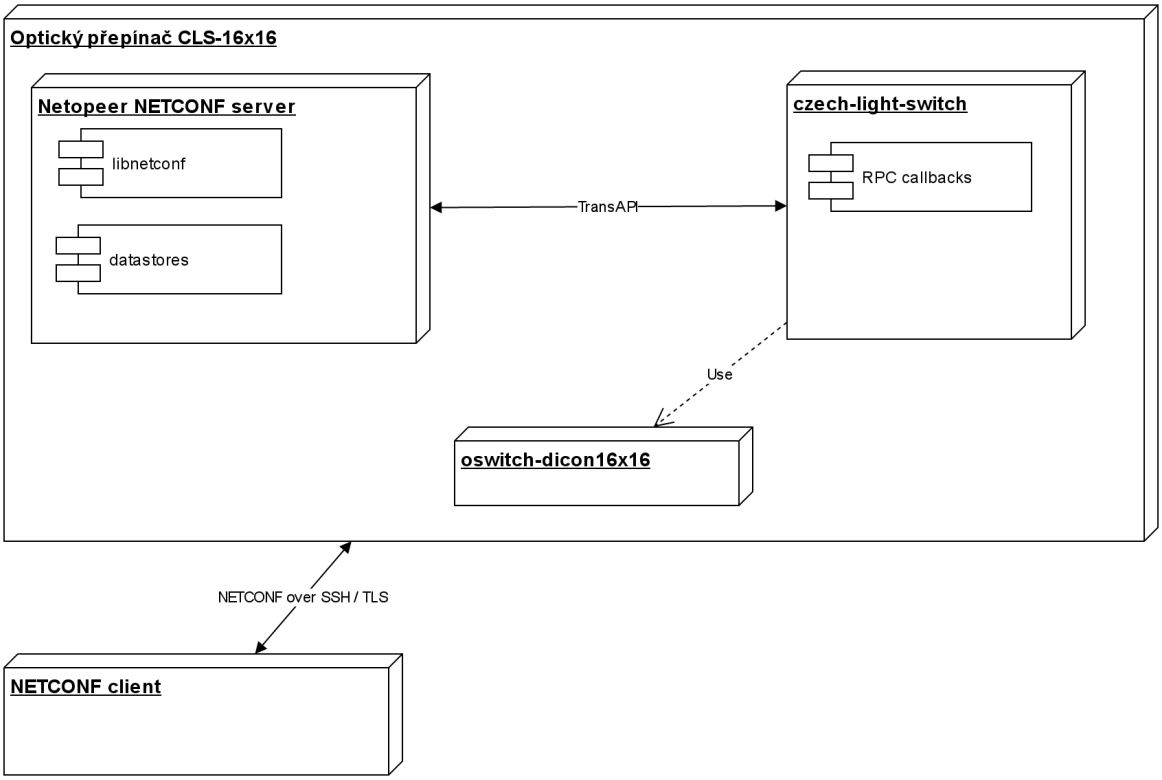
Moduly TransAPI pracují s XML daty pomocí `libxml2`. Každý modul má úložiště se svou konfigurací, které se nazývá datastore. Implicitní datastore vypadá následovně:

```
<?xml version="1.0" encoding="UTF-8"?>
<datastores xmlns="urn:cesnet:tmc:datastores:file">
  <running lock=""/>
  <startup lock=""/>
  <candidate modified="false" lock=""/>
</datastores>
```

Součástí Netopeer serveru jsou tři TransAPI moduly. Jeden je čistě výukový a implementuje Turingův stroj. Další moduly `cfgsystem` a `cfginterfaces` implementují funkcionalitu pro ovládání základních systémových nastavení a pro správu síťových zařízení.

3.3.4 Schéma komponent

Následující schéma popisuje propojení a komunikaci jednotlivých částí.



Obrázek 3.3: Schéma komponent

Kapitola 4

Implementace

Tato kapitola popisuje způsob implementace Netopeer serveru a integraci vlastního TransAPI modulu. Není nutné ani žádoucí postupovat v produkčním prostředí stejným způsobem. Pro produkční prostředí je připraven balík, který při instalaci automaticky provede požadované úkony.

