VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

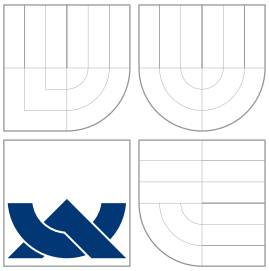
# NETOPEER: KONFIGURAČNÍ PLATFORMA PRO SÍŤOVÁ ZAŘÍZENÍ

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

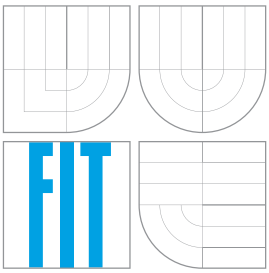
AUTOR PRÁCE
AUTHOR

MAREK ŽIŽLAVSKÝ

BRNO 2010



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

# NETOPEER: KONFIGURAČNÍ PLATFORMA PRO SÍŤOVÁ ZAŘÍZENÍ

NETOPEER: CONFIGURATION PLATFORM FOR NETWORK DEVICES

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

MAREK ŽIŽLAVSKÝ

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. JAN KOŘENEK

BRNO 2010

# Abstrakt

Práca sa zaoberá analýzou možností konfigurácie sieťových zariadení. Podrobne popisuje konfiguračný protokol NETCONF a jeho rozšírenie o asynchrónne doručovanie upozornení. Práca detailne popisuje otvorenú konfiguračnú platformu Netopeer a špecifiká jej pilotného nasadenia v podobe konfiguračného systému sondy FlowMon. Súčasťou predkladanej práce je návrh a implementácia novej architektúry platformy Netopeer 2.0, ktorá doplňuje funkcionalitu platformy o podporu asynchrónneho doručovania správ. Vytvorené riešenie analyzuje z pohľadu bezpečnosti a navrhuje doporučené nastavenia systému. Práca naväzuje na predchádzajúcu bakalársku prácu autora a na existujúce programové vybavenie vyvinuté vrámci projektu Liberouter.



# Abstract

Master's thesis analyzes available network device configuration options and describes NETCONF configuration protocol and NETCONF event notifications extension in details. It describes Netopeer, open configuration platform developed on Liberouter project, and its pitote deployment as FlowMon probe remote configuration system. Newly designed Netopeer architecture, which adds support for NETCONF event notifications, was verified by reference implementation. Security of the new design and implementation was analyzed, and recommended system settings were provided. This Master's thesis is based on results of previous bachelor's thesis of author and on existing software tools developed by the Liberouter project.



# Kľúčové slová

Netopeer, NETCONF, SSH, konfigurácia sieťových zariadení, manažment počítačových sietí, asynchrónne doručovanie upozornení, XML, FlowMon, NetFlow, Liberouter

# Keywords

Netopeer, NETCONF, SSH, network device configuration, network administration, event notifications, XML, FlowMon, NetFlow, Liberouter


# Citácia

Marek Žižlavský: Netopeer: Konfigurační platforma pro síťová zařízení, diplomová práce, Brno, FIT VUT v Brně, 2010

# Netopeer: Konfigurační platforma pro síťová zařízení

## Prehlásenie

Prehlasujem, že som túto prácu vypracoval samostatne pod vedením pána Ing. Jana Kořeneka.

.....
Marek Žižlavský
25. mája 2010

## Poďakovanie

Ďakujem Ing. Janovi Kořenekovi za odbornú pomoc a vedenie tejto práce. Ďakujem projektu Liberouter za možnosť využívať zariadenia na účely vývoja a testovania programového vybavenia vytvoreného ako súčasť tejto diplomovej práce.

# Obsah


**1 Úvod** **3**

**2 Konfigurácia sieťových zariadení** **5**
2.1 Základné konfiguračné nástroje . . . . . 6
2.2 Protokol SNMP . . . . . 6

**3 NETCONF** **9**
3.1 Architektúra . . . . . 10
3.2 Rozšíriteľnosť . . . . . 11
3.3 Model RPC . . . . . 12
3.4 Práca s konfiguračnými dátami . . . . . 12
3.5 Možnosti využitia protokolu . . . . . 13

**4 Asynchrónne správy protokolu NETCONF** **18**
4.1 Architektúra . . . . . 19
4.2 Operácie spojené so zasielaním oznámení . . . . . 20
4.3 Prúdy udalostí . . . . . 21
4.4 Mechanizmus prelínania . . . . . 22

**5 Platforma Netopeer** **23**
5.1 Architektúra . . . . . 23
5.2 Konfiguračný démon . . . . . 24
5.3 Využitie protokolu NETCONF . . . . . 26
5.4 Webové rozhranie . . . . . 29
5.5 Konfiguračný systém sondy FlowMon . . . . . 32
5.5.1 Webové rozhranie sondy FlowMon . . . . . 33

**6 Návrh platformy Netopeer 2.0** **35**
6.1 Architektúra . . . . . 35
6.2 Konfiguračný démon zariadenia . . . . . 36
6.3 Agent . . . . . 37
6.4 Manažér . . . . . 37
6.5 Subsystém . . . . . 38
6.6 Rozhranie príkazového riadku . . . . . 39
6.7 Webové rozhranie . . . . . 40

**7 Implementácia platformy Netopeer 2.0** **42**
7.1 Referenčná implementácia . . . . . 42
7.2 Knižnice funkcií a tried . . . . . 44

**8 Bezpečnosť platformy Netopeer 2.0** **47**

**9 Záver** **50**

**A Objavovanie prúdov udalostí** **54**

**B UML Diagramy** **55**
B.1 Diagram vykonania NETCONF operácie . . . . . 55
B.2 Diagram asynchrónneho doručovania upozornení . . . . . 56
B.3 Diagram tried knižnice libnetconf . . . . . 57

# Kapitola 1

# Úvod

Súčasne s rastúcou komplexnosťou súčasných sietí, rastú aj nároky na nástroje umožňujúce ich efektívne monitorovanie a spravovanie. Nutnosť používať rôzne proprietárne konfiguračné nástroje na správu a monitorovanie sieťových prvkov od rôznych výrobcov môže viezť k predĺženiu reakčného času zásahu správcu siete v prípade výskytu chyby, alebo počas cieleného útoku. Myšlienka konceptu automaticky spravovaných počítačových sietí naráža na problémy efektívnej komunikácie medzi zariadeniami rôznych výrobcov, neexistujúcu špecifikáciu štandardu ktorý by definoval spôsob konfigurácie zariadení daného typu, ako aj na obmedzené možnosti centrálnej správy konfigurácií jednotlivých zariadení.

Protokol Simple Network Management Protocol, definovaný na prelome 80. a 90. rokov 20. storočia, umožňuje konfiguráciu a monitorovanie sieťových zariadení. Implementácia protokolu pre poskytovanie monitorovacích funkcií je pomerne jednoduchá a rozšírená. Naopak možnosti protokolu v oblasti konfigurácie zariadení sú značne obmedzené a ich implementácia je, vhľadom na ponukanú funkcionalitu, neúmerne zložitá. Tieto vlastnosti protokolu SNMP viedli k jeho širokému nasadeniu výrobcami sieťových zariadení, najmä na účely monitorovania siete. Kvôli náročnosti a problémom spojeným s implementáciou konfiguračných funkcií protokolu, dávajú vyrobci prednosť rôznym proprietárnym nástrojom, zväčša vo forme rozhrania príkazového riadku. Táto skutočnosť umocňuje potrebu definície univerzálneho konfiguračného protokolu, ktorého implementácia by bola jednoduchá a zároveň flexibilná, čím by výrobcov motivovala, tento protokol na konfiguráciu svojich zariadení používať.

Problémy spojené s nasadením SNMP v úlohe univerzálneho konfiguračného protokolu sa snaží riešiť moderný konfiguračný protokol NETCONF[4], ktorého špecifikácia bola publikovaná koncom roku 2006. Protokol podporuje základnú funkcionalitu potrebnú na monitorovanie a správu sieťových zariadení. Jeho návrh počíta s možnosťou jednoduchého doplňovania funkcionality protokolu pomocou mechanizmu definície rozšírení. Rozšírenia môžu byť definované v podobe obecne platných štandardov, ako aj v podobe proprietárnych riešení jednotlivých výrobcov.

Najvýznamnejším štandardným rozšírením funkcionality protokolu je špecifikácia asynchrónneho doručovania upozornení [6], definovaná v polovici roku 2008. Rozšírenie pridáva do protokolu novú funkcionalitu, ktorá umožňuje spravovaným zariadeniam asynchrónne zasielať informácie o udalostiach, ktoré sa v systéme vyskytli. To umožňuje návrh komplexných systémov určených na konfiguráciu a monitorovanie siete, založených na protokole NETCONF.

Cieľom tejto diplomovej práce je naštudovanie problematiky vzdialenej konfigurácie sieťových zariadení, špecifikácie protokolu NETCONF, jeho rozšírenia o asynchrónne doručovanie upozornení a detailné zoznámenie sa s architektúrou otvoreného konfiguračného sys-

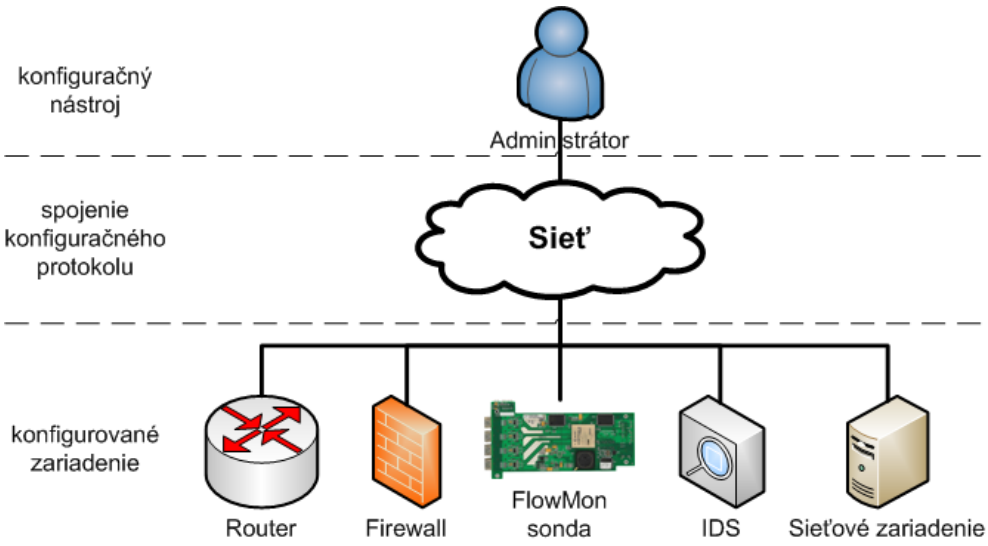
tému *Netopeer*. Systém *Netopeer* je otvorenou implementáciou konfiguračného protokolu NETCONF vyvinutou vrámci projektu Liberouter. Pilotne bol nasadený na projekte pasívnej monitorovacej sondy *FlowMon*, kde slúži na vzdialenú konfiguráciu sondy pomocou webového rozhrania. Táto práca sa zaoberá analýzou možností rozšírenia jej funkcionality o asynchrónne doručovanie upozornení v súlade s požiadavkami definovanými v príslušnom štandarde. Predkladá návrh novej architektúry systému a analyzuje ho z pohľadu bezpečnosti. Správnosť prezentovaného návrhu overuje formou vytvorenia referenčnej implementácie v jazyku C++.

Text predkladanej práce je rozdelený do ôsmych kapitol. V kapitole 2 sú analyzované dostupné nástroje umožňujúce vzdialenú konfiguráciu sieťových zariadení a výhody i nevýhody ich nasadenia v úlohe univerzálneho riešenia na vzdialenú konfiguráciu. Osobitná pozornosť je venovaná protokolu SNMP. Kapitola 3 sa venuje detailnému popisu univerzálneho konfiguračného protokolu NETCONF. Rozšíreniu *NETCONF Event Notifications*, ktoré do protokolu pridáva funkcionalitu asynchrónneho doručovania upozornení, je venovaný text kapitoly 4. Kapitola 5 analyzuje a popisuje architektúru a programové vybavenie konfiguračnej platformy *Netopeer* vo verzii 1.0, vrátane kompletného konfiguračného systému pre NetFlow sondu *FlowMon*, na vývoji ktorého sa autor práce významnou časťou podieľal. Popisu návrhu architektúry konfiguračnej platformy *Netopeer* vo verzii 2.0, vytvoreného ako súčasť tejto diplomovej práce, sa venuje kapitola 6. Referenčná implementácia novej architektúry je popísaná v kapitole 7, vrátane popisu metodiky a výsledkov testovania jej funkcionality. Kapitola 8 sa venuje analýze vytvoreného riešenia z pohľadu bezpečnosti.

# Kapitola 2

# Konfigurácia sieťových zariadení

V oblasti správy sietí bolo v posledných rokoch vyvinutých viacero technológií, konfiguračných protokolov a štandardov. Vývoj smeruje k vytvoreniu nástrojov, ktoré by umožňovali konfiguráciu skupiny sieťových zariadení s využitím jedného rozhrania na správu siete. Cieľom je umožnenie centralizovanej správy a dohľadu nad sieťou. Naplnenie tohto cieľa vyžaduje od rozhrania schopnosť komunikovať s rôznymi typmi zariadení od rôznych výrobcov. To vedie k potrebe definície štandardizovaného konfiguračného protokolu a spôsobu ukladania samotných konfiguračných dát pre jednotlivé typy zariadení.




Obr. 2.1: Základná architektúra konfigurácie sieťových zariadení

Obrázok 2.1 ilustruje typickú architektúru riešenia umožňujúceho centralizovanú správu siete. Administrátor vyžíva rozhranie konfiguračného nástroja, najčastejšie prístupné vo forme webovej aplikácie. Nástroj komunikuje s jednotlivými zariadeniami na sieti pomocou špecializovaného konfiguračného protokolu. Ten musí podporovať možnosť komunikácie s rôznymi typmi sieťových zariadení ako napríklad smerovač, paketový filtér, NetFlow sonda a podobne.

## 2.1 Základné konfiguračné nástroje

Medzi základné konfiguračné nástroje možno zaradiť textové rozhrania používajúce interpretátor príkazového riadku. Tento spôsob konfigurácie je podporovaný väčšinou sieťových zariadení, aj tých najjednoduchších určených pre domácnosti. Zariadenie sprístupňuje správcovi jednoduché konzolové rozhranie, umožňujúce vzdialenú konfiguráciu. V úlohe komunikačného protokolu je vo väčšine prípadov použitý *telnet* [15], prípadne jeho bezpečná varianta v podobe protokolu SSH [16]. Sieťové zariadenia určené na profesionálne nasadenie podporujú aj pripojenie pomocou sériového rozhrania.

Výhodou týchto konfiguračných rozhraní, je použitý spôsob ukladania konfiguračných príkazov v textovej forme. To umožňuje správcom upravovať kompletnú konfiguráciu zariadenia pomocou textového editora, a manuálne udržovať viacero verzii konfigurácie. V súčasnosti však už existujú aj rôzne komplexné riešenia na centralizovanú správu konfiguračných dát, podporujúce jej úpravu cez webové rozhranie, automatické zálohovanie a obnovovanie, ako aj správu konfigurácie viacerých zariadení súčasne. Pokročilé riešenia sú schopné automaticky identifikovať chyby v nastavení, pomocou analýzy vzorov.

Hlavnou nevýhodou konfigurácie pomocou príkazovej riadky je vzájomná inkompatibilita syntaxe a príkazov jazyka interpretátorov, používaných na zariadeniach od rôznych výrobcov. Táto skutočnosť spomaľuje vývoj pokročilých nástrojov, prípadne obmedzuje možnosti ich použitia iba na konfiguráciu zariadení od konkrétneho výrobcu.

Moderné sieťové zariadenia umožňujú intuitívnu konfiguráciu pomocou webového rozhrania. Užívateľ má možnosť nastaviť parametre zariadenia pomocou webového prehliadača. Tento spôsob konfigurácie je rozšírený najmä medzi zariadeniami určenými pre domácnosti, kancelárie a malé podniky. Nevýhody riešenia sa začnú prejavovať najmä v prípade nutnosti zmeny nastavenia na viacerých zariadeniach súčasne, pretože webové rozhrania dodávané jednotlivými výrobcami väčšinou nie sú vzájomne kompatibilné. Ďalšou nevýhodou je obmedzená funkcionalita webových rozhraní. Pokročilé, prípadne málo používané nastavenia a funkcie bývajú často dostupné len s využitím rozhrania príkazového riadku.

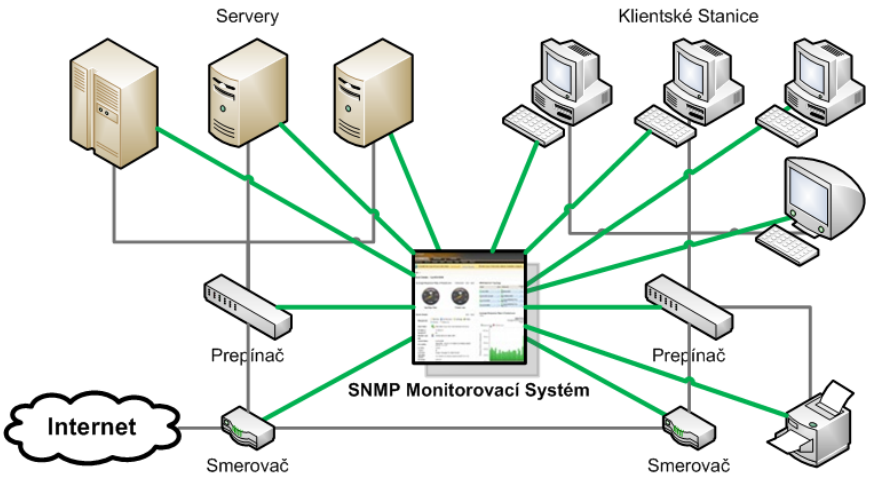
## 2.2 Protokol SNMP

Koncom 80. rokov 20. storočia vyvrcholila snaha organizácie IETF, o vytvorenie štandardizovaného protokolu na správu a monitorovanie siete, k definíciou protokolu *Simple Network Management Protocol*. Protokol SNMP je v súčastnosti zaužívaný najmä ako štandard na účely monitorovania stavu siete, kde je výhodou jeho jednoduchá bezstavová architektúra a existencia štandardizovaných dátových štruktúr na ukladanie informácií o stave zariadenia, ako aj široká podpora protokolu zo strany výrobcov zariadení.

Typické nasadenie protokolu SNMP na monitorovanie siete ilustruje obrázok 2.2. Nástroj na monitorovanie siete využíva protokol SNMP na získavanie informácií o stave zariadení v sieti. Nástroj môže monitorovať napríklad aktuálne vyťaženie procesoru, využitie operačnej pamäte, vyťaženosť jednotlivých sieťových rozhraní, teplotu a spotrebu zariadení a podobne. Pokročilé nástroje dokážu získané údaje automaticky analyzovať a v prípade zistenia problémov na sieti, správcu upozorniť, napríklad zaslaním SMS správy.

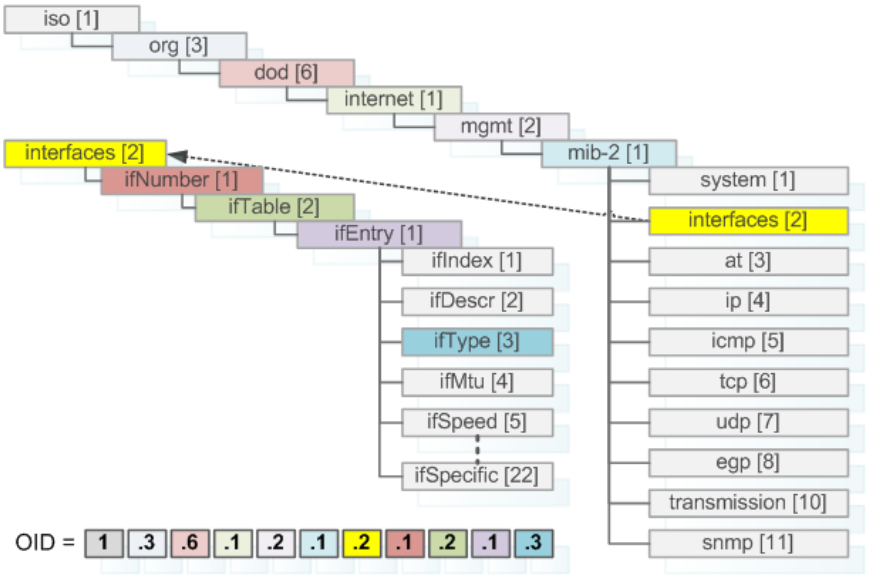
Protokol SNMP používa na ukladanie informácií o stave a aktuálnej konfigurácií zariadenia dátovú štruktúru pomenovanú *Management Information Base*. Spôsob uloženia informácií v MIB databázach, vo forme rozvetvenej stromovej štruktúry, je znázornený na obrázku 2.3. V rámci špecifikácie protokolu boli definované štandardné MIB moduly, spoločné pre väčšinu zariadení daného typu. Napríklad pomocou modulu IF-MIB[17] je možné získať podrobné informácie o jednotlivých rozhraniach sieťového zariadenia. Je možné získať in-

formácie o počte prenesených paketov, počte chybných paketov, názve a type rozhrania, prenosovej rýchlosti a pod. Ďalším často používaným modulom je HR-MIB[18] obsahujúci informácie o samotnom zariadení, napríklad čas od posledného reštartu zariadenia, aktuálne využitie CPU, pamäte a podobne. Tieto štandardizované moduly sú podporované väčšinou významných výrobcov sieťových zariadení.