4.1 Příprava a instalace Netopeer serveru

Příprava Netopeer serveru a vývoj TransAPI modulu probíhaly na virtuálním stroji s distribucí Debian Jessie architektury x86, abych se co nejvíce přiblížil prostředí přepínače co do verzí balíků, knihoven a kompatibility binárních souborů.

Nejprve za pomocí programu `git` stáhneme aktuální verzi `libnetconf` a `Netopeer`.

```
git clone https://github.com/CESNET/libnetconf.git
git clone https://github.com/CESNET/netopeer.git
```

4.1.1 Závislosti a příprava prostředí

Před samotnou kompilací je nutné připravit na systému závislosti. Většina z nich se nachází v repozitáři ve správné verzi.

```
apt-get update
apt-get install pkg-config libtool libxml2-dev libxslt1-dev\
 libcurl4-openssl-dev python-pip xsltproc python-libxml2
pip install pyang
```

Dále potřebujeme nainstalovat balík `libssh-dev` a `libssh-4`, který se bohužel v současném Debianu stable nachází jen ve verzi 0.6.3, ale `libnetconf` pro správnou funkci vyžaduje alespoň verzi 0.6.4. Software `libssh` tak buď stáhneme a zkompilujeme ručně, nebo můžeme využít balíčku z repozitáře například Debian sid, kde se nyní nachází verze 7.3-2.

Upravíme tedy soubor `/etc/apt/sources.list`, kde repozitář `jessie` nahradíme repozitářem `sid`.

```
apt-get update && apt-get install libssh-dev
```

Následně vrátíme zpět repozitář pro jessie a spustíme znovu

```
apt-get update
```

4.1.2 Kompilace

Protože Netopeer staví na `libnetconf`, je nutné ji instalovat první.

```
cd libnetconf
./configure
make && make install
```

Po úspěsně instalaci `libnetconf` přejdeme k instalaci Netopeer serveru.

```
cd -
cd netopeer/server
./configure
make && make install
```

V této chvíli máme ve vývojovém prostředí funkční server, který můžeme spustit pomocí `netopeer-server`.

4.1.3 Instalace na přepínač

Nejprve musíme na přepínači nainstalovat závislosti, tentokrát ale bez `-dev` balíčků.

```
apt-get update
apt-get install python-libxml2 libxslt1.1\
 libcurl3 python2.7
```

Dále musíme nainstalovat `libssh-4`, kde se ovšem opět potýkáme s problémem s verzemi. Řešení je stejné jako v předcházející kapitole.

Nejjednodušší cestou, jak nyní nainstalovat `libnetconf` a `netopeer-server`, je zkopírovat složku se zkompilovaným projektem z vývojového prostředí na přepínač a spustit `make install`.

```
scp -r user@testovaci_server:libnetconf ./
scp -r user@testovaci_server:netopeer ./
apt-get install make binutils
cd libnetconf
make install
cd ../netopeer/server
make install
```

Po instalaci smažeme složky `libnetconf` a `netopeer`. Když se teď pokusíme spustit `netopeer-server`, dostaneme chybovou hlášku

```
# netopeer-server
netopeer-server[4003]: sock_listen:
could not create socket
(Address family not supported by protocol)
```

Tato chyba nám říká, že se server snaží naslouchat na IPv6 adrese, kterou ale bohužel CL distro nepodporuje. Musíme tak ve `startup` datastoru NETCONF serveru nastavit, aby služba naslouchala jen na IPv4 adrese.

Jedná se o soubor `/usr/local/etc/netopeer/cfgnetopeer/datastore-server.xml`, který upravíme následujícím způsobem:

```
<?xml version="1.0" encoding="UTF-8"?>
<datastores xmlns="urn:cesnet:tmc:datastores:file">
  <running lock=""/>
  <startup lock="">
    <netconf xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-server">
      <ssh>
        <listen>
          <interface>
            <address>0.0.0.0</address>
            <port>830</port>
          </interface>
        </listen>
      </ssh>
    </netconf>
  </startup>
  <candidate modified="false" lock=""/>
</datastores>
```

Nyní by se již měl Netopeer server spustit bez chyb. Provedené změny uložíme na CF kartu pomocí příkazu:

```
savechanges
```

Tímto se vytvoří soubor `/mnt/live/memory/data/cldistro/changes/XXX_changes.sb`, který obsahuje změněné soubory proti předchozí instalaci.

4.2 TransAPI modul pro optické přepínání

4.2.1 Vývoj modulu podle YANG modelu

Základem TransAPI modulu je YANG model, který určuje jeho funkcionalitu. Z existujícího YANG modelu můžeme vytvořit kostru implementace pomocí nástrole `lnctool`[2]. Tento nástroj je napsaný v jazyce Python a pro svůj běh potřebuje knihovnu `pyang`.

Nejprve je nutné připravit soubor, kde jsou definované cesty. V tomto případě vypadá soubor `paths_file` takto:

```
clsw=urn:clswitch
/clsw:czech-light-switch/clsw:crossconnects
```

První řádek určuje prefix a XML namespace projektu. Další řádky značí cestu k datům modelu, na jejichž změnu má modul reagovat. Pro účely `czech-light-switch` modulu stačí reagovat na změny v containeru `crossconnects`.

Dalším krokem je spuštění příkazu `lnctool`, který syntakticky validuje model a vygeneruje souborovou strukturu implementace.

```
lnctool -model ./czech-light-switch.yang transapi -paths ./paths_file
```

Soubory TransAPI modulu:

```
- config.guess
- config.sub
- configure.in
- czech-light-switch.c (zdrojový kód funkcionality modulu s pevnou strukturou)
- czech-light-switch-config.rng (soubor pro validaci)
- czech-light-switch-gdefs-config.rng (soubor pro validaci)
- czech-light-switch-schematron.xsl (soubor pro validaci)
- czech-light-switch.yang (původní model ve formátu YANG)
- czech-light-switch.yin (model ve formátu YIN převedený pomocí pyang)
- install-sh
- ltmain.sh
- Makefile.in
- paths_file (definice namespace, prefixu a cesty v modelu)
```

Modul je před importem třeba zkompilovat. Pomocí nástroje `autoreconf` si připravíme `configure` script a soubor `Makefile`.

```
autoreconf --force --install
./configure
make
```

4.2.2 Implementace funkcionality

Veškerá funkcionalita se nachází v souboru `czech-light-switch.c`, který má pevně danou strukturu.

Důležité části kódu

`int transapi_init(xmlDocPtr *running)`

V této metodě se provádí inicializace modulu. Současný stav se nastaví do `running` konfigurace.

`xmlDocPtr get_state_data(xmlDocPtr model, xmlDocPtr running, struct nc_err **err)`

Metoda get_state_data vrací aktuální stavová (nekonfigurační) data v XML.

`struct ns_pair namespace_mapping[] = {{"clsw", "urn:clswitch"}}`

Struktura ns_pair mapuje definované prefixy a namespaces.

```
struct transapi_data_callbacks clbks =  {
    .callbacks_count = 1,
    .data = NULL,
    .callbacks = {
        {.path = "/clsw:czech-light-switch/clsw:crossconnects",
        .func = callback_clsw_czech_light_switch_clsw_crossconnects}
    }
};
```

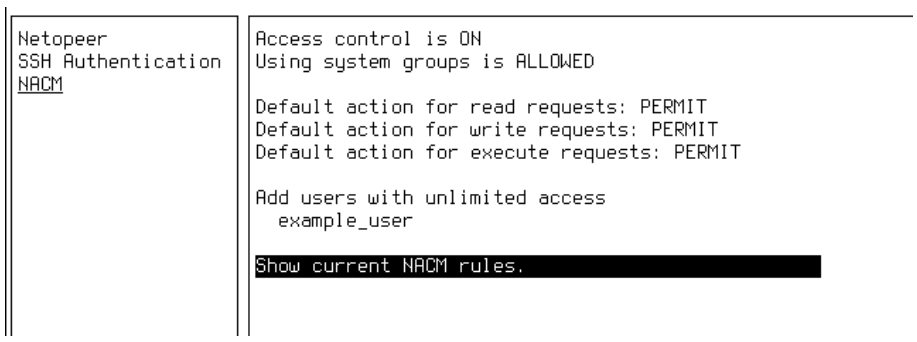
Struktura `clbks` slouží k tomu, aby při změně konfiguračního stromu byly volány ty správné callback metody.

4.2.3 Import modulu do Netopeer serveru

TransAPI modul se do Netopeer serveru importuje nástrojem `netopeer-manager`. Tento nástroj také dokáže měnit nastavení modulu.