Obr. 2.2: Monitorovanie siete pomocou protokolu SNMP

Okrem štandardných modulov existuje aj množstvo proprietárnych MIB modulov vyvinutých samotnými výrobcami zariadení na podporu ich vlastných nástrojov určených na správu siete. Aj vďaka tomu sa protokol SNMP stal významným zdrojom dát najmä pre systémy určené na koreláciu udalostí, detekciu krízových stavov a anomálií, rôzne dohľadové systémy a pod.




Obr. 2.3: Stromová dátová štruktúra databázy MIB

Nasadeniu SNMP vo forme univerzálneho konfiguračného protokolu bráni viacero faktorov. Protokol bol navrhnutý ako „programátorské" rozhranie medzi zariadením a aplikáciou používanou na jeho správu. Preto je jeho využitie bez ďalšieho špecializovaného nástroja značne komplikované.

Štandardizované MIB moduly často neobsahujú zapisovateľné položky, ktoré by mohli byť využité na konfiguráciu zariadení. To viedlo k situácií, že väčšina zaujímavých konfigurovateľných položiek je definovaná v rámci proprietárnych MIB modulov jednotlivých výrobcov zariadení. Tým sa stráca výhoda štandardizovanej formy uloženia dát vo forme spoločných MIB modulov.

Protokol má problémy vyrovnať sa s veľkým počtom objektov v MIB databázach. Jeho výkon je pri získavaní malých objemov dát z veľkého počtu zariadení dostatočný, je však nevhodný na získavanie veľkého objemu dát z malého počtu zariadení, ako napríklad pri získavaní rozsiahlych smerovacích tabuliek zo smerovačov, prípadne informácií o veľkom počte virtualizovaných sieťových rozhraní.

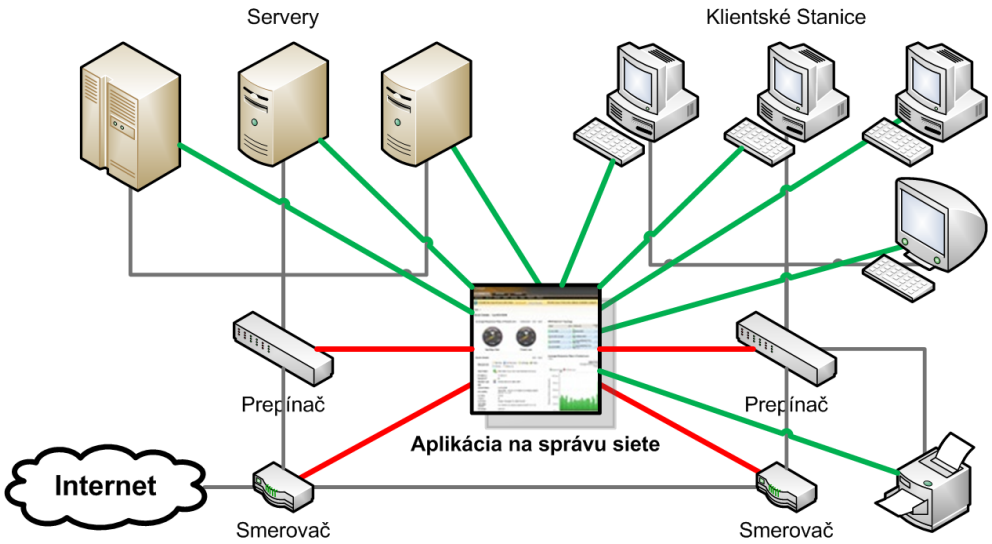
Použitý komunikačný a dátový model spôsobujú, že implementácia konfigurácie pomocou MIB je pre výrobcov komplikovanejšia oproti použitiu klasickej konfigurácie pomocou interpretátora príkazovej riadky. Protokol nedefinuje základné operácie na jednoduché získavanie a reprodukciu konfiguračných dát ako napríklad návrat k predchádzajúcej, prípadne záložnej, konfigurácií. Neumožňuje udržovať viacero verzii konfigurácie. Problematická je aj identifikácia zapisovateľných objektov, využiteľných na účely konfigurácie zariadenia, spomedzi objektov databázy MIB ako aj samotný vysoko špecializovaný systém ich pomenovania a uloženia v stromovej štruktúre.

Správcovia sietí preferujú úlohovo orientovaný (task-oriented) prístup ku konfigurácií zariadení, protokol SNMP však vytvára konfiguračný model orientovaný na dáta (data-centric), čo vyžaduje po nástrojoch určených na správu vytvárať často netriviálne mapovania medzi jednotlivými modelmi.

# Kapitola 3

# NETCONF

Modelové využitie protokolu NETCONF[4] pokročilou aplikáciou na správu siete je znázornené na obrázku 3.1. Aplikácia využíva protokol na správu konfiguračných dát na aktívnych sieťových prvkoch. Na obrázku je toto využitie znázornené červenou farbou. Zároveň však protokol využíva aj na monitorovanie stavu klientských počítačov, serverov a jednej tlačiarne. Tento spôsob využitia je zvýraznený zelenou farbou. Použitie protokolu umožňuje aplikácií použiť jednotný spôsob komunikácie, napriek tomu, že komunikuje so zariadeniami rôznych typov od rôznych výrobcov. Aplikácia musí poznať konkrétne dátové modely, ktoré jednotlivé zariadenia používajú na reprezentáciu svojej konfigurácie, prípadne stavových informácií. Situáciu jej však uľahčuje fakt, že protokol po zariadeniach vyžaduje, aby tieto informácie poskytovali v štandardizovanej podobe definovanej jazykom XML[19].

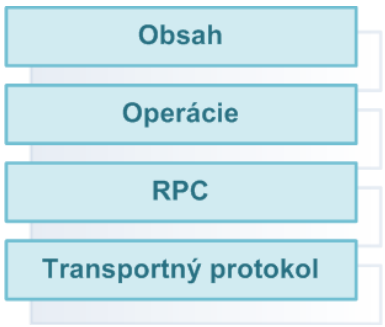



Obr. 3.1: Nasadenie protokolu NETCONF na správu siete

Konfiguračný protokol NETCONF, v základnej podobe (bez dodatočných rozšírení), poskytuje funkcionalitu na základnú správu konfigurácie sieťových zariadení a monitorovanie ich stavu. Definuje napríklad operácie na ukladanie, zmenu a mazanie konfiguračných dát. Konfiguračné dáta zariadenia ako aj riadiace príkazy protokolu sú kódované v jazyku XML. O odosielanie požiadaviek na vykonávanie operácií na vzdialený systém sa stará vrstva protokolu RPC[20], ktorý slúži na vzdialené volanie procedúr. Zabezpečenie integrity a utajenia správ pri prenose, má na starosti niektorý z podporovaných transportných protokolov.

## 3.1 Architektúra

Protokol používa osvedčený model komunikácie klient-server založený na vzdialenom volaní procedúr (RPC). Klientom sa rozumie aplikácia, prípadne skript bežiaci ako súčasť nástroja na správu siete. Server je typicky realizovaný agentom, spusteným na samotnom sieťovom zariadení.




Obr. 3.2: Vrstvy protokolu NETCONF

Architektúru protokolu je možné reprezentovať vo forme štyroch logických vrstiev, ktorých vertikálne usporiadanie ilustruje obrázok 3.2. Vrstva transportného protokolu poskytuje bezpečný a spoľahlivý prenosový kanál, ktorý je používaný vrstvou RPC na odosielanie požiadavkov na vykonanie jednotlivých NETCONF operácií. Vrstva operácií definuje sadu príkazov, ktoré slúžia na prácu s konfiguračnými dátami zariadenia. Samotné konfiguračné dáta reprezentované v jazyku XML, sú vrámci architektúry zaradené do vrstvy obsahu.

**Vrstva transportného protokolu** zabezpečuje spoľahlivý komunikačný kanál medzi administrátorskou aplikáciou (manažér) a konfigurovaným zariadením (agent). Vrstva môže byť implementovaná ľubovoľným protokolom, ktorý spĺňa základnú sadu, najmä bezpečnostných požiadaviek protokolu NETCONF:

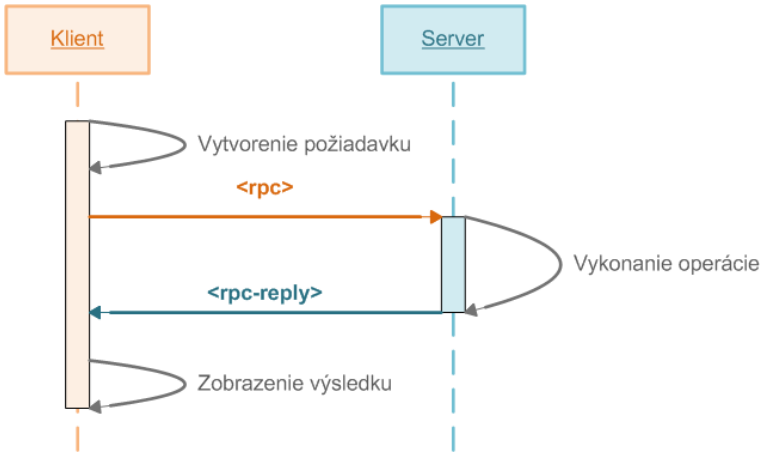
- spoľahlivý spojovaný prenos dát
- zabezpečenie integrity dát počas prenosu
- zabezpečenie utajenia dát počas prenosu
- služby overenia identity komunikujúcich strán

Protokol NETCONF vyžaduje minimálne implementáciu podpory komunikačného protokolu SSH[16], podpora ďalších komunikačných protokolov (BEEP[22], SOAP[21]) je voliteľná. Pri ukončení spojenia na úrovni transportnej vrstvy musí byť systém schopný automaticky uvoľniť všetky zdroje rezervované týmto spojením.

**Vrstva vzdialeného volania procedúr** slúži na odosielanie požiadavkov na vykonanie jednotlivých operácií na vzdialenom systéme. Používa mechanizmus založený na princípe vzdialeného volania procedúr (RPC). Technológia RPC obecne slúži na volanie procedúr a funkcií definovaných v inom adresovom priestore (v inom procese, prípadne na vzdialenom systéme dostupnom pomocou sieťového rozhrania), bez nárokov na riešenie detailov realizácie samotného komunikačného spojenia programátorom aplikácie, ktorá volanie uskutočňuje. O medziprocesovú komunikáciu sa stará práve systém RPC.

Protokol NETCONF používa jednoduchý systém RPC založený na posielaní správ zakódovaných v jazyku XML. Klient vytvorí požiadavok na vykonanie operácie, odošle ho serveru, ktorý sa následne pokúsi vykonať požadovanú operáciu. Server je povinný doručiť

klientovi informáciu o výsledku požadovanej operácie, prípadne o chybe, ktorá nastala počas jej spracovania. Správy protokolu RPC sú kódované v jazyku XML. Ilustrácia princípu komunikácie pomocou protokolu RPC je znázornená na obrázku 3.3.




Obr. 3.3: Princíp vzdialeného volania procedúr

**Vrstva operácií** protokolu definuje iba základnú množinu operácií na prácu s konfiguráciou zariadenia. Ďalšie operácie je možné pridávať pomocou definície rozšírení, viď kapitola 3.2. Základnými operáciami protokolu NETCONF sú operácie <get>, <get-config>, <edit-config>, <copy-config>, <delete-config>, <lock>, <unlock>, <kill-session> a <close-session>. Operácie sú podrobnejšie popísané v kapitole 3.5.

**Vrstva obsahu** je reprezentuje konkrétne konfiguračné dáta zariadenia, kódované v jazyku XML. Špecifikácia protokolu sa nezaoberá definíciou konkrétnej syntaxe a sémantiky konfiguračných dát pre jednotlivé typy sieťových zariadení, ale ponecháva jej definíciu v réžii výrobcov daných zariadení, prípadne doplňujúcich dokumentov RFC.

## 3.2 Rozšíriteľnosť

Tvorcovia špecifikácie protokolu kládli dôraz na zachovanie možnosti jednoduchého rozširovania jeho základnej funkcionality o nové operácie a schopnosti. Protokol definuje štandardizovaný spôsob, ako takéto nové operácie do protokolu začleniť pomocou mechanizmu schopností - *capabilities*. Pri ustanovovaní NETCONF spojenia si komunikujúce strany presne definovaným spôsobom vymenia zoznam podporovaných schopností a následne pri komunikácií využívajú iba tie rozširujúce operácie, ktoré sú podporované oboma účastníkmi spojenia. Schopnosti môžu byť publikované ako štandardy, prípadne sa môže jednať iba o proprietárne rozšírenia konkrétneho výrobcu.

Protokol identifikuje jednotlivé schopnosti pomocou URI[23], prípadne URN[24]. Pravidlá na tvorbu identifikátorov schopností sú špecifikované v dokumente RFC 3553. Základné schopnosti definované ako súčasť definície protokolu NETCONF majú identifikátor v tvare:

```
urn:ietf:params:netconf:capability:{názov}:1.0
```

kde {názov} je meno schopnosti. Na tieto schopnosti sa v texte práce často odkazuje pomocou skrátenej notácie *:názov*, v samotných správach protokolu je však vždy vyžadované použitie plného identifikátoru schopnosti.

## 3.3 Model RPC

Požiadavok na volanie vzdialenej procedúry je reprezentovaný elementom <rpc>. Parametre volanej procedúry sú taktiež kódované v jazyku XML, v prípade operácií protokolu NETCONF sa väčšinou jedná o samotné konfiguračné dáta uložené v jazyku XML.

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <názov-netconf-operácie>
    <!-- parametre... -->
  </názov-netconf-operácie>
</rpc>
```

Odpoveď servera je reprezentovaná elementom <rpc-reply>. Na identifikáciu odpovede slúži povinný atribút *message-id*, zasielaný klientom ako súčasť elementu <rpc>. Špecifikácia protokolu NETCONF vyžaduje aby server ako súčasť RPC odpovede zachoval aj všetky prípadné ďalšie atribúty elementu <rpc> zaslané klientom, v nezmenenej podobe.

Odpoveď servera na požiadavok na vykonanie operácie, ktorá ako výsledok vracia konfiguračné dáta, je reprezentovaný správou <rpc-reply> obsahujúcou element <data>. Obsah elementu <data> tvoria samotné konfiguračné dáta v jazyku XML.

```
<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <data>
    <!-- konfiguračné dáta... -->
  </data>
</rpc>
```

V prípade výskytu chyby vracia server správu <rpc-reply> obsahujúcu element <rpc-error>, ktorý môže obsahovať detailný popis chyby.

```
<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <rpc-error>
    <!-- informácia o chybe... -->
  </rpc-error>
</rpc>
```

V prípade úspešného vykonania operácie, ktorá nevracia ako výsledok žiadne dáta, vracia server správu <rpc-reply> obsahujúcu element <ok/>.

```
<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <ok/>
</rpc>
```

Špecifikácia protokolu NETCONF povoľuje klientovi zaslanie ďalšieho požiadavku RPC, ešte pred doručením odpovede na predchádzajúci požiadavok a vyžaduje aby server odpovedal na požiadavky presne v poradí v akom mu boli doručené.

## 3.4 Práca s konfiguračnými dátami

Informácie, ktoré je možné získať zo zariadenia pomocou protokolu NETCONF možno rozdeliť do dvoch tried:

**Konfiguračné dáta** - sú tvorené množinou zapisovateľných dát, ktoré sú potrebné na transformáciu systému z jeho počiatočného (východzieho) stavu do aktuálneho (zvoleného) stavu.

**Stavové informácie** - obsahujú rozširujúce informácie o stave zariadenia a štatistické údaje získané zariadením počas doby jeho prevádzky a sú určené iba na čítanie.

Konfiguračné dáta sú uložené vo forme dokumentov v jazyku XML. Protokol definuje jedno základné úložisko pomenované *running*. Toto úložisko obsahuje aktuálne konfiguračné dáta, podľa ktorých sú parametre zariadenia momentálne nastavené. Rozšírenia *:startup* a *:candidate* definujú ďalšie nezávislé konfiguračné úložiská.

Úložisko *startup* obsahuje konfiguráciu, podľa ktorej sú parametre zariadenia nastavené ihneď po štarte (prípadne reštarte) zariadenia, bez ohľadu na zmeny vykonané v úložisku *running*.

Úložisko *candidate* slúži ako pracovné úložisko, na ukladanie priebežných zmien v konfiguračných dátach, ktoré môžu byť následne uložené späť do úložiska *running*. Je vytvorené automaticky pri ustanovení nového NETCONF spojenia ako kópia aktuálneho stavu úložiska *running*.

Všetky vyššie spomínané úložiská jedného zariadenia sú spoločné pre všetky s simultánne nadviazané spojenia. Preto je doporučené využívať operácie na uzamykanie (<lock> a <unlock>) pri akejkoľvek manipulácií s dátami v úložisku.

Niektoré operácie, napríklad <get-config>, umožňujú pri získavaní dát z konfiguračného úložiska aplikovať užívateľom definovateľné filtre, ktoré slúžia na výber dielčich častí týchto dát. Základný princíp filtrovania vychádza z hierarchickej reprezentácie štruktúry dát a jeho uloženia v podobe dokumentu jazyka XML. Pomocou filtra, ktorého definícia je taktiež zapísaná v jazyku XML, je možné presne špecifikovať jednotlivé podstromy celkovej konfigurácie zariadenia, ktoré sa majú ďalej spracovať určenou operáciou. Na výber podstromov, ktoré sa majú zahrnúť do výsledku, je možné pri definícii filtra použiť viacero metód.

**Výber pomocou špecifikácie elementu** - základný spôsob výberu podstromu. Výsledkom filtrovania je podstrom, ktorého vrcholom je elementu z filtra zodpovedajúci element konfigurácie.

**Výber pomocou menných priestorov** - umožňuje špecifikovať menný priestor, do ktorého musia byť zaradené elementy konfigurácie, aby sa stali súčasťou výsledku. Tento spôsob filtrovania je nutné kombinovať s výberom pomocou špecifikácie elementu.

**Výber podľa hodnoty atribútu** - ak niektorý z elementov definovaných filtrom obsahuje atribút (alebo aj viacej atribútov), výsledkom filtrácie je podstrom, ktorého koreňový element zodpovedá tomto elementu filtra a súčasne má definované rovnaké atribúty so zodpovedajúcimi hodnotami, ako sú hodnoty atribútov špecifikovaných v elemente definovanom ako súčasť filtra.

**Výber podľa obsahu elementu** - výber pomocou zadania konkrétneho textového obsahu určitého elementu.

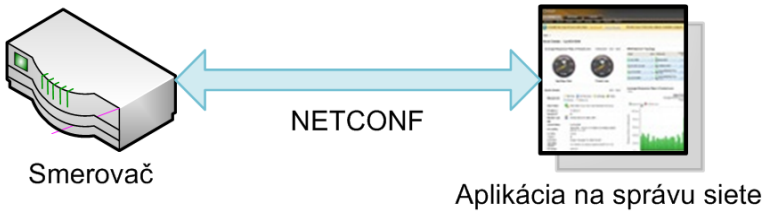
Ak zariadenie podporuje rozšírenie *:xpath* protokolu NETCONF, je možné na definíciu filtra použiť výrazy zapísané v jazyku XPath[25]. Výsledkom aplikácie takéhoto filtra na konfiguračné dáta je podmnožina dát zodpovedajúca danému XPath výrazu.

## 3.5 Možnosti využitia protokolu

V základnej podobe protokol poskytuje funkcionalitu zabezpečeného komunikačného kanálu, ktorý slúži na prenos citlivých dát v podobe konfigurácie zariadenia, prípadne stavových informácií, medzi samotným zariadením a aplikáciou určenou na správu siete.

Typické nasadenie protokolu NETCONF je zobrazené na obrázku 3.4. Protokol aplikácií poskytuje prostriedky na riadenie stavu spojenia, získavanie aktuálnej konfigurácie zariadenia, nahrávanie novej konfigurácie, získanie stavových informácií, editovanie existujúcej konfigurácie priamo na zariadení, prípadne zmazanie vybranej konfigurácie. Tým, že protokol

podporuje prácu s viacerými konfiguračnými úložiskami, je aplikácií umožnené udržovať viac verzií konfigurácie, priamo na zariadení.




Obr. 3.4: Využitie protokolu NETCONF na správu zariadenia

Pred volaním samotných operácií protokolu, je potrebné nadviazať spojenie so vzdialeným systémom. Proces **ustanovovania spojenia** musí prebehnúť ihneď po nadviazaní spojenia na úrovni transportnej vrstvy. V rámci neho prebieha vyjednávanie o podporovaných schopnostiach, ktoré sú jednotlivými účastníkmi podporované. Skôr ako je možné začať používať jednotlivé operácie protokolu NETCONF, musí každá strana odoslať správu obsahujúcu element <hello>, ktorého obsah je tvorený zoznamom ňou podporovaných schopností. Na úspešné nadviazanie spojenia je vyžadované aby obidvaja účastníci spojenia implementovali minimálne sadu základných operácií protokolu NETCONF, definovanú v rámci schopnosti *"urn:ietf:params:netconf:base:1.0"*. Súčasťou správy <hello> zaslanej serverom je aj jedinečný identifikátor spojenia odoslaný ako hodnota elementu <session-id>. Správa <hello> odosielaná klientom nesmie definíciu elementu <session-id> obsahovať. Príklad správy <hello> zaslanej serverom:

```
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <capabilities>
    <capability>urn:ietf:params:netconf:base:1.0</capability>
    <capability>urn:ietf:params:netconf:capability:startup:1.0</capability>
    <capability>http://liberouter.org/flowmon/1.0/management</capability>
  </capabilities>
  <session-id>4</session-id>
</hello>
```

Po úspešnom nadviazaní spojenia, je možné začať využívať jednotlivé operácie protokolu. Medzi základné operácie patrí operácia **<get-config>**, ktorá slúži na získanie obsahu vybraného konfiguračného úložiska, prípadne jeho časti, špecifikovanej príslušným filtrom. Parametre operácie sú:

**source** - názov dotazovaného konfiguračného úložiska, napr. *running*.

**filter** - element filtra určujúci podstrom konfiguračného XML, ktorý bude vrátený ako výsledok operácie. Operácia podporuje dva typy filtrovania, určené hodnotou atribútu *type* elementu <filter>. Hodnota *"subtree"* označuje štandardne dostupné filtrovanie pomocou špecifikácie elementu, viď kapitola 3.4.

Na zmenu obsahu konfiguračného úložiska slúži operácia **<edit-config>**. Operácia nahrá celú alebo zadanú časť konfigurácie na určené miesto v cieľovej konfigurácií. Nepracuje teda s konfiguračným úložiskom iba ako s celkom, ale umožňuje vybrať časť konfiguračných dát v ňom obsiahnutých a tie upraviť, prípadne iba pridať do úložiska nové dáta. Každý element v zdrojovej konfigurácií určenej parametrom <config> môže mať definovaný atribút *operation*, ktorého hodnota určuje akciu, ktorú operácia <edit-config> s elementom zdrojovej a cieľovej konfigurácie vykoná. Prípustné hodnoty atribútu *operation* sú:

**merge** - konfiguračné dáta elementu zdrojovej a cieľovej konfigurácie sú zlúčené. Zlučovanie prebieha rekurzívne - pre všetkých potomkov aktuálneho uzlu sa do cieľovej konfigurácie doplnia všetky elementy, ktoré v nej chýbajú. Zhodné elementy sú prepísané elementmi obsiahnutými v zdrojovej konfigurácií a elementy, ktoré sa nachádzajú iba v cieľovej konfigurácií sú ponechané bez zmeny.

**replace** - element v zdrojovej konfigurácií nahradí jemu prislúchajúci element v cieľovej konfigurácií. Ak zodpovedajúci element v cieľovej konfigurácií neexistuje, je generovaná správa o chybe.

**create** - nový element zdrojovej konfigurácie je pridaný do cieľovej konfigurácie. Podmienkou je, že zodpovedajúci element v cieľovej konfigurácií nesmie existovať, inak je generovaná správa o chybe.

**delete** - odpovedajúci element je z cieľovej konfigurácie odstránený. Ak element v cieľovej konfigurácií neexistuje, je generovaná správa o chybe.

Operácia <edit-config> umožňuje ovplyvniť spôsob, akým je zmena vykonaná pomocou nastavenia parametrov:

**target** - názov konfiguračného úložiska, ktorého obsah má byť zmenený. Napr. *running* alebo *candidate*.

**default-operation** - špecifikuje východziu operáciu, v prípade, že element zdrojovej konfigurácie nemá definovaný atribút *operation*. Parameter je nepovinný. V prípade, že jeho hodnota nie je definovaná, použije sa východzia hodnota *merge*.

**test-option** - parameter je možné použiť iba v prípade, že zariadenie podporuje rozšírenie *:validate*. Hodnota *test-then-set* spôsobí, že operácia <edit-config> bude vykonaná, iba v prípade, že modifikované dáta budú validné. Hodnota *set* vynúti zmenu konfigurácie bez ohľadu na výsledok predošlého validačného procesu.

**error-option** - parameter môže nadobúdať tri rôzne hodnoty. Hodnota *stop-on-error* zastaví vykonávanie operácie <edit-config> pri prvom výskyte chyby a je východzou hodnotou parametra. Hodnota *continue-on-error* spôsobí, že sa bude pokračovať v zmenách v konfigurácií, záznam o chybách bude uchovaný a následne sa ako výsledok operácie odošle negatívna odpoveď protokolu RPC. Hodnota *rollback-on-error* pri výskyte chyby zaručí, že obsah konfiguračného úložiska bude navrátený do pôvodného stavu. Na využitie tejto možnosti je nutné aby agent podporoval rozšírenie *:rollback-on-error*.

**config** - konfiguračné dáta v jazyku XML používané ako zdrojové dáta operácie.

Na nahrávanie konfigurácie do vybraného konfiguračného úložiska zariadenia, slúži operácia **<copy-config>**. Operácia umožňuje jednak nahratie novej konfigurácie, vytvorenej pomocou aplikácie určenej na správu siete, do vybraného úložiska zariadenia, ako aj kopírovanie konfigurácie medzi jednotlivými úložiskami na zariadení. To je možné využiť napríklad na prekopírovanie obsahu úložiska *run*ning do úložiska *startup* a zabezpečiť tak aplikovanie požadovanej konfigurácie aj po reštarte zariadenia. Operácia vytvorí alebo nahradí kompletný obsah cieľového konfiguračného úložiska obsahom zadaného zdrojového konfiguračného úložiska. Ak cieľové úložisko už existuje, je jeho obsah prepísaný, inak je vytvorené nové.

**target** - názov konfiguračného úložiska, ktoré slúži ako cieľ operácie.

**source** - názov zdrojového konfiguračného úložiska, prípadne element <config> priamo obsahujúci konfiguračné dáta.

Správca siete niekedy potrebuje vymazať existujúcu konfiguráciu zariadenia. Aplikácia na správu siete môže na tento účel použiť operáciu **<delete-config>**. Operácia slúži na kompletné zmazanie obsahu vybraného konfiguračného úložiska. Úložisko *running* nie je možné pomocou operácie vymazať. Parametre operácie:

**target** - názov konfiguračného úložiska, ktoré má byť operáciou vymazané.

Nasadenie protokolu NETCONF na účely monitorovania siete, vyžaduje podporu na získavanie stavových informácií zariadenia. Na túto úlohu je možné použiť operáciu **<get>**, ktorá podobne ako operácia <get-config> slúži na získanie obsahu konfiguračného úložiska, ale na rozdiel od nej umožňuje získať iba obsah úložiska *running*. Konfigurácia je následne doplnená o stavové informácie a štatistiky zozbierané zariadením počas doby jeho prevádzky. Operácia podporuje filtrovanie dát pomocou parametra *filter*, ktorého význam je rovnaký ako v prípade operácie <get-config>.

Každý správne navrhnutý komunikačný protokol by mal definovať korektné ukončenie spojenia. Protokol NETCONF na tento účel definuje operáciu **<close-session>**, po zavolaní ktorej je agent povinný uvoľniť všetky zdroje asociované so spojením a odomknúť všetky úložiská zamknuté pomocou operácie <lock>. Každý ďalší požiadavok od klienta obdržaný v rámci spojenia, je po volaní operácie <close-session> ignorovaný.

Niekedy môže nastať situácia, keď je potrebné ukončiť iné, simultánne prebiehajúce spojenie so zariadením. Napríklad v situácií, keď správca potrebuje náhle zmeniť konfiguráciu zariadenia, ale tá je uzamknutá iným spojením. NETCONF tento problém rieši pomocou operácie **<kill-session>**, ktorej zavolanie zabezpečí násilné ukončenie vybraného spojenia. Server po obdržaní žiadosti ukončí všetky prebiehajúce operácie spojenia a uvoľní všetky zdroje používané týmto spojením. Parametre operácie:

**session-id** - identifikátor spojenia, ktoré má byť prerušené. Operáciou nie je možné násilne ukončiť spojenie s rovnakým identifikátorom ako má spojenie, ktoré operáciu zavolalo.

Obsah jednotlivých konfiguračných úložísk zariadenia je prístupný všetkým simultánne nadviazaným spojeniam. Aby bolo možné predísť nečakaným zmenám v konfigurácií, je potrebné definovať mechanizmus ich uzamykania. Na tento účel slúži operácia **<lock>**, ktorá uzamkne vybrané konfiguračné úložisko. Parametre operácie:

**target** - Názov konfiguračného úložiska, ktoré má byť operáciou uzamknuté.

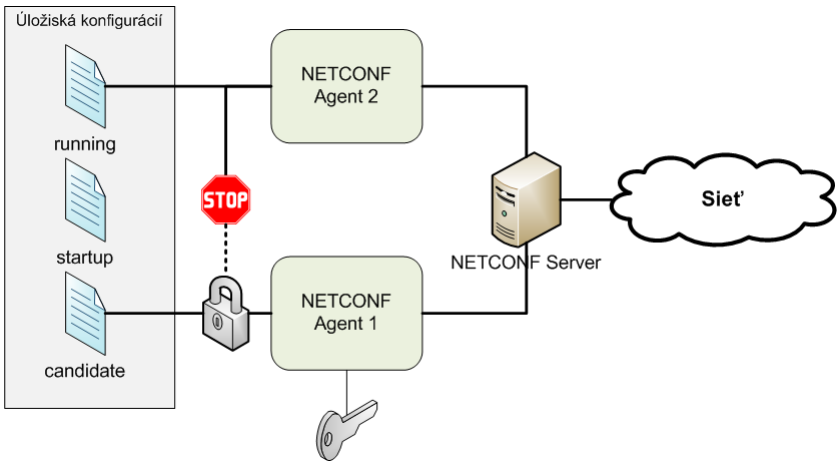
Operáciou možno uzamknúť iba úložisko, ktoré nie je aktuálne zamknuté. V prípade pokusu o zamknutie už uzamknutého úložiska, je generovaná správa o chybe, ktorej súčasťou nutne

musí byť aj identifikácia spojenia, ktoré aktuálne drží zámok. Táto identifikácia môže byť následne využitá na násilné ukončenie daného spojenia operáciou <kill-session>.

Po dokončení práce s konfiguračným úložiskom, je potrebné umožniť ostatným aktívnym spojeniam jeho opätovné využívanie. Na tento účel slúži operácia **<unlock>**, ktorá odomkne konfiguračné úložisko, uzamknuté volaním operácie <lock>. Klient nemá právo odomknúť úložisko, ktoré sám nezamkol. Parametre operácie:

**target** - Názov konfiguračného úložiska, ktoré má byť operáciou odomknuté.

Vyššie popísaný systém fungovania zámkov ilustruje obrázok 3.5.




Obr. 3.5: Princíp zamykania konfiguračných úložísk

# Kapitola 4

# Asynchrónne správy protokolu NETCONF

Protokol NETCONF, tak ako bol v základnej podobe špecifikovaný, podporuje iba synchrónne volanie operácií, čo uspokojuje potreby kladené na konfiguračný protokol z pohľadu správy a práce s konfiguračnými dátami zariadenia. Obmedzenie protokolu na synchrónne volanie operácií však neumožňuje jeho využitie na prenos informácií o asynchrónnych udalostiach systému, ktoré je nutné na umožnenie jeho nasadenia v podobe univerzálneho protokolu na správu sieťových zariadení. Toto obmedzenie sa protokol snaží odstrániť pomocou definície doplnkových rozšírení.

*NETCONF Event Notifications*[6] je anglický názov špecifikácie, ktorá rozširuje funkcionalitu protokolu o asynchrónne doručovanie správ - oznámení o udalostiach v systéme. Schopnosť bola pomenovaná identifikátorom *urn:ietf:params:netconf:capability:notification:1.0.*

Dokument RFC 5277 definuje nasledujúce pojmy:

**Udalosť** - niečo čo sa vyskytne a môže mať pre pozorovateľa istý konkrétny význam, napríklad zmena v konfigurácií, výskyt chyby, zmena stavových informácií, dosiahnutie prahovej hodnoty nejakého parametra, externý vstup do systému a pod.

**Opakovanie (replay)** - schopnosť na vyžiadanie odoslať, respektíve znova odoslať predchádzajúce udalosti, ktoré boli uložené v logu. Táto funkcionalita je implementovaná na strane servera a je vyvolávaná klientom.

**Prúd (stream)** - prúd udalostí je tvorený množinou oznámení o udalosti - notifikačných správ, ktoré zodpovedajú nejakým vopred daným kritériám. Prúd udalostí je dostupný pre klientov protokolu NETCONF, ktorý si môžu zaregistrovať odber správ vyskytujúcich sa v tomto prúde.

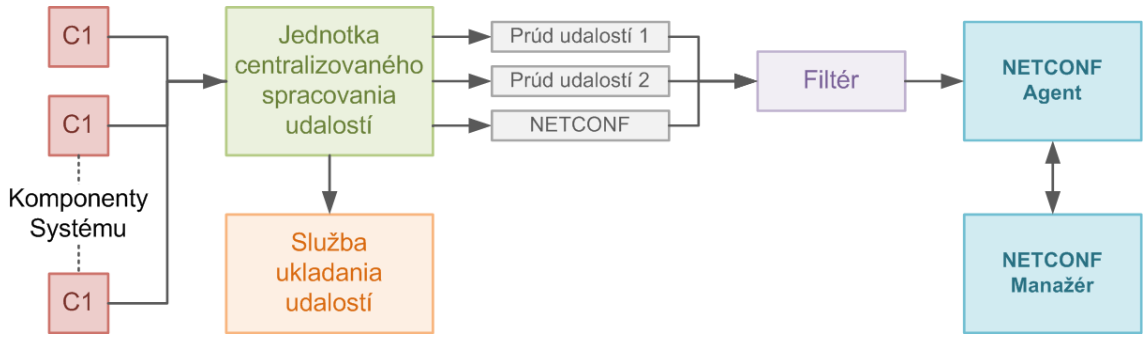
**Filter** - parameter určujúci podmnožinu množiny všetkých udalostí takú, že správy v nej obsiahnuté sú pre klienta zaujímavé. Filter môže byť zložený.

Špecifikácia asynchrónneho zasielania oznámení si kladie za cieľ splnenie nasledovných požiadaviek:

- možnosť reprezentovať oznámenia s využitím rovnakého dátového modelu aký je použitý na reprezentáciu samotnej konfigurácie.
- riešenie by malo podporovať rozumnú veľkosť notifikačných správ.
- prenos notifikačných správ by mal prebiehať po spojovanom kanále.
- presne špecifikovaný mechanizmus prihlasovania sa k odberu oznámení iniciovaný klientom.
- možnosť filtrácie notifikačných správ na strane servera.
- notifikačné dáta by mali byť samopopisné - využitie možností jazyka XML.
- server by mal mať schopnosť oznámenia lokálne ukladať a v prípade potreby ich znova odoslať klientom (replay).

## 4.1 Architektúra

Referenčná architektúra systému na zasielanie asynchrónnych upozornení, definovaná v rámci špecifikácie rozšírenia *:notifications*, je znázornená na obrázku 4.1. Návrh predpokladá existenciu viacerých komponentov, ktoré môžu byť zdrojom udalostí v systéme. Udalosti sú následne centrálne spracovávané a rozdeľované do jednotlivých prúdov udalostí, prípadne môžu byť ukladané do perzistentného úložiska, aby mohli byť klientom oneskorene doručené. Na udalosti v prúde udalostí si môže klient vynútiť aplikovanie filtra, ktorý zabezpečí odfiltrovanie všetkých správ o ktorých zasielanie nemá klient záujem.




Obr. 4.1: Referenčná architektúra systému zasielania oznámení

Návrh vyžaduje na uloženie informácií o udalostiach použitie jazyka XML. To umožňuje jednoduché začlenenie tohto nového typu správ do existujúcej architektúry, navrhnutej ako súčasť špecifikácie protokolu NETCONF. Zároveň sa tým znižujú nároky kladené na systémy implementujúce toto rozšírenie, ktoré by inak boli spojené s nutnosťou podpory nového formátu na ukladanie dát.

## 4.2 Operácie spojené so zasielaním oznámení

Rozšírenie *:notifications* pridáva k základným operáciám protokolu NETCONF nové operácie umožňujúce klientovi registrovať si požiadavku na zasielanie oznámení a serveru tieto oznámenia klientovi odosielať.

Klient musí o zasielanie asynchrónnych upozornení aktívne požiadať. Na odoslanie požiadavky slúži operácia **<create-subscription>**. Správy sú následne asynchrónne zasielané klientovi až do doby, kým klient neukončí spojenie, prípadne nevyprší časový limit platnosti žiadosti určený parametrom *stopTime*. Parametre operácie:

**stream** - nepovinný parameter, špecifikujúci prúd udalostí o ktorých zasielanie má klient záujem.

**filter** - nepovinný parameter, ktorý slúži na určenie filtra, ktorý sa aplikuje na udalosti zasielané v rámci vybraného prúdu. Definícia filtra je zhodná s definíciou filtra používaného na filtráciu konfiguračných dát používaného základnými operáciami protokolu NETCONF.

**startTime** - slúži na špecifikáciu požiadavky na zasielanie upozornení o udalostiach, ktoré sa v systéme vyskytli ešte pred zaslaním požiadavky na ich odoberanie (funkcionalita replay). Hodnota parametra je typu *dateTime* a musí označovať časový moment v minulosti. V prípade, že udalosti uložené v logu nesiahajú do doby určenej hodnotou parametra *startTime*, server začne opätovne zasielať iba správy o udalostiach od najstaršieho záznamu v logu. Implementácia musí podporovať zadávanie času vrátane upresnenia časovej zóny.

**stopTime** - nepovinný parameter typu *dateTime* špecifikujúci okamžik, do ktorého má klient záujem dostávať správy o udalostiach (môže byť aj v budúcnosti). Ak nie je hodnota parametru definovaná, budú udalosti zasielané až do zrušenia požiadavky o ich zasielanie, napríklad ukončením NETCONF spojenia. Parameter môže byť použitý iba v kombinácií s parametrom *startTime*. Implementácia musí podporovať zadávanie času vrátane upresnenia časovej zóny.

Server začne po obdržaní požiadavku od klienta postupne odosielať upozornenia o udalostiach, ktoré v systéme nastali. Upozornenia sú zasielané asynchrónne ihneď po spracovaní informácií o udalosti do podoby XML dokumentu. Na samotné odoslanie upozornenia slúži operácia **<notification>**, ktorá pošle informácie o udalosti klientovi, ktorý inicioval jej odoberanie pomocou volania operácie *<create-subscription>*. Operácia *<notification>* nie je volaním vzdialenej procedúry (RPC), ale iba správou na vyššej vrstve protokolu a preto od klienta nevyžaduje odpoveď. Obsahom správy je plnohodnotný dokument v jazyku XML. Parametre operácie:

**eventTime** - Čas výskytu udalosti. Parameter je typu *dateTime*. Implementácia musí podporovať časové zóny.

Registráciu požiadavku na odoberania notifikačných správ je možné zrušiť volaním operácie *<close-session>* v kontexte spojenia ktoré požiadavok vytvorilo, alebo vynúteným ukončením *NETCONF* spojenia volaním operácie *<kill-session>*. Ak bola pri registrácií požiadavky na odber oznámení špecifikovaná hodnota atribútu *<stopTime>* bude táto požiadavka zrušená automaticky po uplynutí špecifikovaného času. Samotné NETCONF spojenie však, aj po uplynutí tejto doby, ostane aktívne.