Při importu modulu je nutné mít soubor s modelem ve formátu YIN. Je-li na zařízení nainstalovaná utilita `pyang`, provede se konverze automaticky, v opačném případě je nutné soubor YIN vygenerovat z YANG jinak. Pro přidání `czech-light-switch` modulu spustíme:

```
netopeer-manager add --name czech-light-switch\
 --model czech-light-switch.yin --transapi .libs/czech-light-switch.so\
 --datastore /usr/local/etc/netopeer/clswitch-datastore.xml
```



Obrázek 4.1: TUI rozhraní nástroje netopeer-configurator

Správně provedenou instalací se modul automaticky aktivuje. Po instalaci modulu budeme chtít pravděpodobně změnit bezpečnostní nastavení NACM. To můžeme udělat pomocí nástroje `netopeer-configurator`.

Před spuštěním nesmíme zapomenout vložit do datastoru počáteční nastavení, výchozí nastavení je v přiloze.

```
cp default-clswitch-datastore.xml /usr/local/etc/netopeer/clswitch-datastore.xml
```

4.2.4 Ověření funkčnosti

Pro ověření funkčnosti použijeme program `netoper-cli`. Nejprve se přihlásíme na spuštěný NETCONF server:

```
$: netopeer-cli
netconf> connect --login root 192.168.56.12
root@192.168.56.12 password:
```

Zavoláním příkazu `status` se dozvíme základní informace o NETCONF serveru a souhrn podporované funkcionality - jeho „capabilities“.

```
netconf> status
Current NETCONF session:
  ID          : 3
  Host        : 192.168.56.12
  Port        : 830
  User        : root
  Transport   : SSH
  Capabilities:
urn:ietf:params:netconf:base:1.0
urn:ietf:params:netconf:base:1.1
urn:ietf:params:netconf:capability:writable-running:1.0
urn:ietf:params:netconf:capability:candidate:1.0
...
urn:ietf:params:netconf:capability:validate:1.0
urn:ietf:params:netconf:capability:validate:1.1
urn:ietf:params:netconf:capability:with-defaults:1.0?basic-mode=...
urn:ietf:params:netconf:capability:url:1.0?scheme=scp,file
urn:clswitch?module=czech-light-switch&revision=2016-12-13
urn:cesnet:tmc:netopeer:1.0?module=netopeer-cfgnetopeer&revision=...
...
```

Pomocí příkazu `get` lze získat současnou konfiguraci a state date. Pro přehlednost můžeme na příkaz `get` aplikovat filtry.

```
netconf> get --filter
<netopeer xmlns="urn:cesnet:tmc:netopeer:1.0" />

Result:
\begin{verbatim}
<netopeer xmlns="urn:cesnet:tmc:netopeer:1.0">
<modules>
  <module>
    <name>czech-light-switch</name>
    <enabled>true</enabled>
  </module>
</modules>
</netopeer>
```

Konečně můžeme též vyfiltrovat informace poskytnuté `czech-light-switch` modulem.

```
netconf> get --filter
<czech-light-switch xmlns="urn:clswitch" />

Result:
<czech-light-switch xmlns="urn:clswitch">
    <crossconnects>
      <crossconnect>
        <in-port>1</in-port>
        <out-port>0</out-port>
      </crossconnect>
...
      <crossconnect>
        <in-port>16</in-port>
        <out-port>0</out-port>
      </crossconnect>
    </crossconnects>
    <switch-matrix-size>16</switch-matrix-size>
  </czech-light-switch>
```

Kapitola 5

Testování

Testování implementace probíhalo pomocí programu `netopeer-cli`. Pro otestování implementace jsem zvolil několik testovacích scénářů.

Poznámka: V následujícím textu znamená úvodní znak `#` spuštění v příkazové řádce s právy uživatele root na přepínači. `netconf>` na začátku řádku znamená rozhraní programu `netopeer-cli` na zařízení, ze kterého je přepínač operátorem ovládán.

5.1 Úprava konfigurace

K úpravě konfigurace se používá příkaz `edit-config [datastore]`.

Nejprve vyzkouším změnit zapojení jednoho portu, změnu poté zkontroluji pomocí původní utility `oswitch-dicon16x16` a příkazu `get`.

```
# oswitch-dicon16x16 -l | grep -e "IN-1 " // kontrola zapojení
IN-1 OUT-0

netconf> edit-config running // editace running konfigurace

<czech-light-switch xmlns="urn:clswitch">
    <crossconnects>
      <crossconnect>
        <in-port>1</in-port>
        <out-port>16</out-port>
      </crossconnect>
    </crossconnects>
</czech-light-switch>

Result OK

# oswitch-dicon16x16 -l | grep -e "IN-1 " \\ kontrola zapojeni
IN-1 OUT-16
```

```
netconf> get --filter // kontrola running konfigurace s použitím filteru
<czech-light-switch xmlns="urn:clswitch">
    <crossconnects>
      <crossconnect>
        <in-port>1</in-port>
      </crossconnect>
    </crossconnects>
</czech-light-switch>

Result:
<czech-light-switch xmlns="urn:clswitch">
    <crossconnects>
      <crossconnect>
        <in-port>1</in-port>
        <out-port>16</out-port>
      </crossconnect>
    </crossconnects>
</czech-light-switch>
```

Příkaz `edit-config` pracoval dle očekávání.

5.2 Kopírování nastavení

V tomto testu vyzkouším kopírování mezi datastory, klíčový je příkaz `copy-config -source source_datastore target_datastore`.

Za testovací případ si vezmu následující scénář:

1. Start NETCONF serveru
2. Přepnutí portu za běhu v `running` konfiguraci
3. Zkopírování do `startup` konfigurace
4. Vypnutí NETCONF serveru
5. Manuální nastavení odlišného propojení přes systémové CLI
6. Zapnutí NETCONF serveru
7. Kontrola přepnutí portu

Průběh testu:

```
# netopeer-server -d  // start serveru

# oswitch-dicon16x16 -l | grep -e "IN-1 " // ověření stavu portu 1
IN-1 OUT-0
```

```
netconf> edit-config running // přepnutí portu 1 na port 5
<czech-light-switch xmlns="urn:clswitch">
    <crossconnects>
      <crossconnect>
        <in-port>1</in-port>
        <out-port>5</out-port>
      </crossconnect>
    </crossconnects>
</czech-light-switch>

Result OK

# oswitch-dicon16x16 -l | grep -e "IN-1 " // ověření stavu portu 5
IN-1 OUT-5

netconf> get-config --filter running // ověření running config datastoru
<czech-light-switch xmlns="urn:clswitch">
    <crossconnects>
      <crossconnect>
        <in-port>1</in-port>
      </crossconnect>
    </crossconnects>
</czech-light-switch>

Result:
<czech-light-switch xmlns="urn:clswitch">
    <crossconnects>
      <crossconnect>
        <in-port>1</in-port>
        <out-port>5</out-port>
      </crossconnect>
    </crossconnects>
</czech-light-switch>

netconf> get-config --filter startup // ověření startup config datastoru
<czech-light-switch xmlns="urn:clswitch">
    <crossconnects>
      <crossconnect>
        <in-port>1</in-port>
      </crossconnect>
    </crossconnects>
</czech-light-switch>

Result:
<czech-light-switch xmlns="urn:clswitch">
```

```
    <crossconnects>
      <crossconnect>
        <in-port>1</in-port>
        <out-port>0</out-port>
      </crossconnect>
    </crossconnects>
</czech-light-switch>

netconf> copy-config --source running startup // kopírování datastoru

Result OK
netconf> get-config --filter startup // ověření startup config datastoru
<czech-light-switch xmlns="urn:clswitch">
    <crossconnects>
      <crossconnect>
        <in-port>1</in-port>
      </crossconnect>
    </crossconnects>
</czech-light-switch>

Result:
<czech-light-switch xmlns="urn:clswitch">
    <crossconnects>
      <crossconnect>
        <in-port>1</in-port>
        <out-port>5</out-port>
      </crossconnect>
    </crossconnects>
</czech-light-switch>


# pkill netopeer-server // vypnutí serveru

# oswitch-dicon16x16 -i 1 -o 16 // manuální nastavení propojení
IN-1 OUT-16

# netopeer-server -d // start netopeer serveru

# oswitch-dicon16x16 -l | grep -e "IN-1 " // ověření stavu portu
IN-1 OUT-5
```

Test proběhl dle očekávání, bez chyb.