## 4.3 Prúdy udalostí

Upozornenia o udalostiach, ktoré nastávajú vrámci jedného komponentu systému, alebo sú spolu inak logicky previazané, je možné zoskupovať do prudov udalostí. To umožňuje klientovi odoslať požiadavok na odoberanie iba udalostí z prúdov udalostí, ktoré sú pre neho zaujímavé. Dostupné prúdy udalostí môžu byť preddefinované výrobcom zariadenia - *pre-configured*, alebo používateľsky konfigurovateľné - *user configurable*. Implementácia môže taktiež podporovať definíciu obidvoch druhov súčasne. Definícia prúdov udalostí je súčasťou konfigurácie zariadenia. Výrobca zariadenia teda môže umožniť nastavovanie vybraných parametrov prúdov udalostí pomocou štandardných operácií protokolu, určených na prácu s konfiguráciou zariadenia.

Každá implementácia NETCONF agenta podporujúca rozšírenie :*notifications* musí, podľa špecifikácie, definovať minimálne východzí prúd udalostí s názvom "NETCONF". Obsah prúdu udalostí môže byť tvorený upozorneniami o prebiehajúcich operáciach protokolu.

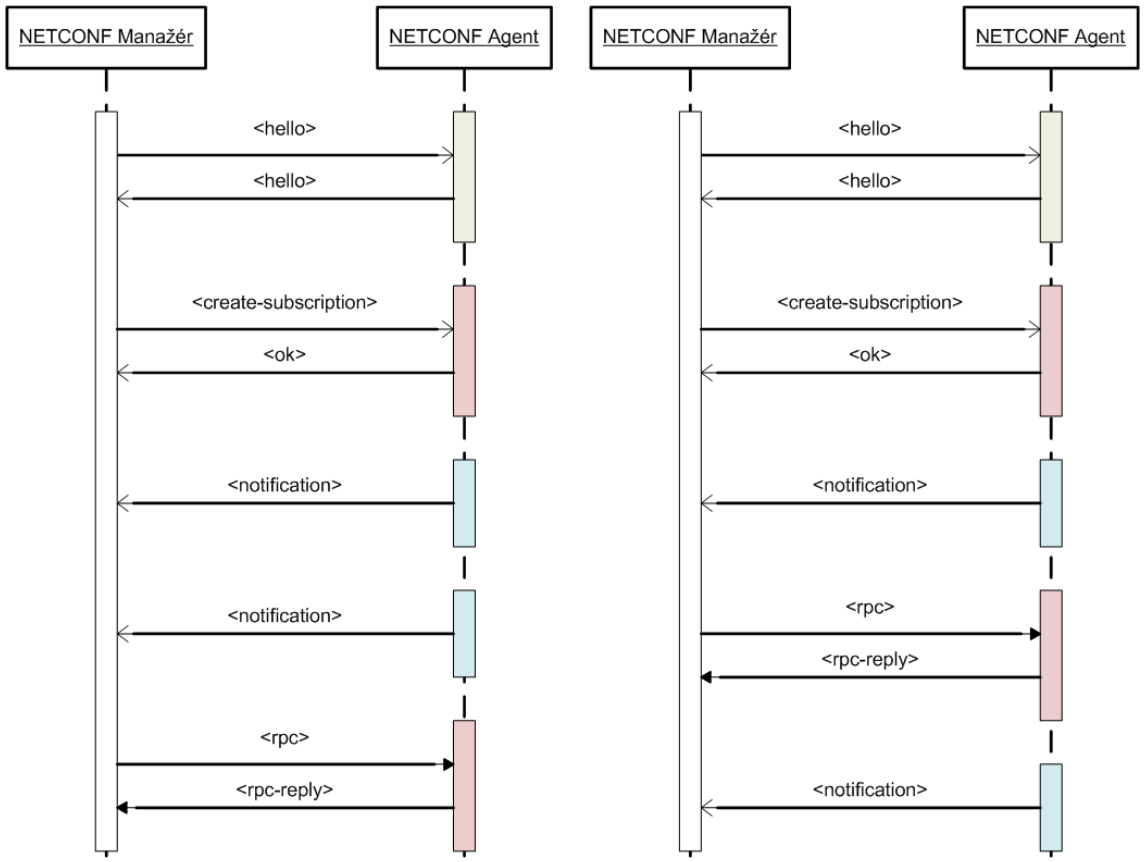
Zdrojom udalostí môžu byť rôzne komponenty systému zariadenia, napríklad hardvérové čítače, ale aj súčasti softvérového vybavenia ako napríklad syslog, SNMP a pod. Špecifikácia rozšírenia *:notifications* sa nezaoberá definíciou samotných zdrojov dát pre prúdy udalostí a necháva ju plne v réžii implementátora systému.

Na získanie zoznamu serverom podporovaných prúdov udalostí je možné použiť štandardnú operáciu <get> s parametrom *filter* nastaveným na získanie podstromu konfigurácie zariadenia obsahujúcu element <streams>. Zoznam dostupných prúdov obsahuje názov a popis jednotlivých prúdov vrátane rozširujúcich informácií o možnosti využitia služby opakovaného zasielania udalostí pre daný prúd, prípadne časovú značku najstaršej archivovanej udalosti, ktorú je služba schopná odoslať. Jedinou povinnou súčasťou popisu je názov prúdu - element <name>. Príklad záznamu komunikácie objavujúcej dostupné prúdy udalostí je uvedený v prílohe A.

## 4.4 Mechanizmus prelínania

Mechanizmus prelínania je definovaný ako rozšírenie *:interleave*[1] a umožňuje používať jedno spojenie protokolu NETCONF na súčasné zasielanie synchrónnych aj asynchrónnych správ.

Priebeh komunikácie medzi klientom a serverom bez použitia a následne s použitím rozšírenia *:interleave* je znázornený na obrázku 4.2. Bez využitia rozšírenia *:interleave* nie je možné spojenie, na ktorom je aktívne zasielanie upozornení, využiť na bežné operácie protokolu. Naopak s využitím rozšírenia je možné spojenie využívať na bežné synchrónne operácie aj napriek tomu, že je na ňom súčasne aktivované aj odoberanie upozornení.




Obr. 4.2: Komunikácia bez využitia/s využitím rozšírenia *:interleave*

Toto voliteľné rozšírenie protokolu pomáha zlepšiť škálovateľnosť celého riešenia, pretože umožňuje zníženie celkového počtu simultánnych NETCONF spojení, ktoré by inak boli potrebné na uspokojenie potrieb daných nárokmi operátora, prípadne aplikácie určenej na správu siete.

Rozšírenie nepridáva žiadne nové operácie, pridáva však jednu chybovú hlášku a to v prípade, že sa klient pokúsi vykonať operáciu <create-subscription> na spojení, ktoré má už zasielanie notifikačných správ aktivované.

[1] urn:ietf:params:netconf:capability:interleave:1.0

# Kapitola 5

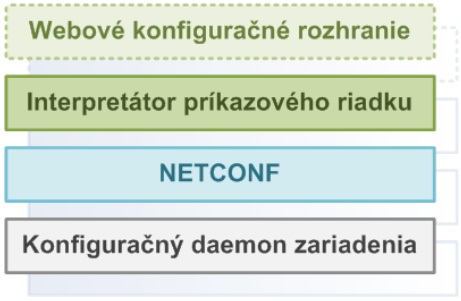
# Platforma Netopeer

Konfiguračný systém *Netopeer* je vyvíjaný ako otvorený a bezpečný aplikačný rámec určený na konfiguráciu a manažment sieťových zariadení založený na protokole NETCONF. Hlavným cieľom pri návrhu systému bolo umožnenie jeho nasadenia v kombinácií s rôznymi typmi zariadení. V ideálnom prípade je dátový model konfigurácie zariadenia, vyjadrený v niektorom jazyku určenom na modelovanie dát, jedinou súčasťou systému závislou na konkrétnom zariadení[1]. V rámci návrhu platformy bol vytvorený obecný dátový modely pre NetFlow sondu. Navrhnutý model bol následne využitý v projekte Liberouter na konfiguráciu NetFlow sondy *FlowMon*.

## 5.1 Architektúra

Architektúru platformy *Netopeer* možno vertikálne rozdeliť do troch základných vrstiev, v prípade napojenia na webové konfiguračné rozhranie možno rozlíšiť vrstvy štyri.

Usporiadanie jednotlivých vrstiev ilustruje obrázok 5.1. Konfiguračný démon sa stará o nastavovanie parametrov zariadenia podľa konfigurácie, ktorú načíta z úložiska, ktoré je spravované pomocou protokolu NETCONF. Na zadávanie príkazov môže byť použité jednoduché rozhranie v podobe príkazového riadku, alebo webové rozhranie. Webové rozhranie tvorí obálku nad rozhraním príkazového riadku. Umožňuje užívateľovi zostaviť jednotlivé príkazy pomocou intuitívneho grafického rozhrania.




Obr. 5.1: Obecná architektúra konfiguračnej platformy *Netopeer*

Najnižšia vrstva je tvorená konfiguračným démonom, ktorý je úzko naviazaný na konkrétny hardvér sieťového zariadenia a umožňuje nastavenie parametrov zariadenia podľa zadanej

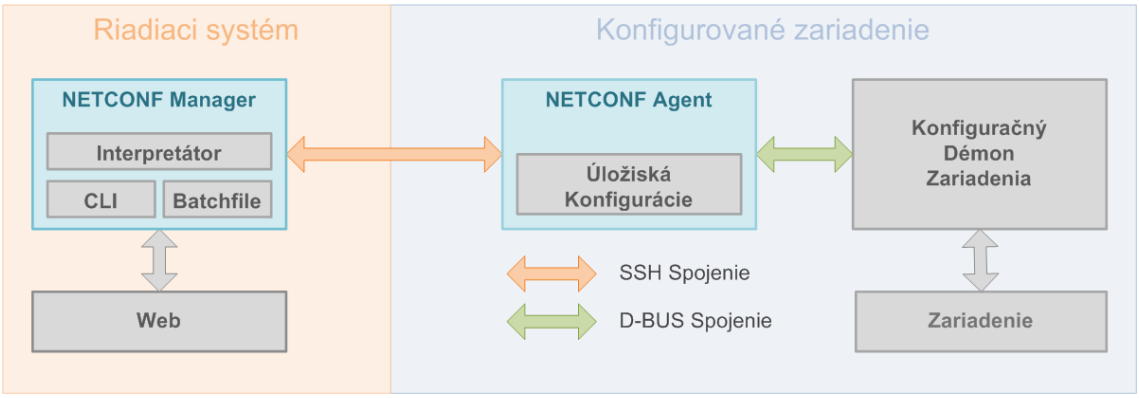
[1] Časť funkcionality konfiguračného démona, ktorá sa stará o nastavovanie parametrov zariadenia podľa zadanej konfigurácie a je závislá na konkrétnej hardvérovej implementácii zariadenia, nie je v rámci textu práce považovaná za súčasť konfiguračnej platformy.

konfigurácie, prípadne vyčítanie stavových informácií a ich uloženie do formátu XML. Podrobnejší popis architektúry konfiguračného démona je uvedený v kapitole 5.2.

Stredná vrstva reprezentuje zabezpečený komunikačný kanál medzi administrátorským rozhraním a zariadením. Stará sa o vykonávanie jednotlivých operácií protokolu, správu konfiguračných úložísk na strane zariadenia, nadväzovanie a rušenie spojenia a overovanie identity užívateľa.

Vrchnú vrstvu tvorí administrátorské rozhranie v podobe interpretátora príkazového riadku, ktorý slúži na zadávanie príkazov a na zobrazovanie doručených výsledkov operácií.

Najvyššia vrstva architektúry je tvorená webovým rozhraním, ktoré využíva služieb nižších vrstiev. Rozhranie slúži na zmenu konfiguračných dát a na vykonávanie príkazov bez nutnosti používať textové rozhranie interpretátora príkazového riadku.




Obr. 5.2: Komponenty konfiguračnej platformy *Netopeer*

Obrázok 5.2 ilustruje vzájomné prepojenie jednotlivých komponentov platformy. Webové rozhranie komunikuje s manažérom pomocou dávkových súborov typu *batchfile*, obsahujúcich jednotlivé príkazy. Manažér je prepojený s agentom pomocou protokolu NETCONF, v úlohe transportného protokolu je využitý protokol SSH. Agent komunikuje s konfiguračným démonom pomocou systému D-Bus.

**Manažér** - konzolová aplikácia, ktorá slúži na zadávanie príkazov. S okolím komunikuje v interaktívnom režime pomocou textovej konzoly a príkazového riadku, v neinteraktívnom režime dokáže spracovávať príkazy zadané vo forme textového *batchfile* súboru. Stará sa o nadviazanie SSH spojenia so spravovaným zariadením a o spustenie NETCONF agenta.

**Agent** - konzolová aplikácia, ktorá je v operačnom systéme zariadenia spúšťaná ako SSH subsystém. S NETCONF manažérom komunikuje pomocou protokolu NETCONF, na komunikáciu s konfiguračným démonom zariadenia využíva systém D-Bus.

## 5.2 Konfiguračný démon

Konfiguračný démon sa stará samotnú konfiguráciu zariadenia podľa parametrov nastavených v konfigurácií. Spôsob nastavovania parametrov je špecifický pre jednotlivé typy zariadení a preto návrh platformy *Netopeer* počíta s nutnosťou podporovať rôzne verzie konfiguračných démonov v závislosti na použitom zariadení. Platforma na tento účel definuje jednotné komunikačné rozhranie medzi agentom a konfiguračným démonom založené na komunikačnom systéme D-Bus. To umožňuje logické začlenenie konfiguračného démona do platformy vo forme zásuvného modulu.

Rozhranie medzi agentom a konfiguračným démonom je tvorené D-Bus rozhraním *org.liberouter.netopeer*. Konfiguračný démon vystupuje v D-Bus systéme ako služba *org.liberouter.netopeer*, ktorá implementuje sadu metód:

**commit** - metóda slúži na informovanie konfiguračného démona o obdržaní požiadavky na nastavenie parametrov zariadenia podľa aktuálneho obsahu konfiguračného úložiska *candidate*.

**copy** - metóda slúži na informovanie konfiguračného démona o zmene obsahu konfiguračného úložiska. V prípade, že bol zmenený obsah úložiska *running*, je démon povinný nastaviť parametre zariadenia podľa aktuálnej konfigurácie.

**delete** - upozorňuje konfiguračného démona o vykonaní požiadavku na zmazanie obsahu vybraného konfiguračného úložiska. Volanie metódy má informatívny charakter a väčšinou nevedie k zmene nastavenia parametrov zariadenia.

**edit** - upozorňuje konfiguračného démona na možné zmeny v obsahu konfiguračného úložiska. V prípade, že bol zmenený obsah úložiska *running*, je démon povinný nastaviť parametre zariadenia podľa aktuálnej konfigurácie.

**get** - metóda slúži na získanie informácií o aktuálnom stave zariadenia.

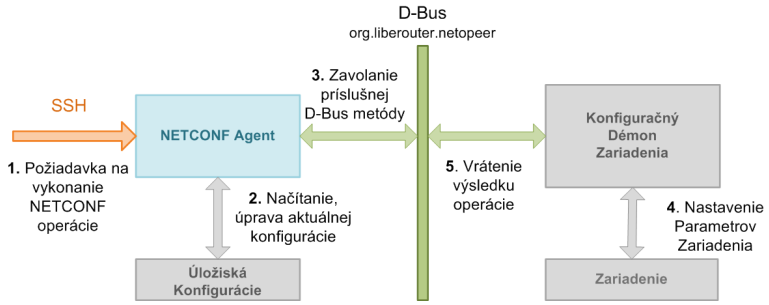
Deklarácia metód obsahuje dva vstupné a dva výstupné parametre:

**path** - vstupný parameter typu *string* obsahujúci identifikátor objektu reprezentujúceho konfiguračné úložisko v systéme D-Bus.

**inputData** - vstupný parameter typu *string* využívaný operáciou get, obsahujúci konfiguračné dáta, načítané agentom z úložiska *running*, doplnené o XML elementy reprezentujúce stavové informácie zariadenia. Konfiguračný démon následne vyplní príslušné hodnoty týchto elementov informáciami získanými priamo zo zariadenia.

**state** - výstupný parameter typu *boolean* používaný na predávanie výsledku operácie. Hodnota *true* indikuje, že operácia prebehla úspešne, naopak hodnota *false* je démonom vrátená v prípade chyby.

**outputData** - výstupný parameter typu *string* využívaný operáciou get na vrátenie konfiguračných dát doplnených o stavové informácie. V prípade výskytu chyby slúži parameter *outputData* na navrátenie textovej informácie o chybe.




Obr. 5.3: Priebeh spracovania požiadavku na vykonanie NETCONF operácie

Jednotlivé metódy, implementované konfiguračným démonom, sú volané agentom po obdržaní požiadavku na vykonanie príslušnej NETCONF operácie zo strany NETCONF manažéra. Agent pred zavolaním samotnej D-Bus metódy vykoná všetky požadované operácie

nad konfiguračnými dátami a v prípade potreby upraví obsah konfiguračných úložísk zariadenia. Výsledkom zavolania metódy je nastavenie parametrov zariadenia podľa príslušnej konfigurácie. Proces je znázornený na obrázku 5.3.

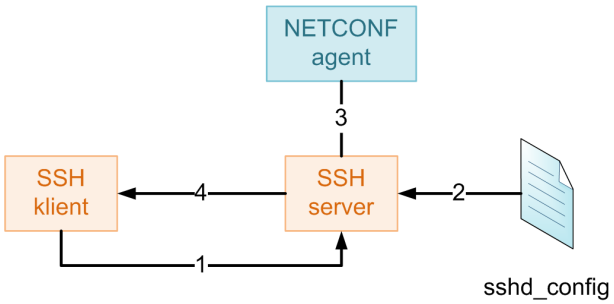
Konfiguračný démon by mal podporovať možnosť spustenia vo forme systémovej služby (démona) operačného systému Linux, čo zabezpečí jeho automatické spustenie po štarte operačného systému.

## 5.3 Využitie protokolu NETCONF

Najdôležitejšou súčasťou platformy *Netopeer* je vrstva zabezpečujúca komunikačné funkcie protokolu NETCONF. Vrstva je tvorená dvoma komponentami, ktoré medzi sebou komunikujú pomocou transportného protokolu SSH. Protokol SSH zabezpečuje utajený spojovaný prenos dát a služby overenia identity užívateľa. Platforma *Netopeer* využíva SSH vo verzii 2, ktorá prináša oproti predchádzajúcej verzii množstvo bezpečnostných vylepšení a najmä možnosť využitia nového konceptu spúšťania aplikácií na vzdialenom systéme vo forme SSH subsystému.

### Nadviazanie spojenia

V priebehu naväzovania NETCONF spojenia je najskôr vytvorené spojenie na úrovni transportného protokolu SSH, ktoré zabezpečuje vzájomnú autentikáciu komunikujúcich strán. Server zasiela klientovi otlačok svojho súkromného kľúča, ktorý jednoznačne určuje jeho identitu. Identita klienta môže byť overená pomocou zadaného užívateľského mena a hesla, prípadne s využitím identifikačného kľúča klienta. Následne je na strane servera spustený program implementujúci funkcionalitu agenta. Platforma *Netopeer* využíva na spustenie agenta koncept SSH subsystému, pretože pri tomto spôsobe spúšťania odpadá potreba poznať presnú cestu k programu agenta[2]. Celý proces nadviazania SSH spojenia ilustruje obrázok 5.4. Agent požiada SSH server o spustenie subsystému s názvom *netopeer*. Server načíta nastavenie zo súboru sshd_config, aby zistil, názov spúšťateľného súboru aplikácie. Následne aplikáciu spustí a presmeruje štandardný vstup a výstup programu do otvoreného komunikačného kanála.




Obr. 5.4: Spustenie programu agenta ako SSH subsystém

Po úspešnom ustanovení SSH spojenia prebieha fáza ustanovovania spojenia na úrovni protokolu NETCONF. Mechanizmus ustanovovania spojenia je bližšie popísaný v kapitole 3.5.

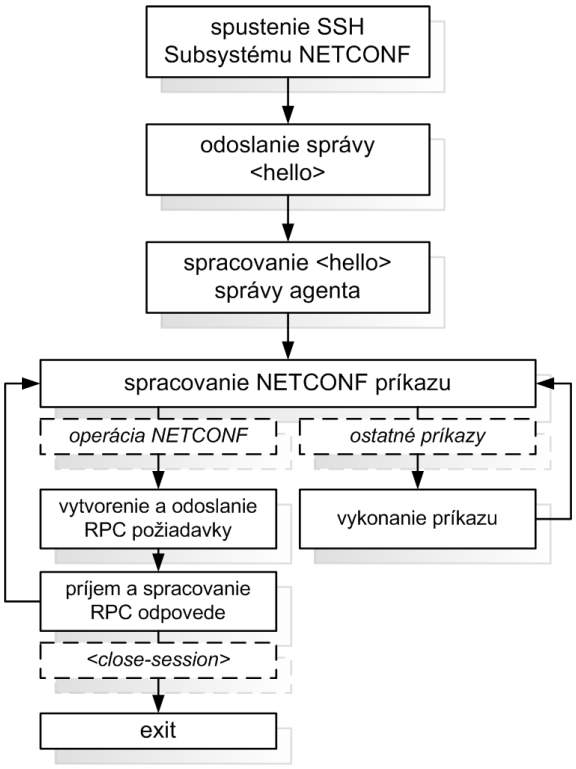
Komunikácia medzi manažérom a agentom je realizovaná postupným predávaním súborových prúdov. Samotné predávanie je realizované prostredníctvom dátových rúr (*pipes*). O vytvore-

[2]príslušná cesta je prednastavená v definícií subsystému v konfiguračnom súbore ssh_config

nie spojenia sa stará manažér. Po vytvorení spojenia má agent spustený ako SSH subsystém svoj štandardný vstup a výstup (*stdin*, *stdout*) prepojený pomocou SSH tunelu s manažérom. Po vytvorení spojenia má agent spustený ako SSH subsystém svoj štandardný vstup a výstup (*stdin*, *stdout*) prepojený so SSH klientom, ktorý ho ďalej predáva manažérovi (alebo naopak číta vstup pre subsystém). Týmto spôsobom je realizované plne duplexné spojenie medzi agentom a manažérom.

### Manažér

Program manažéra implementuje jednoduché užívateľské rozhranie umožňujúce zadávanie príkazov, ktoré vyvolávajú jednotlivé operácie definované v protokole NETCONF. Rozhranie umožňuje nadviazať a používať jedno spojenie v rámci jednej spustenej inštancie programu, t.j. umožňuje konfiguráciu iba jedného zariadenia. Možnosť konfigurácie viacerých zariadení súčasne môže poskytovať až nástavba v podobe pokročilého konfiguračného nástroja, ktorý využíva aplikáciu manažéra v úlohe podporného programu implementujúceho spojenie pomocou protokolu NETCONF.




Obr. 5.5: Schéma práce programu manažéra

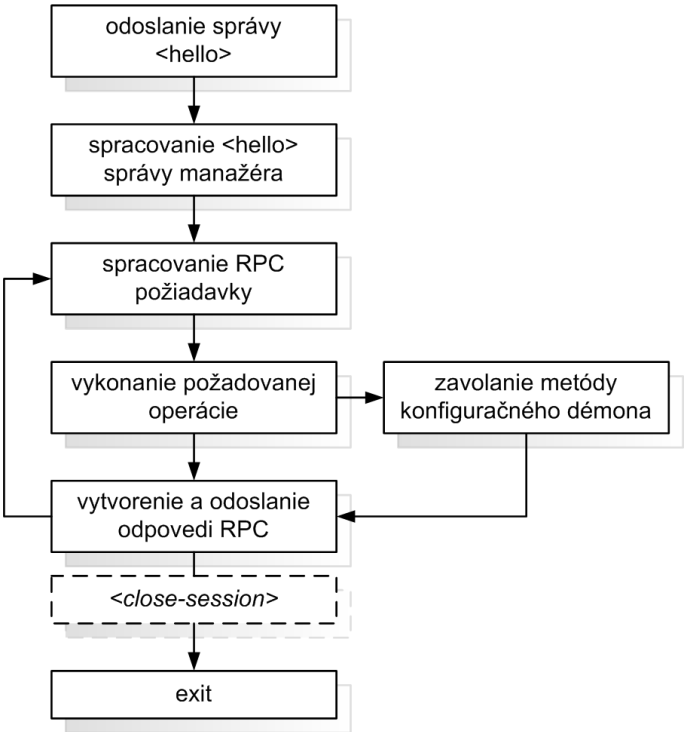
Obrázok 5.5 ilustruje riadiacu programovú slučku manažéra. Program sa ihneď po spustení pokúsi nadviazať spojenie pomocou protokolu NETCONF s agentom. Po úspešnom nadviazaní spojenia začne postupne v cykle vyčítať príkazy od užívateľa. Po úspešnom načítaní užívateľského príkazu vytvorí a odošle RPC správu požadujúcu vykonanie vybranej operácie. Program následne čaká na vykonanie operácie na strane agenta a na doručenie informácie o výsledku. Po zobrazení doručenej informácie užívateľovi program čaká na zadanie nového príkazu.

### Agent

Program agenta tvorí najkomplikovanejšiu súčasť platformy *Netopeer*, ktorá sa stará o vykonávanie jednotlivých operácií protokolu NETCONF, správu konfiguračných úložísk a komunikáciu s konfiguračným démonom zariadenia. Aplikácia agenta je automaticky spustená na konfigurovanom zariadení ako SSH subsystém, ihneď po úspešnom nadviazaní spojenia pomocou protokolu SSH.

Určité aspekty správania sa programu agenta je možné upravovať pomocou konfiguračného súboru */etc/liberouter/netopeer.conf*. Ten môže obsahovať nastavenie názvov súborov reprezentujúcich jednotlivé konfiguračné úložiská. Ďalej umožňuje špecifikovať názvy súborov, ktoré obsahujú šablóny v jazyku XSL a slúžia na označenie kľúčových elementov v dokumente konfigurácie, používané operáciou <edit-config> a šablóny používané pri doplňovaní konfigurácie uloženej v úložisku *running* o stavové elementy, ktorých hodnoty následne doplní konfiguračný démon, používané operáciou <get>. Konfiguračný súbor, ktorý nastavuje východzie hodnoty pre všetky konfigurovateľné parametre má nasledujúci obsah:

```
# running repository
running="/etc/netopeer/device_running.xml"
# startup repository
startup="/etc/netopeer/device_startup.xml"
# candidate repository
candidate="/etc/netopeer/device_candidate.xml"
# XSL stylesheet for adding key attribute in key elements
xsl_keys="/etc/netopeer/xsl/keys.xsl"
# XSL stylesheet for adding state elements in running configuration
xsl_stats="/etc/netopeer/xsl/stats.xsl"
```




Obr. 5.6: Schéma práce programu agenta

Riadiacu slučku programu agenta ilustruje obrázok 5.6. Po úspešnej výmene *<hello>* správ s manažérom začne agent čakať na požiadavky, ktoré po doručení postupne spracováva.

V prípade potreby nadviaže spojenie s konfiguračným démonom, ktorého informuje o práve vykonanej operácií (príkladom môže byť informácia o vykonaní operácie *<delete-config>*), prípadne po ňom požaduje zmenu nastavenia parametrov zariadenia, alebo doplnenie hodnôt stavových elementov konfigurácie. Po dokončení operácie zasiela agent manažérovi informáciu o výsledku.

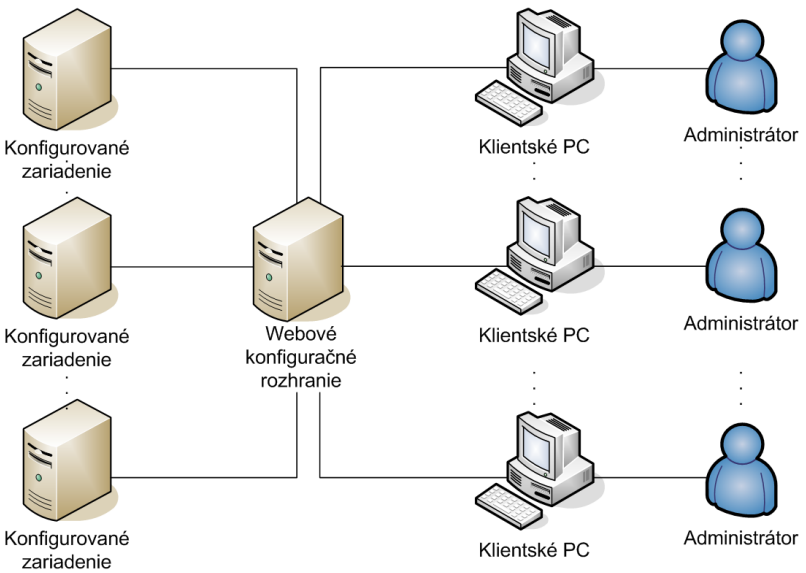
### Implementované rozšírenia protokolu

Implementácia protokolu okrem základných operácií podporuje aj niektoré štandardné rozšírenia. Jedným z nich je podpora konfiguračných úložísk *startup* a *candidate*. Úložisko *startup* obsahuje konfiguráciu, podľa ktorej je zariadenie nastavené ihneď po štarte/reštarte systému. *Candidate* slúži na ukladanie priebežných zmien v konfigurácií, ktoré je následne možné uložiť ako *running* pomocou rozširujúcej operácie <commit>. Operácia <discard-changes> slúži na zvrátenie zmien urobených v konfigurácií *candidate*, tak, že jej obsah prepíše obsahom aktuálnej konfigurácie *running*.

## 5.4 Webové rozhranie

Webové rozhranie tvorí najvyššiu vrstvu architektúry systému *Netopeer*. Slúži na editovanie konfiguračných dát uložených vo forme XML dokumentov. Návrh architektúry webového konfiguračného rozhrania platformy *Netopeer* počíta jednak s možnosťou inštalácie priamo v systéme konfigurovaného zariadenia, ako aj s možnosťou inštalácie na samostatný, na tento účel vyhradený, server. Výhodou inštalácie na samostatný server, je zvýšenie bezpečnosti celého riešenia (odpadá tak nutnosť inštalovať webový server na samotnom zariadení), zníženie nárokov na hardvérové prostriedky zariadenia ako aj možnosť používať jednu centrálnu inštaláciu webového rozhrania na konfiguráciu viacerých zariadení daného typu súčasne.

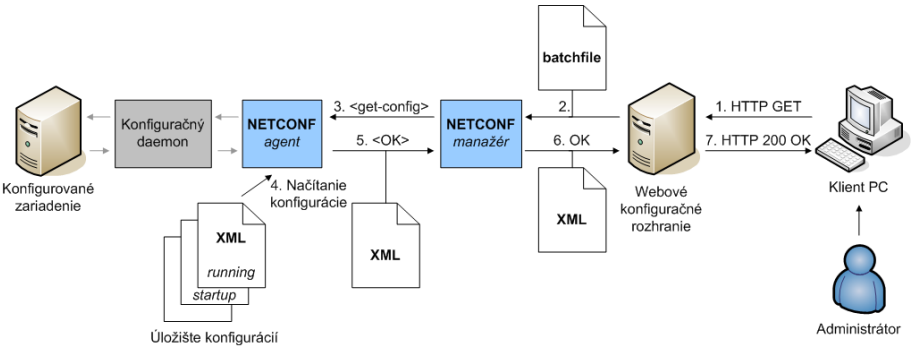
Možnosti využitia webového rozhrania inštalovaného na samostatnom servere ilustruje obrázok 5.7. Server umožňuje viacerým správcom siete využívať webové rozhranie súčasne. Rovnako umožňuje využiť jednu inštaláciu webového rozhrania na konfigurovanie viacerých zariadení.




Obr. 5.7: Architektúra webového konfiguračného rozhrania

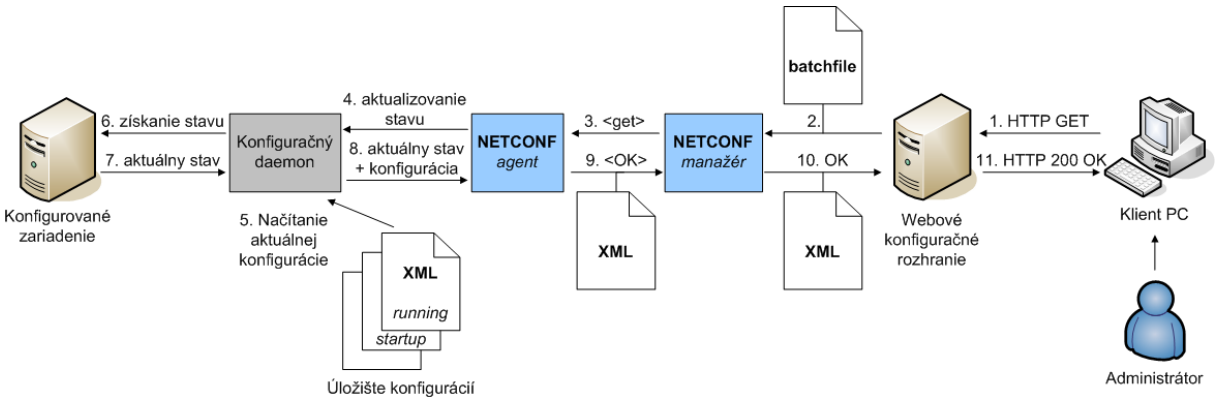
Samotné konfiguračné rozhranie je tvorené webovou aplikáciou, ktorá na komunikáciu s konfigurovaným zariadením využíva služieb nižších vrstiev platformy Netopeer, konkrétne vrstvu protokolu NETCONF. Na zadávanie príkazov rozhranie používa neinteraktívny režim manažéra, ktorý umožňuje zadávanie príkazov vo forme dávkových súborov *batchfile*.

### Využitie protokolu NETCONF




Obr. 5.8: Načítanie konfigurácie zariadenia pomocou protokolu NETCONF

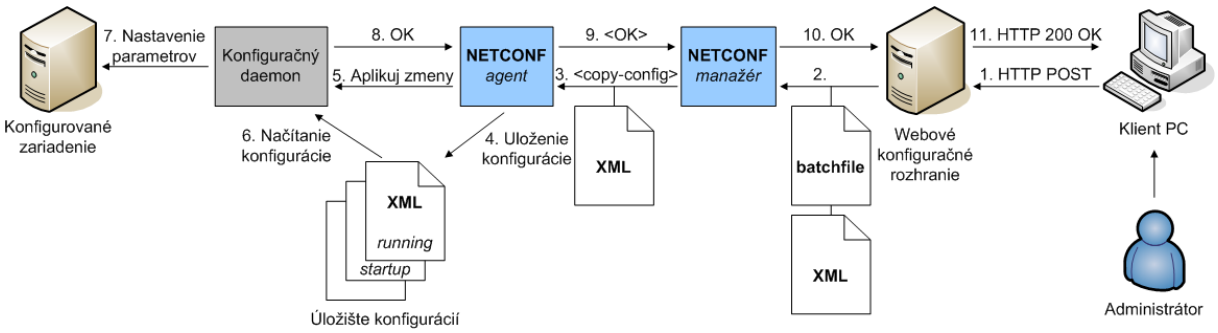
Obrázok 5.8 ilustruje postup pri načítaní konfigurácie. Správca siete odošle požiadavok na zobrazenie stránky webovému rozhraniu. To vygeneruje batchfile súbor, používaný na komunikáciu s manažérom, obsahujúci príkaz <get-config>. Následne je spustená nová inštancia manažéra, ktorá daný batchfile spracuje, t.j. nadviaže spojenie s agentom, požiada o vykonanie príkazu a výsledok v podobe dokumentu XML predá webovému rozhraniu zápisom na štandardný výstup, ktorý si webové rozhranie prečíta a spracuje. Po získaní potrebných dát je spojenie NETCONF ukončené a súbor batchfile je automaticky vymazaný. Následne webové rozhranie vygeneruje a odošle stránku s odpoveďou pre klienta. Pri operácií načítania konfigurácie nie je vyžadovaná priama interakcia agenta s konfiguračným démonom.




Obr. 5.9: Načítanie stavových informácií zariadenia pomocou protokolu NETCONF

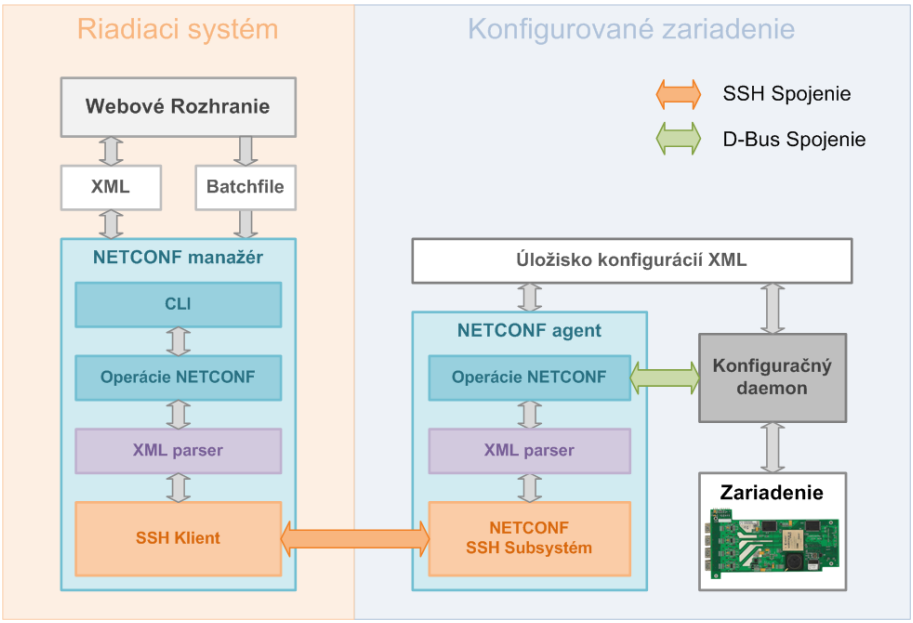
Postup pri načítaní stavových informácií zariadenia je podobný s postupom pri načítaní samostatnej konfigurácie. Jediným rozdielom je použitie operácie <get> namiesto operácie <get-config>. Pri tejto operácií však už je nutná interakcia agenta s konfiguračným démonom zariadenia. Agent požiada konfiguračného démona o vyčítanie stavových informácií priamo zo zariadenia a o doplnenie konfigurácie *running* o tieto informácie. Výsledný dokument je odoslaný manažérovi, ktorý ho ďalej predá webovému rozhraniu. Webové rozhranie

ho spracuje podobne ako v prípade načítania konfigurácie. Spojenie NETCONF je aj v tomto prípade ihneď po vykonaní operácie ukončené. Postupnosť jednotlivých krokov ilustruje obrázok 5.9.




Obr. 5.10: Nastavenie konfigurácie zariadenia pomocou protokolu NETCONF

Obrázok 5.10 znázorňuje postup pri konfigurácií zariadenia prostredníctvom webového rozhrania. Užívateľ pomocou formulárov webového rozhrania postupne mení obsah konfiguračného XML dokumentu. V momente keď je s vykonanými nastaveniami spokojný zadá pokyn na vykonanie operácie *commit*, ktorá iniciuje uloženie konfigurácie na vybrané zariadenie ako aj prípadnú zmenu parametrov daného zariadenia. V prvom kroku je zmenená konfigurácia uložená do dočasného súboru. Následne je vygenerovaný batchfile s príkazom <copy-config> a je spustená nová inštancia manažéra. Manažér nadviaže spojenie s agentom a požiada o vykonanie operácie <copy-config>. Ako parameter operácie sú agentovi odoslané príslušné konfiguračné dáta, uložené v dočasnom konfiguračnom súbore, pred pripravenom webovým rozhraním. Agent prepíše obsah vybraného úložiska získanou konfiguráciou a v prípade, že sa jedná o úložisko *running* kontaktuje konfiguračného démona s požiadavkou na zmenu nastavenia parametrov zariadenia. O úspešnom vykonaní operácie je manažér informovaný agentom pomocou RPC správy <ok>. Spojenie NETCONF je následne ukončené a súbor s dočasne uloženou konfiguráciou ako aj súbor batchfile sú automaticky vymazané.

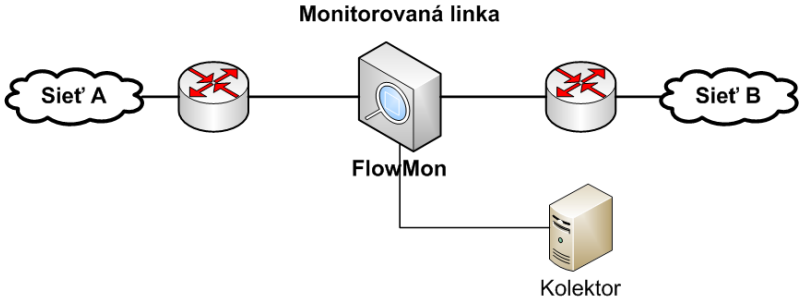



Obr. 5.11: Architektúra systému *Netopeer* používaná v rámci projektu Liberouter

Celkový pohľad na architektúru platformy *Netopeer* v úlohe hlavnej konfiguračnej platformy používanej na vzdialenú konfiguráciu sieťových zariadení vyvíjaných ako súčasť aktivity projektu Liberouter je znázornený na obrázku 5.11. Prepojenie medzi agentom a manažérom realizované protokolom SSH je vyznačené oranžovou šípkou. Prepojenie medzi agentom a konfiguračným démonom pomocou systému D-Bus je vyznačené zelenou šípkou.

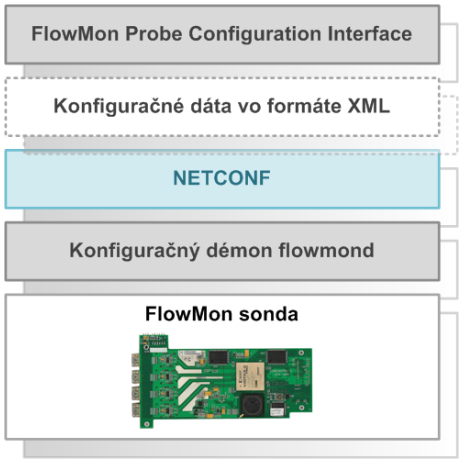
## 5.5 Konfiguračný systém sondy FlowMon

Pilotné nasadenie konfiguračnej platformy *Netopeer* prebehlo v rámci projektu pasívnej monitorovacej sondy *FlowMon*. Sonda slúži na zaznamenávanie štatistických informácií o prebiehajúcich tokoch v sieti. Tieto informácie sú následne pomocou exportérov odosielané na kolektor, kde je možné takto zozbierané dáta ďalej analyzovať. Sonda podporuje odosielanie dát vo formáte *NetFlow* verzie 5 a 9 a vo formáte *IPFIX*. Typické nasadenie sondy FlowMon na monitorovanie sieťovej linky ilustruje obrázok 5.12.




Obr. 5.12: Nasadenie sondy *FlowMon* na monitoring linky medzi sieťami A-B

Konfiguračný systém sondy FlowMon podporuje dva spôsoby konfigurácie. Prvý spôsob slúži na lokálnu konfiguráciu sondy pomocou štandardného rozhrania príkazového riadku systému Linux. Umožňuje užívateľovi konfiguráciu pomocou priamej editácie konfiguračných súborov (*flowmon.conf*), spúšťania shellových skriptov a konfiguračného démona *flowmond*. Tento spôsob je možné využiť aj na vzdialenú konfiguráciu po sieti, pripojením sa na zariadenie pomocou protokolu SSH. Druhou možnosťou je využiť možnosti konfiguračnej platformy *Netopeer* a na konfiguráciu použiť protokol NETCONF, konfiguračné súbory v jazyku XML a intuitívne webové rozhranie.

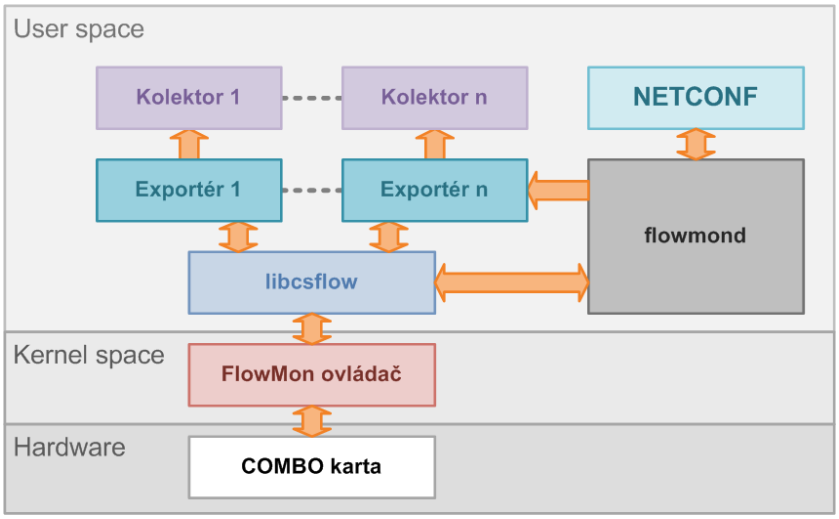



Obr. 5.13: Architektúra konfiguračného systému sondy *FlowMon*

**Architektúra konfiguračného systému** sondy *FlowMon* je definovaná architektúrou samotnej konfiguračnej platformy *Netopeer*. Najvyššia vrstva je tvorená webovým rozhraním *FlowMon Probe Configuration Interface*, stredná protokolom NETCONF a spodná konfiguračným démonom *flowmond*.

Vertikálne usporiadanie jednotlivých vrstiev architektúry ilustruje obrázok 5.13. Spodnú vrstvu tvorí konfiguračný démon *flowmond*, ktorý slúži na nastavenie parametrov sondy podľa konfigurácie, na ktorej správu je využitý protokol NETCONF. Na zadávanie príkazov slúži webové rozhranie, ktoré na vykonanie príkazu používa služby manažéra.

**Konfiguračný démon *flowmond*** implementuje najnižšiu vrstvu platformy *Netopeer*. Pomocou služieb knižnice *libcsflow* a ovládača priamo komunikuje s firmvérom sondy *FlowMon*, ktorý je nahratý v čipu FPGA[3] hardvérového akcelerátoru COMBO. Slúži na premietnutie obsahu konfiguračného súboru XML do nastavenia parametrov sondy. Je zodpovedný za spúšťanie a ukončovanie exportérov, ktoré slúžia na odosielanie záznamov o tokoch na kolektor. Konfiguračný démon komunikuje s agentom protokolu NETCONF pomocou rozhrania D-Bus.




Obr. 5.14: Architektúra monitorovacej sondy *FlowMon*

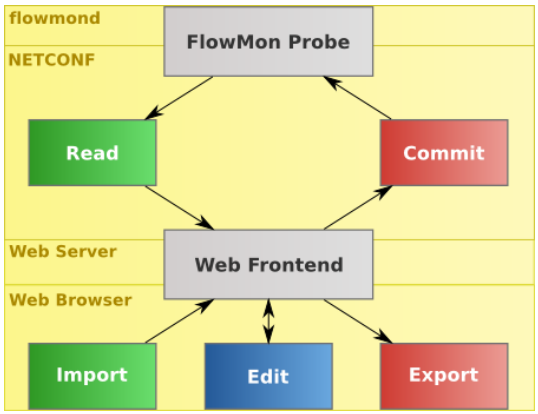
Architektúra systému na strane zariadenia je zobrazená na obrázku 5.14. Sonda používa hardvérovú platformu COMBO6X vyvinutá v rámci projektu Liberouter. S hardvérom karty komunikuje ovládač, ktorý beží ako súčasť jadra operačného systému. Služby ovládača sú pomocou knižnice *libcsflow* sprístupnené jednotlivým aplikáciám. Konfiguračný démon *flowmond* používa knižnicu *libcsflow* na nastavovanie parametrov sondy.

### 5.5.1 Webové rozhranie sondy FlowMon

Webové rozhranie *FlowMon Probe Configuration Interface* implementuje funkciu špecializovaného editora konfiguračných XML súborov sondy a implementuje najvyššiu vrstvu konfiguračnej platformy *Netopeer*. Editovaná konfigurácia môže byť pomocou protokolu NETCONF získaná priamo z konfiguračného úložiska na sonde, prípadne importovaná zo súboru pomocou webového prehliadača. Po následnej editácií obsahu prostredníctvom webových formulárov je možné výslednú konfiguráciu nahrať do vybraného úložiska na sonde, prípadne exportovať do súboru XML. Nahranie konfigurácie do úložiska *running* spôsobí okamžitú zmenu nastavenia parametrov sondy, prípadne spustených exportérov. Webové rozhranie je

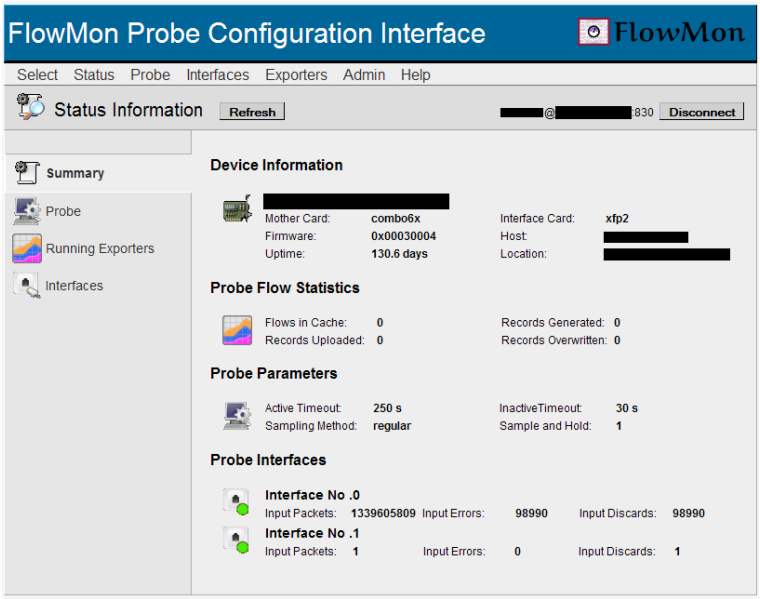
[3]Field-Programmable Gate Array

možné použiť aj na prehľadné zobrazenie stavových informácii získaných zo sondy. Vďaka podpore rozšírenia *:powecontrol* umožňuje webové rozhranie reštartovanie, prípadne vypnutie systému sondy FlowMon. Celý proces konfigurácie sondy *FlowMon* pomocou webového rozhrania ilustruje obrázok 5.15.




Obr. 5.15: Schéma procesu konfigurácie sondy *FlowMon*

Prvá verzia rozhrania bola vyvinutá ako súčasť bakalárskej práce autora. V rámci tejto diplomovej práce bola vyvinutá nová verzia rozhrania, ktorá má oproti pôvodnej značne rozšírenú funkcionalitu, a v súčasnosti podporuje nastavovanie väčšiny konfigurovateľných parametrov sondy. Funcionalita webového rozhrania bola rozšírená o podporu validácie konfiguračných dát voči referenčnému dátového modelu konfigurácie prostredníctvom XML validátora, využívajúceho na popis dát jazyk *RelaxNG*. K výsledkom tejto diplomovej práce možno zaradiť aj definíciu samotného dátového modelu konfiguračných dát sondy *FlowMon*. Nové webové rozhranie bolo úspešne zaradené do softvérovej výbavy dodávanej so sondou FlowMon a je aktívne využívané na jej konfiguráciu.