5.3 Validace nastavení

Mechanismy uvnitř NETCONF serveru by se měly postarat, aby do datastoru nešla vložit konfigurace, která není validní. Nemyslím tím jen syntakticky špatné XML, ale především nesoulad struktury dat s YANG modelem. Ne všechna omezení navíc lze v modelu vyjádřit - taková omezení pak musí validovat samotný TransAPI modul.

Scénář testu:

- Nastavení propojení, které sice odpovídá modelu, ale není realizovatelné. (IN 1, OUT 17)
- Nastavení vzájemně se vylučujících propojení (IN 1, OUT 1 a IN 2, OUT 1)
- Nastavení dat, která jsou pouze stavová (velikost přepínací matice)

Průběh 1. testu:

```
netconf> edit-config running // nastavení propoje 1 -> 17

<czech-light-switch xmlns="urn:clswitch">
    <crossconnects>
      <crossconnect>
        <in-port>1</in-port>
        <out-port>17</out-port>
      </crossconnect>
    </crossconnects>
</czech-light-switch>

NETCONF error: operation-failed (application) - Failed to apply configuration changes to

netconf> get --filter // kontrola running konfigurace
<czech-light-switch xmlns="urn:clswitch">
    <crossconnects>
      <crossconnect>
        <in-port>1</in-port>
      </crossconnect>
    </crossconnects>
  </czech-light-switch>

  Result:
<czech-light-switch xmlns="urn:clswitch">
    <crossconnects>
      <crossconnect>
        <in-port>1</in-port>
        <out-port>5</out-port>
      </crossconnect>
    </crossconnects>
  </czech-light-switch>
```

Test proběhl dle očekávání, TransAPI modul odmítl nastavit propoj na neexistující port 17, neprovedla se žádná změna.

Průběh 2. testu:

```
netconf> edit-config running // nastavení propoje 1 -> 1 & 2 -> 1

<czech-light-switch xmlns="urn:clswitch">
    <crossconnects>
      <crossconnect>
        <in-port>1</in-port>
        <out-port>1</out-port>
      </crossconnect>
      <crossconnect>
        <in-port>2</in-port>
        <out-port>1</out-port>
      </crossconnect>
    </crossconnects>
</czech-light-switch>

NETCONF error: operation-failed (application) - Failed to apply configuration changes to device

netconf> get --filter // kontrola running konfigurace
<czech-light-switch xmlns="urn:clswitch">
    <crossconnects>
      <crossconnect>
        <in-port>1</in-port>
      </crossconnect>
     <crossconnect>
       <in-port>2</in-port>
     </crossconnect>
    </crossconnects>
  </czech-light-switch>

  Result:
<czech-light-switch xmlns="urn:clswitch">
    <crossconnects>
      <crossconnect>
        <in-port>1</in-port>
        <out-port>5</out-port>
      </crossconnect>
     <crossconnect>
       <in-port>2</in-port>
       <out-port>0</out-port>
     </crossconnect>
    </crossconnects>
  </czech-light-switch>
```

Test proběhl dle očekávání, TransAPI modul odmítl nastavit konfliktní nastavení, neprovedla se žádná změna.

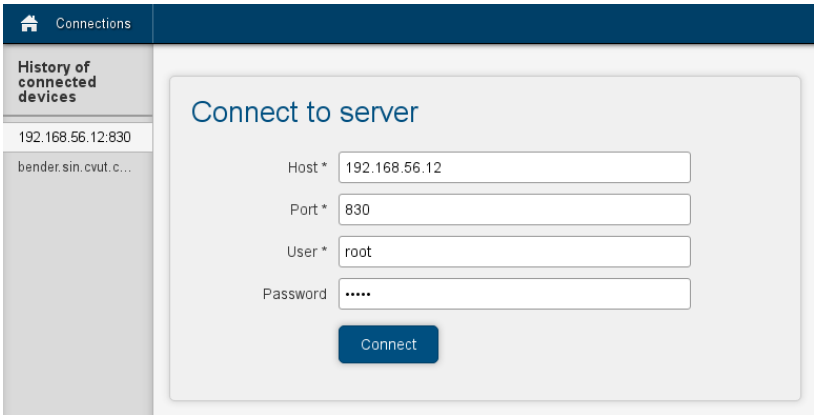
Průběh 3. testu:

```
netconf> edit-config running // nastavení velikosti switch-matrix-size

<czech-light-switch xmlns="urn:clswitch">
    <switch-matrix-size>37</switch-matrix-size>
</czech-light-switch>

NETCONF error: operation-failed (application) -
Datastore fails to validate
(Did not expect element switch-matrix-size there)

netconf> get --filter // kontrola running konfigurace
<czech-light-switch xmlns="urn:clswitch">
    <switch-matrix-size/>
</czech-light-switch>

  Result:
<czech-light-switch xmlns="urn:clswitch">
    <switch-matrix-size>16</switch-matrix-size>
  </czech-light-switch>
```

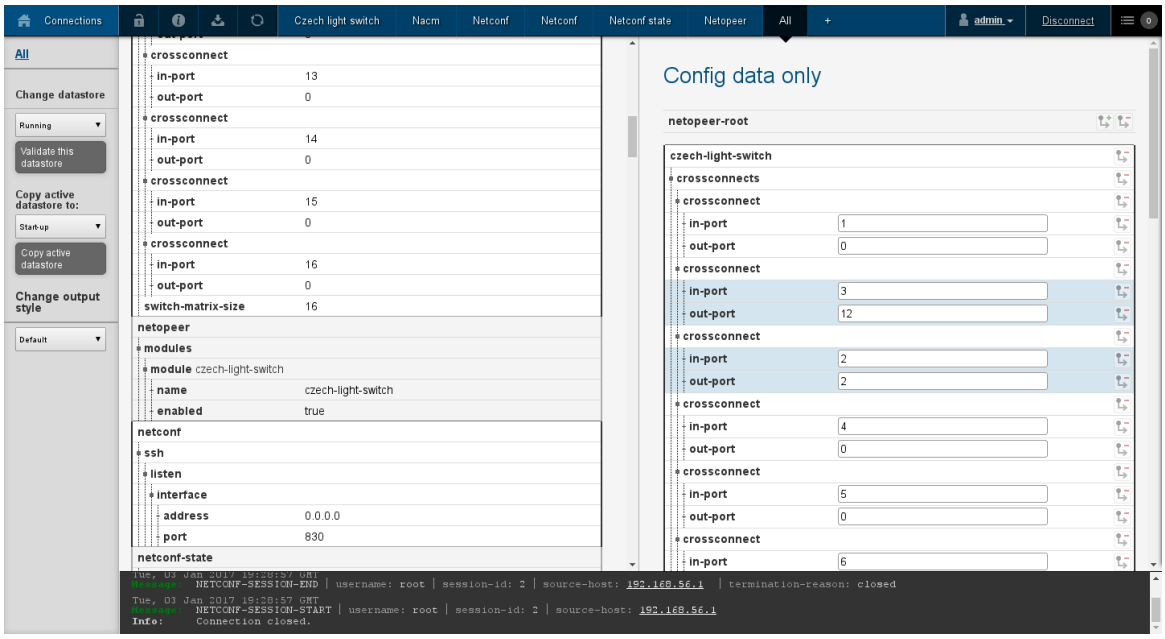
Test proběhl dle očekávání, požadované nastavení neprošlo validací `libnetconf` a TransAPI modulu nebylo vůbec předáno.

5.4 Netopeer GUI

Netopeer GUI je webová aplikace, která slouží jako NETCONF klient. Aplikace umožňuje vytvářet a editovat XML konfiguraci v grafickém prostředí prohlížeče.



Obrázek 5.1: Přihlášení na NETCONF server v Netopeer GUI



Obrázek 5.2: Výpis nastavení zařízení v Netopeer GUI

Kapitola 6

Závěr

Hlavním cílem této bakalářské práce bylo navrhnout a implementovat NETCONF rozhraní do optického přepínače Czech Light. V rámci této práce byl navržen konfigurační model zařízení v jazyce YANG. Dále byl popsán výběr, instalace a konfigurace softwarových komponent potřebných pro práci s NETCONF protokolem.

Výsledkem je TransAPI modul pro Netopeer NETCONF server, který implementuje funkcionalitu popsanou v YANG modelu. Byl popsán postup instalace, konfigurace a otestování takového řešení s ohledem na specifika software a hardware přepínače.

Práce se okrajově též zabývá analýzou používaných protokolů vzdálené správy a softwarově definovanými sítěmi.

Literatura

[1] CESNET, z. s. p. o. CzechLight Switch & CzechLight Multicast switch. Dostupné z: <http://czechlight.cesnet.cz/documents/download/43-EN-CSNT-CLS_v1.3.pdf>.