Obr. 5.16: Prostredie rozhrania *FlowMon Probe Configuration Interface*

# Kapitola 6

# Návrh platformy Netopeer 2.0

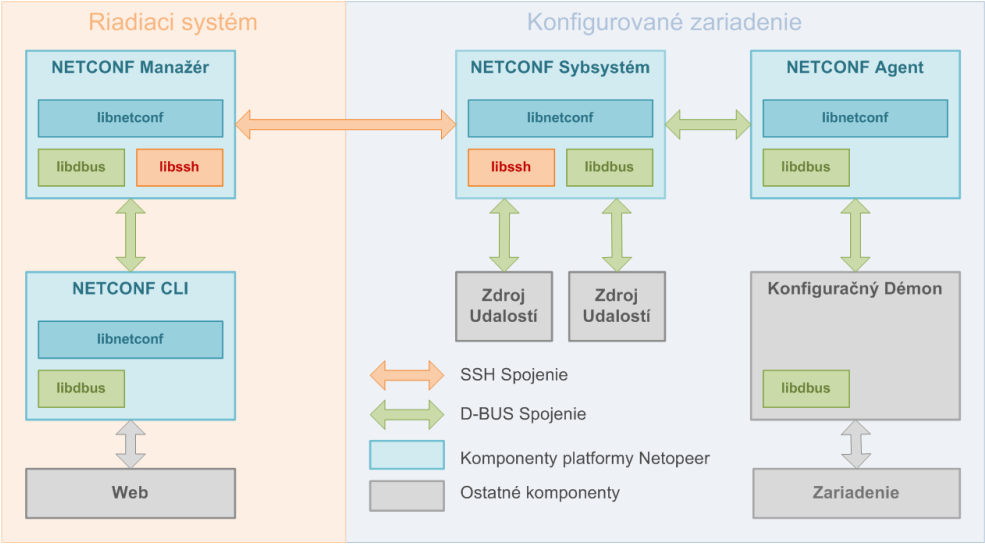
Konfiguračná platforma *Netopeer* vo verzii 1.0 neumožňuje využívať pokročilé možnosti asynchrónneho doručovania upozornení, definované ako súčasť špecifikácie rozšírenia *Event Notifications* protokolu NETCONF. Hlavným cieľom pri návrhu platformy *Netopeer* verzie 2.0 preto bolo definovať novú architektúru, ktorá by doručovanie asynchrónnych správ umožňovala. Návrh kladie dôraz na dodržanie požiadavku na zachovanie maximálnej možnej kompatibility s existujúcim riešením, najmä z pohľadu napojenia na systémovú službu konfiguračného démona zariadenia a napojenia na webové konfiguračné rozhranie. Zmeny sa preto dotkli najmä architektúry dvoch kľúčových komponent systému v podobe komponenty manažéra a komponenty agenta. Kapitola sa v úvode venuje popisu nového návrhu architektúry platformy ako celku a následne v podkapitolách postupne popisuje zmeny dizajnu jednotlivých komponent konfiguračného systému *Netopeer* 2.0.

## 6.1 Architektúra

Podpora konceptu asynchrónneho doručovania upozornení vyžaduje od manažéra schopnosť udržať nadviazané spojenie protokolu NETCONF otvorené počas celej doby, kým má užívateľ o ich doručovanie záujem. Existujúce riešenie manažéra v podobe jednoduchej konzolovej aplikácie by umožňovalo využívať túto funkcionalitu iba v kombinácií s použitím interaktívneho režimu ovládania. Webové rozhranie však na zadávanie príkazov protokolu NETCONF využíva neinteraktívny režim, ktorý umožňuje udržať spojenie otvorené iba počas doby vykonávania jednotlivých príkazov zadaných vo forme *batchfile* súboru. Problémom je aj bezstavový spôsob práce protokolu HTTP a samotného webového servera. To viedlo k potrebe rozdelenia funkcionality manažéra do dvoch samostatných komponent v podobe systémovej služby manažéra a samostatnej konzolovej aplikácie, ktorá implementuje funkcionalitu interpretátora príkazového riadku CLI (*Command Line Interface*), používaného na zadávanie príkazov. Systémová služba manažéra má za úlohu nadväzovanie a udržovanie otvoreného NETCONF spojenia a sprostredkovávanie volania jednotlivých operácií.

Zmeny sa dotkli aj komponenty agenta používanej na strane zariadenia. Motiváciou týchto zmien bolo navrhnúť architektúru v podobe, ktorá by v budúcnosti umožnila doplnenie funkcionality asynchrónneho doručovania správ o možnosti definované v špecifikácií rozšírenia *:replay*, viď popis operácie *<create-subscription>* v kapitole **??**. To viedlo k potrebe rozdelenia pôvodného komponentu agenta do dvoch samostatných komponent v podobe systémovej služby agenta a samostatnej konzolovej aplikácie subsystém, spúšťanej vo forme SSH subsystému, používaného na preposielanie požiadaviek na vykonanie operácií službe agenta a na preposielanie asynchrónnych upozornení o udalostiach v systéme službe

manažéra. Systémová služba agenta má na starosti samotné vykonávanie operácií protokolu NETCONF a komunikáciu s konfiguračným démonom zariadenia. Návrh počíta s možnosťou následného doplnenia služby agenta o funkcionalitu ukladania asynchrónnych upozornení do dočasného úložiska, odkiaľ ich bude možné v prípade potreby načítať, čo je základným predpokladom na budúce rozšírenie platformy *Netopeer* 2.0 o podporu štandardného rozšírenia :*replay*, ktoré slúži na dočasné ukladanie asynchrónnych upozornení na strane agenta a možnosť ich oneskoreného doručovania manažérovi.




Obr. 6.1: Návrh architektúry systému *Netopeer 2.0*

Kompletná schéma navrhnutej architektúry systému, vrátane vzájomného prepojenia jednotlivých komponent je znázornená na obrázku 6.1. Návrh počíta s využitím dvoch komunikačných systémov na prepojenie jednotlivých komponent:

**lokálne spojenie** je realizované prostredníctvom systému D-Bus. Ten slúži na efektívne prepojenie komponent v rámci jedného systému. V obrázku sú jednotlivé spojenia medzi komponentami ilustrované pomocou šípok zvýraznených zelenou farbou.

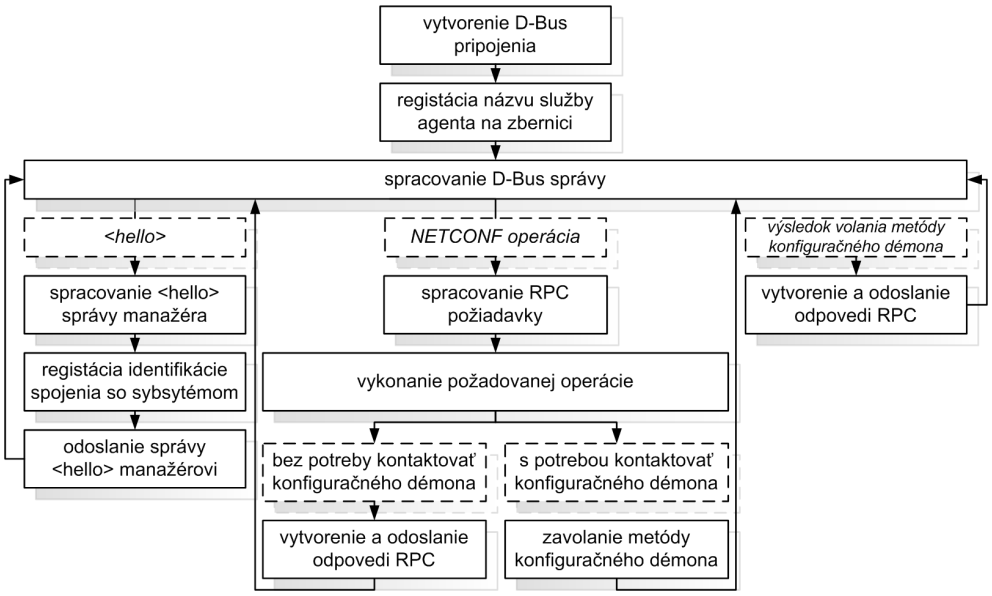
**vzdialené spojenie** medzi riadiacim systémom a konfigurovaným zariadením je realizované prostredníctvom protokolu SSH. Tým je vyhovené požiadavkám špecifikácie protokolu NETCONF. V obrázku je spojenie naznačené pomocou šípky zvýraznenej oranžovou farbou.

## 6.2 Konfiguračný démon zariadenia

Nový návrh architektúry zachováva osvedčený spôsob komunikácie NETCONF agenta s konfiguračným démonom zariadenia popísaný v kapitole 5.2. Vďaka tomu bolo možné zachovať kompatibilitu novej platformy s existujúcimi implementáciami konfiguračných démonov vyvinutých a používaných v projekte Liberouter. Zároveň tým bola zachovaná koncepcia využitia konfiguračného démona ako zásuvného modulu, ktorý slúži na vykonávanie operácií závislých na konkrétnom zariadení.

## 6.3 Agent

Ako súčasť návrhu novej architektúry platformy *Netopeer* 2.0 bolo potrebné kompletne prepracovať spôsob spracovania NETCONF požiadavkov komponentom agenta. Komponent na komunikáciu s okolím, v podobe komponentu subsystému a konfiguračného démona zariadenia, používa výhradne komunikačný systém D-Bus. Komponent musí umožňovať spustenie vo forme systémovej služby (démona) operačného systému.




Obr. 6.2: Schéma práce programu NETCONF Agent

Hlavná programová riadiaca slučka komponenty je schematicky znázornená na obrázku 6.2. Kompletné spracovanie si vystačí s využitím jedného vlákna, ktoré je súčasne hlavným vláknom samotného programu. Po štarte agent iniciuje pripojenie na systémovú zbernicu D-Bus a zaregistruje sa ako služba. Následne začne v cykle spracovávať príchodzie požiadavky odoslané NETCONF subsystémom, pomocou volania metódy *NetconfRequest*:

**hello** Správa je doručená pri ustanovovaní spojenia s agentom. Agent vytvorí a zaregistruje jedinečný identifikátor spojenia, ktorý následne slúži ako hodnota atribútu *<session-id>* na identifikáciu spojenia protokolu NETCONF. Agent vygeneruje správu *<hello>*, ktorú následne odošle manažérovi.

**operácia** Po doručení požiadavku na vykonanie operácie, je samotná operácia agentom vykonaná. Operácia môže vyžadovať na svoje dokončenie využitie služby poskytovanej konfiguračným démonom. V takom prípade agent vytvorí požiadavok na vykonanie príslušnej služby a počká na jeho dokončenie. Následne vygeneruje správu protokolu NETCONF obsahujúcu výsledok danej operácie a odošle ju manažérovi.

## 6.4 Manažér

Komponent manažér v rámci návrhu platformy *Netopeer* 2.0, podobne ako komponent agent, vystupuje ako služba operačného systému. Na komunikáciu s okolím využíva systém D-Bus a protokol SSH.

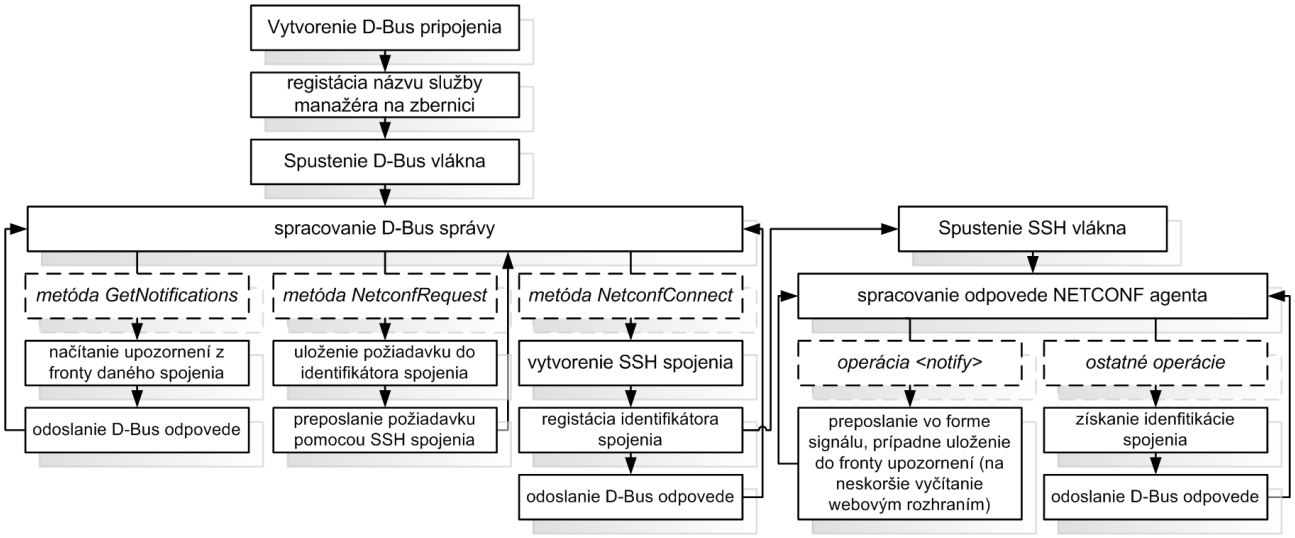
Hlavná programová riadiaca slučka komponenty je schematicky znázornená na obrázku 6.3. Komponent vyžaduje na kompletné spracovanie použitie viacerých programových vlákien.

Okrem hlavného programového vlákna, ktoré má na starosti spracovanie požiadavkov doručených pomocou systému D-Bus, vytvára ďalšie pomocné vlákna, ktoré slúžia na spracovanie výsledkov doručených pomocou protokolu SSH. Po spustení manažér iniciuje pripojenie na systémovú zbernicu D-Bus a zaregistruje sa ako služba. Následne začne v cykle spracovávať príchodzie požiadavky:

**NetconfConnect** metóda slúži na iniciovanie vytvorenia SSH spojenia. Po úspešnom nadviazaní spojenia manažér zaregistruje jedinečný identifikátor spojenia a vytvorí pomocné vlákno, ktoré má za úlohu spracovanie dát zaslaných agentom pomocou tohto SSH spojenia.

**NetconfRequest** metóda slúži na odoslanie požiadavku na vykonanie NETCONF operácie. Odoslanie požiadavku agentovi a spracovanie odpovede prebieha v rámci komponenty manažéra asynchrónne, čo umožňuje v budúcnosti rozšíriť funkcionalitu platformy *Netopeer* o podporu rozšírenia *:interleave*, bez nutnosti meniť existujúcu architektúru agenta.

**GetNotifications** metóda je používaná webovým rozhraním na vyčítanie aktuálneho obsahu fronty upozornení. Tým je zabezpečená možnosť využitia asynchrónneho doručovania upozornení a ich následného zobrazovania pomocou webového rozhrania.



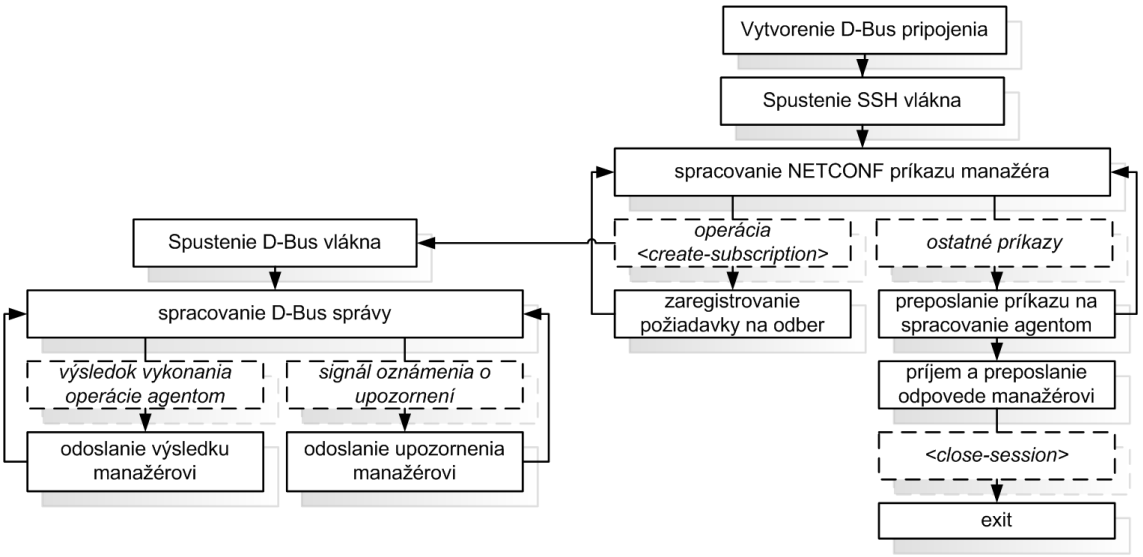

Obr. 6.3: Schéma práce programu Manažér

## 6.5 Subsystém

Komponent subsystém je v systéme konfigurovaného zariadenia spúšťaná vo forme SSH subsystému. S okolím komunikuje pomocou systému D-Bus a pomocou protokolu SSH. Každé otvorené NETCONF spojenie s konfigurovaným zariadením iniciuje spustenie novej inštancie subsystému.

Hlavná programová riadiaca slučka komponenty je schematicky znázornená na obrázku 6.4. Komponent vyžaduje na kompletné spracovanie použitie dvoch programových vlákien. Jedno z vlákien tvorí súčasne aj hlavné vlákno programu. Pomocné vlákno je vytvorené až po zaregistrovaní odberu asynchrónne doručovaných upozornení a slúži na spracovanie signálov, ktoré reprezentujú jednotlivé upozornenia.

Ihneď po štarte subsystému je iniciované pripojenie na systémovú zbernicu systému D-Bus, ktoré je používané na volanie metód služby komponenty agenta. Následne začne komponent v cykle spracovávať NETCONF požiadavky doručené pomocou protokolu SSH. Doručenie požiadavky na vykonanie operácie *<create-subscription>* iniciuje zaregistrovanie požiadavky na odber upozornení z príslušného zdroja a spustenie pomocného D-Bus vlákna, ktoré slúži na ich spracovanie. Ostatné operácie protokolu subsystém posiela ďalej na spracovanie agentovi pomocou synchrónneho volania D-Bus metódy *NetconfRequest*.




Obr. 6.4: Schéma práce programu Subsystém

## 6.6 Rozhranie príkazového riadku

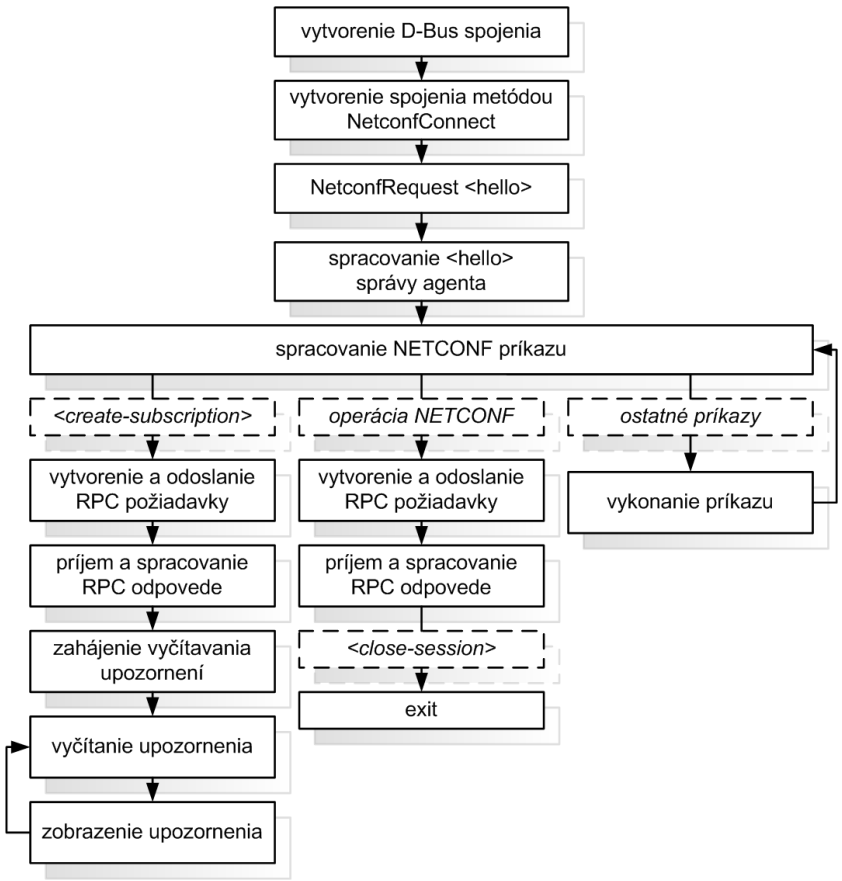
Komponent CLI implementuje funkcionalitu interpretátora príkazového riadku a slúži na zadávanie jednotlivých príkazov reprezentujúcich operácie protokolu NETCONF. Komponent zachováva plnú spätnú kompatibilitu s pôvodným riešením vyvinutým ako súčasť platformy *Netopeer* 1.0, realizovanom vo forme komponenty manažéra. Kompatibilita s pôvodným riešením je jednak na úrovni podporovaných argumentov príkazovej riadky systému, ako aj na úrovni jednotlivých príkazov a syntaxe jazyka samotného interpretátora.

Komponent vyžaduje na kompletné spracovanie použitie jedného vlákna, ktoré je súčasne aj hlavným vláknom programu. Po spustení iniciuje pripojenie na systémovú zbernicu systému D-Bus a následne volá metódu *NetconfConnect*, pomocou ktorej požiada komponent manažéra o nadviazanie SSH spojenia so systémom agenta. Po úspešnom nadviazaní SSH spojenia zahájí vytvorenie spojenia protokolu NETCONF, pomocou volania metódy *NetconfRequest*, ktoré realizuje operáciu *<hello>*. Následne synchrónne počká na doručenie výsledku, s obsahom tvoreným správou *<hello>*, ktorá reprezentuje odpoveď agenta. Po úspešnom vytvorení NETCONF spojenia sprístupní aplikácia jednoduché konzolové rozhranie (v prípade, že bola spustená v interaktívnom režime), ktoré umožňuje zadávať jednotlivé príkazy, reprezentujúce operácie protokolu. V prípade spustenia v neinteraktívnom režime načíta komponent zoznam príkazov zo zadaného *batchfile* súboru. Následné spracovanie príkazov prebieha v hlavnej slučke programu. Návrh platformy *Netopeer* 2.0 rozširuje zoznam podporovaných príkazov interpretátora o príkazy určené na prácu s asynchrónne doručovanými upozorneniami.

**create-subscription** príkaz slúži na registrovanie odberu asynchrónne doručovaných upozornení. Názov prúdu udalostí a názov súboru, ktorý obsahuje definíciu filtra je možné zadať ako parametere príkazu. Po úspešnom zaregistrovaní odberu upozornení prechádza riadenie programu v interaktívnom režime do cyklu v ktorom postupne vyčítavá jednotlivé upozornenia a zobrazuje ich na obrazovku. V neinteraktívnom režime ostáva riadenie v hlavnej slučke programu kde pokračuje spracovaním ďalších príkazov.

**get-notifications** príkaz slúži na získanie aktuálneho obsahu fronty upozornení udržiavanej komponentom manažéra. Vyčítané upozornenia sú z fronty automaticky zmazané. Pomocou príkazu *get-notifications* je možné využívať funkcionalitu asynchrónneho doručovania upozornení aj zo synchrónneho prostredia webového rozhrania.

Hlavná programová riadiaca slučka komponenty CLI je schematicky znázornená na obrázku 6.5.




Obr. 6.5: Schéma práce programu CLI

## 6.7 Webové rozhranie

Návrh architektúry platformy *Netopeer* 2.0 s pohľadu webového rozhrania zachováva plnú kompatibilitu s pôvodnou architektúrou platformy *Netopeer* 1.0. Cieľom bolo umožniť spoluprácu novej platformy s existujúcimi webovými rozhraniami vytvorenými pre použitie s platformou vo verzii 1.0. Návrh do budúcnosti počíta s možnosťou doplnenia platformy o nové komponenty, implementujúce funkcionalitu existujúceho komponentu CLI vo forme modulu,

prípadne iného typu rozšírenia použiteľného s niektorým z jazykov používaných na vývoj webových rozhraní. Tým by bolo v budúcnosti možné odstrániť potrebu generovania dočasných súborov s konfiguráciou a súborov typu *batchfile*. Modul, prípadne iný typ rozšírenia, by v budúcnosti mohol začať využívať služby komponentu manažéra priamo pomocou volania jednotlivých D-Bus metód implementovaných komponentom. Architektúra platformy *Netopeer* 2.0, navrhnutá ako súčasť tejto diplomovej práce, takúto možnosť prepojenia plne podporuje.

# Kapitola 7

# Implementácia platformy Netopeer 2.0

Programové vybavenie platformy *Netopeer* 2.0 vytvorené v rámci tejto diplomovej práce, vo forme referenčnej implementácie, prísne dodržuje požiadavky a špecifikáciu architektúry navrhnutej a popísanej v kapitole 6. Použité vývojové prostredie *NetBeans* a programovací jazyk C++ boli volené s ohľadom na budúce nasadenie platformy v projekte Liberouter, tak aby zodpovedali nástrojom, ktoré sú v projekte na vývoj bežne používané. Vytvorené programové vybavenie využíva služieb knižníc *libssh2*, *libxml2*, *libxslt*, *libdbus* a *libreadline*, ktoré sú bežne dostupné v prostredí operačného systému Linux.

## 7.1 Referenčná implementácia

V rámci referenčnej implementácie platformy *Netopeer* 2.0 bola vytvorená sada komponentov definovaných v architektonickom návrhu platformy. Všetky komponenty boli naprogramované v jazyku C++. Samotná implementácia striktne dodržuje architektonický návrh platformy *Netopeer* 2.0. Okrem komponent samotnej platformy boli vytvorené aj rôzne pomocné programy a nástroje použité na overenie funkcionality knižníc a komponent platformy.

Komplexnosť celého systému je možné vyjadriť pomocou sekvenčných diagramov priebehu vykonania synchrónnej a asynchrónnej operácie protokolu NETCONF. Diagram synchrónneho volania je znázornený na obrázku B.1, diagram asynchrónneho volania na obrázku B.2.

Správnosť implementácie vybraných funkčných celkov a častí zdrojového kódu platformy *Netopeer* 2.0 bola overená pomocou unit testov. Funkcionalita jednotlivých komponent vytvorených ako súčasť referenčnej implementácie bola overená pomocou vzájomných integračných testov jednotlivých komponent systému. Funkčnosť novej platformy ako celku a dodržanie spätnej kompatibility komunikačných rozhraní s pôvodnou implementáciou, v miestach, kde to bolo podľa architektonického návrhu vyžadované, bola overená pomocou regresných testov s existujúcim programovým vybavením platformy *Netopeer* 1.0.

### Unit testy

Na implementáciu unit testov bol využitý open source nástroj UniTest++[26]. Efektívnosť unit testov pri odhaľovaní rôznych druhov chýb v implementácií, sa ukázala byť značne vysoká. Unit testami bola pokrytá časť zdrojového kódu vybraných knižníc.

**libnetconf** - kompletne bola pokrytá funkcionalita spojená so XML serializáciou a deserializáciou jednotlivých typov NETCONF správ, vrátane správ protokolu RPC a triedy

*MessageFactory*.

**libutils** - kompletne bola pokrytá funkcionalita triedy *CmdArgParser*, implementujúcej efektívny nástroj na zjednodušenie spracovania argumentov príkazového riadku.

**liblogging** - testami bola pokrytá logika triedy *BaseLogger* vyhodnocujúca vykonanú akciu na základe typu správy.

### Integračné testy

Pomocou integračných testov bola overená správnosť implementácie vybraných knižníc, ktoré sú využívané programovým vybavením platformy *Netopeer*.

**libsshwrapper** - funkcionalita knižnice bola overená pomocou integračného testu testovacieho SSH klienta a SSH subsytému s nástrojmy Open SSH.

**libdbuswrapper** - funkcionalita knižnice bola overená pomocou implementácie fiktívneho konfiguračného démona a jeho následného integračného testu s komponentom agenta.

**libutils** - integračné testy testovacieho SSH klienta overili aj funkcionalitu implementovanú triedou *CmdArgParser* a implementáciu konverzných funkcií slúžiacich na získanie hodnôt vybraných datových typov z textového reťazca.

**liblogging** - pomocou integračného testu knižnice libdbuswrapper bola overená aj funkcionalita knižnice liblogging, ktorá je fiktívnym konfiguračným démonom ako aj agentom aktívne využívaná.

Rozšírenie funkcionality plaftormy *Netopeer* 2.0 o možnosť asynchrónneho doručovania upozornení bolo otestované s využitím konfiguračného démona, ktorý simuluje chovanie reálneho systému. Démon bol využitý na generovanie testovacích upozornení na vstupe do systému. Test overil funkčnosť spracovania asynchrónne doručovaných upozornení jednotlivými komponentami plaformy, počínajúc novým komponentom agenta, cez komponent subsystému, komponent manažéra až po komponent CLI na výstupe systému.

### Regresné testy

Funkčnosť nového programového vybavenia platformy bola overená aj pomocou regresných testov s existujúcim programovým vybavením konfiguračného systému sondy *FlowMon*, vyvinutého v rámci aktivít softvérovej skupiny projektu Liberouter.

Pomocou regresného testu s testovacím konfiguračným démonom platformy *Netopeer* 1.0 bolo overené zachovanie kompatibility komunikačného rozhrania medzi agentom a konfiguračným démonom na úrovni rozhrania systému D-Bus.

Zachovanie kompatibility komunikačného rozhrania medzi komponentom CLI a webovým rozhraním bola overená pomocou regresného testu novej implementácie platformy s webovým rozhraním *FlowMon Probe Configuration Interface,* pôvodne implementovaným na platforme *Netopeer* 1.0.

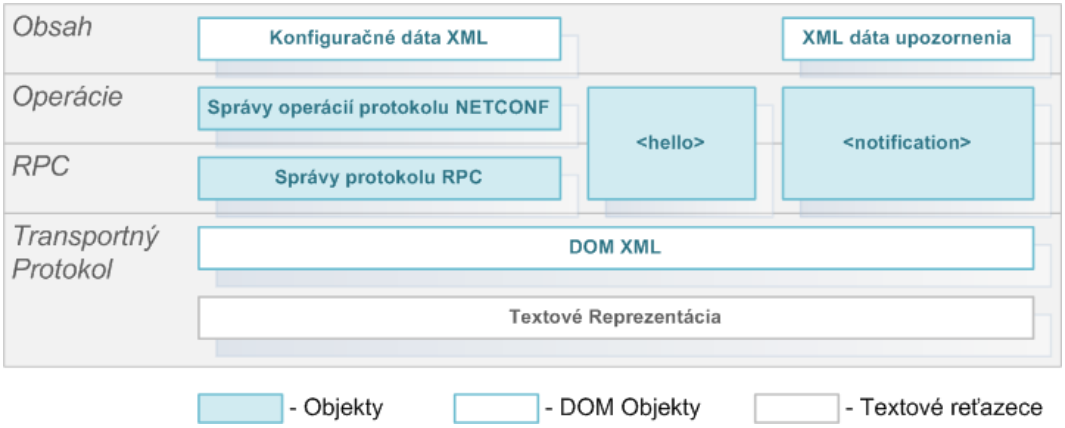
Správnosť implementácie samotného protokolu NETCONF bola overená pomocou regresného testu novej implementácie manažéra s existujúcou implementáciou agenta, vytvorenou ako súčasť bakalárskej práce Radka Krejčího[3].

## 7.2 Knižnice funkcií a tried

Ako súčasť tejto diplomovej práce bola vytvorená sada knižníc zapúzdrujúcich prácu so systémom D-Bus, protokolom SSH, správami protokolu NETCONF a rôzne pomocné knižnice. Tieto knižnice boli následne využité pri implementácii jednotlivých komponent konfiguračného systému platformy *Netopeer* 2.0.

### libnetconf

Knižnica *libnetconf* poskytuje sadu tried na prácu so správami protokolu NETCONF. Užívateľ knižnice, v podobe aplikácie, sa nemusí starať o správne generovanie a parsovanie textového obsahu jednotlivých správ protokolu, ale využíva abstraktnú dátovú reprezentáciu jednotlivých typov správ, poskytovaných knižnicou, v podobe objektov.




Obr. 7.1: Dátová reprezentácia správ na jednotlivých vrstvách protokolu NETCONF.

Dátová reprezentácia správ protokolu NETCONF používaná knižnicou *libnetconf* na jednotlivých vrstvách architektúry protokolu, je znázornená na obrázku 7.1. Diagram tried knižnice je znázornený na obrázku v prílohe B.3.

**vrstva obsahu** - na reprezentáciu konfiguračných dát zariadenia a dát upozornení používa knižnica XML DOM objekty štandardnej systémovej knižnice *libxml2*.

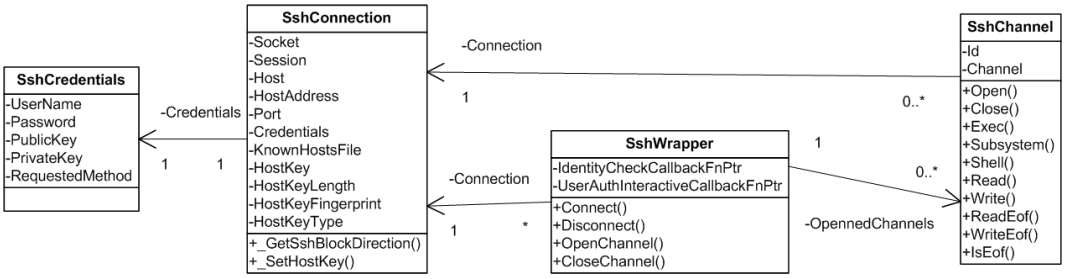
**vrstva operácií** - na reprezentáciu operácií protokolu využíva knižnica sadu tried reprezentujúcich jednotlivé operácie. Príkladom môže byť trieda *GetConfigMessage* reprezentujúca operáciu *<get-config>*. Triedy neslúžia na reprezentáciu samotných konfiguračných dát, ktoré sú súčasťou vrstvy obsahu protokolu.

**vrstva rpc** - správy RPC vrstvy sú reprezentované triedou *RpcMessage*. Trieda sa stará iba o spracovanie správ samotného protokolu RPC, nezaoberá sa spracovaním jednotlivých operácií protokolu NETCONF, tie trieda považuje za dáta operácií protokolu RPC.

**vrstva transportného protokolu** - knižnica na úrovni vrstvy transportného protokolu reprezentuje jednotlivé správy protokolu v textovej podobe, ktoré je možné priamo používať na odosielanie a prijímanie pomocou transportnej vrstvy. Samotnú realizáciu odosielania a prijímania, knižnica ponecháva v réžii užívateľskej aplikácie.

### libsshwrapper

Knižnica *libsshwrapper* zapúzdruje komunikáciu pomocou protokolu SSH. Tvorí obálku nad knižnicou *libssh2,* naprogramovanú v jazyku C++. Umožňuje využívať väčšinu možností protokolu SSH2, vrátane rôznych spôsobov autentikácie užívateľa a podporuje prácu so súborom *know_hosts,* využívaným systémom SSH na ukladanie informácií o identite vzdialeného systému. Diagram tried knižnice *libsshwrapper* je znázornený na obrázku 7.2.

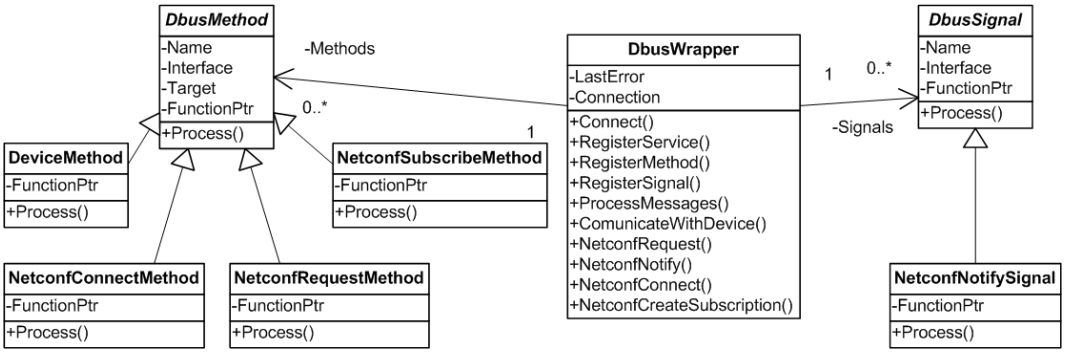



Obr. 7.2: Diagram tried knižnice libsshwrapper

Na vytvorenie spojenia so vzdialeným systémom slúži trieda *SshWrapper*. Knižnica podporuje vytvorenie viacerých inštancií triedy a umožňuje tak komunikáciu s viacerými zariadeniami súčasne. Jedno spojenie môže mať otvorených viac komunikačných kanálov, reprezentovaných triedou *SshChannel*, ktorá poskytuje operácie na zápis a čítanie dát z dátového prúdu kanála, spustenie príkazu, spustenie terminálu a spustenie subsystému na vzdialenom systéme. Knižnica podporuje synchrónny aj asynchrónny prístup k čítaniu a zápisu dát.

### libdbuswrapper

Knižnica *libdbuswrapper* zapúzdruje komunikáciu pomocou komunikačného systému D-Bus. Je implementovaná v podobe obálky nad knižnicou *libdbus*, naprogramovanej v jazyku C++. Knižnica *libdbus* slúži na nízkoúrovňovú prácu s komunikačným protokolom D-Bus a preto nie je vhodná na priame použitie pri vývoji aplikácií. Na tento účel slúžia rôzne implementácie obálok - *dbus binding* zapúzdrujúce prácu s knižnicou, ktoré sú dostupné pre rôzne programovacie jazyky. Dostupné riešenia pre jazyk C++ sú neúmerne zložité, v porovnaní s potrebami platformy *Netopeer* 2.0 a preto bola ako súčasť tejto diplomovej práce implementovaná jednoduchá obálka knižnice *libdbus*, ktorá viac vyhovuje potrebám projektu. Obálka je tvorená sadou tried, ktoré sprístupňujú prácu s komunikačným systémom D-Bus. Diagram tried knižnice *libdbuswrapper* je znázornený na obrázku 7.3.




Obr. 7.3: Diagram tried knižnice libdbuswrapper

Na vytvorenie spojenia so zbernicou systému D-Bus slúži trieda *DbusWrapper*. Knižnica umožňuje vytvorenie viacerých inštancií triedy *DbusWrapper* a tak umožňuje udržovať viac aktívnych pripojení na zbernicu súčasne. Trieda umožňuje registrovanie aplikácie v podobe služby na zbernici systému D-bus. Knižnica definuje niekoľko typov metód a jeden typ signálu. Aplikácia si môže pomocou triedy *DbusWrapper* registrovať vlastné funkcie obsahujúce zdrojový kód, ktorý bude volaný v prípade doručenia požiadavku na vykonanie daného typu metódy, prípadne doručenia signálu. Sadu metód a signálov podporovaných knižnicou je možné jednoducho rozširovať, v súčasnosti obsahuje iba metódy, ktoré boli potrebné na implementovanie komunikačného rozhrania medzi jednotlivými komponentmi platformy *Netopeer* 2.0.

# Kapitola 8

# Bezpečnosť platformy Netopeer 2.0

Celková bezpečnosť konfiguračnej platformy *Netopeer* je vymedzená využitím operačného systému Linux a použitými nástrojmi a knižnicami. Pri správnom nastavení je možné zachovať rozumnú mieru zabezpečenia bez nutnosti obmedzovať funkcionalitu a tým aj výslednú použiteľnosť platformy v praxi.