[2] Dokumentace knihovny libnetconf. Dostupné z: <https://rawgit.com/CESNET/libnetconf/master/doc/doxygen/html/d9/d25/transapi.html>.

[3] FIELDING, R. et al. Hypertext Transfer Protocol – HTTP/1.1. RFC 2616, RFC Editor, 1999. Dostupné z: <http://www.rfc-editor.org/rfc/rfc2616.txt>.

[4] GitHub repozitář knihovny libnetconf. Dostupné z: <https://github.com/cesnet/libnetconf>.

[5] GitHub repozitář projektu Netopeer. Dostupné z: <https://github.com/cesnet/netopeer>.

[6] HAVLIš, O. Vývoj a nasazení Czech Light. Dostupné z: <http://optolab.utko.feec.vutbr.cz/wp-content/uploads/Vývoj-a-nasazení-CzechLight.pdf>.

[7] M. BJORKLUND, E. – SYSTEMS, T. YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF). RFC 6020, RFC Editor, 2010. Dostupné z: <http://www.rfc-editor.org/rfc/rfc6020.txt>.

[8] Mikrotik RouterOS wiki. Dostupné z: <http://wiki.mikrotik.com/wiki/Manual:API>.

[9] Network Configuration Wiki. Dostupné z: <https://trac.ietf.org/trac/netconf>.

[10] OPENFLOW.ORG. Dostupné z: <http://archive.openflow.org/documents/openflow-spec-v1.1.0.pdf>.

[11] POSTEL, J. – REYNOLDS, J. TELNET PROTOCOL SPECIFICATION. RFC 854, RFC Editor, 1983. Dostupné z: <http://www.rfc-editor.org/rfc/rfc854.txt>.

[12] R. ENNS, E. et al. Network Configuration Protocol (NETCONF). RFC 6241, RFC Editor, 2011. Dostupné z: <http://www.rfc-editor.org/rfc/rfc6241.txt>.

[13] RAMASWAMI, R. – SIVARAJAN, K. – SASAKI, G. *Optical Networks: A Practical Perspective*. San Francisco, CA, USA : Morgan Kaufmann Publishers Inc., 3rd edition, 2009.

[14] RESCORLA, E. HTTP Over TLS. RFC 2818, RFC Editor, 2000. Dostupné z: <http://www.rfc-editor.org/rfc/rfc2818.txt>.

[15] VOJTěCH, J. Infrastruktura fotonických služeb. Dostupné z: <https://photonics.cesnet.cz/_media/publications/network-architecture/2016/telemetrologie16_v3.pdf>.

[16] Webová stránka společnosti Juniper. Contrail architecture white paper. Dostupné z: <http://www.juniper.net/us/en/local/pdf/whitepapers/2000535-en.pdf>.

[17] YLONEN, T. et al. The Secure Shell (SSH) Protocol Architecture. RFC 4251, RFC Editor, 2006. Dostupné z: <http://www.rfc-editor.org/rfc/rfc4251.txt>.

Příloha A

Seznam použitých zkratek

AON Aktivní optické sítě

CLI Command Line Interface

DWDM Dense Wavelength-Division Multiplexing

EDFA Erbium Doped Fibre Amplifier

FSO Free-space optic

FTTx Fiber to the x

NREN National Research and Education Network

OEO opticko-elektricko-optických

PON Pasivní optická síť

RPC Remote Procedure Call

TCP Transmission Control Protocol

TUI Text User Interface

WDM Wavelength-Division Multiplexing

Příloha B

Obsah přiloženého CD

```
- czech-light-switch-module - složka obsahuje zdrojové kódy modulu
config.guess
config.log
config.status
config.sub
configure
configure.in
czech-light-switch.c
czech-light-switch-config.rng
czech-light-switch-gdefs-config.rng
czech-light-switch.la
czech-light-switch-schematron.xsl
czech-light-switch.yang
czech-light-switch.yin
install-sh
libtool
ltmain.sh
Makefile
Makefile.in
paths_file

- default-clswitch-datastore.xml - výchozí datastore modulu

- overlay-packages - adresář s balíky určenými k instalaci
README.TXT - instrukce k instalaci
packages - adresář se samotnými balíky

- text - adresář s textem práce
```