## Operačný systém

Zabezpečenie operačného systému Linux môže byť pri správnom nastavení na vysokej úrovni. Platforma *Netopeer* využíva štandardné užívateľské účty systému, čo umožňuje aplikovanie všetkých vstavaných bezpečnostných mechanizmov systému, vrátanie rozšírenia SE Linux[1].

**Manažér** - na zaručenie maximálnej možnej bezpečnosti je doporučené vytvoriť špeciálny užívateľský účet, pod ktorým bude bežať systémová služba manažéra. Následne je možné obmedziť prístupové práva k súborovému systému iba do adresárov, v ktorých sa nachádzajú systémové knižnice *libxml2*, *libdbus* a *libssh2*, využívané aplikáciou a do adresára */var/log*, kam služba zapisuje informácie o behu v podobe log súboru. To zaručí, že aj v prípade zneužitia chyby v programe, nebude môcť útočník kompromitovaný program použiť na zmenu nastavení systému, prípadne krádež citlivých dát.

**Agent** - podobne je doporučené vytvoriť špeciálny účet aj pre systémovú službu agenta a následne obmedziť prístupové práva k súborovému systému podobne ako v prípade služby manažéra. Agent navyše vyžaduje prístup na čítanie a zápis do adresára */etc/netopeer* v ktorom sa nachádzajú súbory reprezentujúce jednotlivé konfiguračné úložiská, XSL šablóny a konfiguračný súbor *netopeer.conf*. Tento adresár by nemal mať povolené práva na spúšťanie súborov. Na využitie možnosti vzdialeného reštartu a vypnutia zariadenia, je potrebné užívateľskému účtu pod ktorým beží agent povoliť spúšťanie príkazu */sbin/reboot*, prípadne */sbin/poweroff*.

**Subsystém** - SSH subsystém platformy *Netopeer* je spúšťaný pod užívateľským účtom, ktorý bol použitý na nadviazanie SSH spojenia. Doporučené nastavenie je vytvoriť na účely vzdialenej konfigurácie zariadenia osobitný užívateľský účet pre každého administrátora, ktorý potrebuje so zariadením pracovať.

**CLI** - kozolové rozhranie platformy Netopeer môže byť spustené pod identitou práve prihláseného užívateľa, alebo pod účtom, pod ktorým beží webový server. Užívateľský účet

[1] Security-Enhanced Linux

pod ktorým beží webový server musí mať nastavené práva na prístup k systémovej zbernici D-Bus, čo môže pri nesprávnej konfigurácií webového serveru predstavovať bezpečnostné riziko. Doporučené je nepoužívať anonymnú autentifikáciu.

**Konfiguračný Démon** - by mohol zdieľať spoločný užívateľský účet s agentom. Takéto nastavenie by však vyžadovalo povoliť mu v systéme práva na vytváranie zariadení pomocov systémovej služby *udev*. V prípade konfiguračného systému sondy *FlowMon* by bolo potrebné povoliť aj práva na zmenu sieťových nastavení systému, pretože webový konfiguračný systém sondy umožňuje aj zmenu nastavení administračného rozhrania sondy. Takéto nastavenie môže predstavovať bezpečnostné riziko a preto je doporučené vytvoriť pre systémovú službu konfiguračného démona vlastný užívateľský účet.

## Komunikačný systém D-Bus

Úroveň zabezpečenia komunikačného systému D-Bus používaného platformou *Netopeer* je možné konfigurovať pomocou nastavenia prístupových práv k zbernici. Nastavenia je možné špecifikovať pre jednotlivé užívateľské účty, alebo skupiny. Nastavenie je uložené v konfiguračnom súbore */etc/dbus-1/system.d/netopeer-dbus.conf*. Konfiguračný systém *Netopeer* vyžaduje povolenie nasledovných práv:

**Manažér** - užívateľský účet pod ktorým je spustený manažér musí mať povolené právo registrovať si na zbernici názov *org.liberouter.netopeer.manager*.

**Agent** - užívateľský účet pod ktorým je spustený agent musí mať povolené právo registrovať si na zbernici názov *org.liberouter.netopeer.agent* a právo odosielať a prijímať správy z adresy *org.liberouter.netopeer* ktorá je používaná službou konfiguračného démona.

**Subsystém** - užívateľský účet pod ktorým je spustený subsystém musí mať povolené odosielanie a prijímanie správ z adresy *org.liberouter.netopeer.agent* a v prípade využívania rozšírenia *:notifications* aj prijímanie správ zo všetkých adries registrovaných jednotlivými systémovými službami realizujúcimi zdroje udalostí. Doporučené nastavenie je vytvoriť užívateľskú skupinu, ktorá má nastavené príslušné práva a následne do nej zaradiť všetky užívateľské účty, ktoré sa budú na zariadenie pomocou subsystému pripájať.

**CLI** - užívateľský účet pod ktorým je spustené konzolové užívateľské rozhranie musí mať povolené odosielanie a prijímanie správ z adresy *org.liberouter.netopeer.manager*, ktorá je registrovaná službou manažéra. Doporučené nastavenie je vytvoriť užívateľskú skupinu, ktorá má toto právo nastavené a následne do nej zaradiť všetky užívateľské účty, ktoré budú konzolové rozhranie používať.

**Konfiguračný Démon** - Užívateľský účet pod ktorým je spustený konfiguračný démon musí mať povolené právo registrovať si na zbernici názov *org.liberouter.netopeer*.

## Komunikačný systém SSH

Protokol SSH je všeobecne považovaný za bezpečný, pretože zabezpečuje vzájomnu autentifikáciu komunikujúcich strán a rieši integritu a utajenie správ počas prenosu. Platforma *Netopeer* podporuje automatické overovanie identity servera pomocou súboru *known_hosts*. Na overenie identity klienta je možné použiť niektorý zo štandardne dostupných mechanizmov protokolu SSH:

- overenie pomocou užívateľského mena a hesla
- overenie pomocou užívateľského mena a identifikačného kľúča

Doporučené nastavenie je použitie overovania pomocou zadania užívateľského mena a hesla.

## Webový server apache

Správne nakonfigurovaný webový server *apache* možno považovať za dostatočne bezpečný vzhľadom na zameranie a typické nasadenie platformy *Netopeer*. Webové konfiguračné rozhranie platformy je doporučené inštalovať na dedikovaný webový server, ktorý okrem konfiguračného rozhrania nehostuje žiadne ďalšie webové aplikácie tretích strán. Webový server by mal byť nakonfigurovaný aby umožňoval pripojenie iba prostredníctvom protokolu HTTPS[2]. Zároveň je doporučené používať modul *mod_auth* a definovať osobitný užívateľský účet pre každého správcu, ktorý bude webové rozhranie platformy používať.

Užívateľský účet pod ktorým beží systémová služba webového serveru vyžaduje povolenie na čítanie a zápis do adresára v ktorom sú uložené dočasné XML súbory obsahujúce konfiguračné dáta a dávkové súbory *batchfile* obsahujúce príkazy protokolu NETCONF. Na využitie funkcionality importovania konfigurácie zo súboru, pomocou webového prehliadača je potrebné povoliť prístup na zápis do pomocného adresára, do ktorého sú tieto súbory webovým serverom dočasne ukladané. Je výslovne doporučené využívať na tieto účely samostatný adresár. V žiadnom prípade by na tento účel nemal slúžiť adresár, do ktorého webové rozhranie ukladá dočasné súbory s konfiguráciou a súbory typu *batchfile*.

Webové rozhranie konfiguračného systému sondy *FlowMon* je naprogramované v jazyku PHP verzie 5, ktorý možno považovať za bezpečný, vzhľadom na nasadenie systému. Pri implementácií bol kladený dôraz na dodržanie základných zásad bezpečnej tvorby zdrojového kódu v tomto jazyku ako napríklad nepoužívanie direktívy *register_globals*, kontrólovanie všetkých užívateľských vstupov, vrátane obsahu kolekcií *$_GET*, *$_POST* a *$_REQUEST*, vymazávanie obsahu *session* po odhlásení užívateľa a podobne.

Hlavné doporučenie na nastavenie samotného prostredia interpretátora jazyka PHP, s pohľadu bezpečnosti, sa týka nastavenia direktívy *display_errors* na hodnotu „OFF", nastavenia direktívy *session.use_only_cookies* na hodnotu „1" a povolenia iba tých rozširujúcich modulov, ktoré sú nevyhnutné na beh webového konfiguračného rozhrania[3]. Samotný interpretátor jazyka PHP by mal byť nakonfigurovaný vo forme modulu *mod_php* webového serveru apache, spúšťanie interpretátora pomocou rozhrania CGI[4] nie je doporučené.

[2]Hypertext Transfer Protocol Secure, podporovaný serverom *apache* po nainštalovaní modulu *mod_ssl*

[3]Niektoré distribúcie Linuxu vyžadujú dodatočné povolenie modulu na prácu s XML pomocou direktívy -with-xml

[4]Common Gateway Interface

# Kapitola 9

# Záver

V rámci predloženej diplomovej práce bola naštudovaná problematika konfigurácie sieťových zariadení. Boli analyzované výhody i nevýhody rôznych v súčasnosti rozšírených spôsobov konfigurácie sieťových zariadení. Hlavný dôraz bol kladený na analýzu možností vzdialenej konfigurácie sieťových zariadení. Podrobne boli analyzované vlastnosti protokolu SNMP, ktorý je v sučasnosti najpoužívanejším nástrojom využívaným na monitorovanie siete, má však značné nedostatky v prípade nasadenia v úlohe konfiguračného protokolu.

Podrobne bola naštudovaná a popísaná špecifikácia protokolu NETCONF a špecifikácia jeho rozšírenia o podporu asynchrónneho doručovania upozornení. Detailne bola naštudovaná architektúra konfiguračnej platformy *Netopeer*, spolu s existujúcim programovým vybavením, ktoré je momentálne využívané najmä na konfiguráciu pasívnej monitorovacej NetFlow sondy *FlowMon* a paketového filtra *NIFIC* vyvíjaných ako súčasť aktivít projektu Liberouter. Autor bol detailne oboznámený s existujúcimi zdrojovými kódmi jednotlivých komponent konfiguračnej platformy a pokúsil sa identifikovať hlavné problémy a nedostatky existujúceho riešenia.

Diplomová práca predkladá návrh novej architektúry konfiguračnej platformy *Netopeer* 2.0, ktorý rozširuje funkcionalitu najmä o možnosť asynchrónneho doručovania upozornení. Navrhnutá architektúra bola implementovaná vo forme referenčnej implementácie, ktorá potvrdila správnosť a praktickú použiteľnosť návrhu. Súčasne bolo implementované pomoné programové vybavenie a testovacie nástroje, ktoré pomohli overiť použiteľnosť návrhu a samotnej referenčnej implementácie v nasadení za podmienkok ktoré verne simulujú reálne nasadenie systému. Vzhľadom na komplexnosť platformy ako celku sa autorovi nepodarilo zmysluplne využiť jej nové možnosti spojené s asynchrónnym doručovaním upozornení v rámci konfiguračného systému sondy *FlowMon*. Aj napriek tomu bola celková funkcionalita konfiguračného systému sondy značne vylepšená. S užívateľského hľadiska má na tom najväčšiu zásluhu nová verzia webového rozhrania, podporujúca nastavenie väčšiny konfigurovateľných parametrov sondy ako aj definícia dátového modelu konfigurácie vo forme schémy v jazyku RelaxNG, ktorá umožňuje validáciu konfiguračných dát voči referenčnému dátovému modelu.

Vytvorené riešenie bolo analyzované z pohľadu bezpečnosti a boli definované základné doporučenia pre jeho bezpečné nasadenie na vzdialenú konfiguráciu sieťových zariadení. Pri dodržaní doporučených nastavení, poskytuje navrhnutý konfiguračný systém dostatočnú mieru zabezpečenia, vzhľadom na prepokladané nasadenie.

Architektúra konfiguračného systému platformy *Netopeer* 2.0, navrhnutá a implementovaná ako súčasť tejto diplomovej práce, je dobrým a stabilným základom pre ďalší rozvoj celého konfiguračného riešenia v budúcnosti. Možné pokračovanie práce je vo využití asynchrónneho doručovania správ o udalostiach v systéme, v rámci konfiguračného systému zari-

adení vyvíjaných projektom Liberouter. Vytvorené riešenie taktiež umožňuje ďalšie rozširovanie funkcionality platformy, realizované postupným pridávaním podpory pre nové štandardné rozšírenia protokolu NETCONF.

# Literatúra


[1] Pavel Čeleda, Milan Kováčik, Tomáš Konír, Vojtech Krmíček, Petr Špringl, a Martin Žádnik. Copyright © 2006. Cesnet. FlowMon Probe, http://www.cesnet.cz/doc/techzpravy/2006/flowmon-probe/.

[2] Pavel Čeleda, Milan Kováčik, Radek Krejčí, Jaroslav Kysela, a Petr Špringl. Copyright ©2005. Cesnet. Software for NetFlow Monitoring Adapter, http://www.cesnet.cz/doc/techzpravy/2005/netflow.

[3] Radek Krejčí. Konfigurace síťových zařízení protokolem NETCONF. bakalárska práce, Brno, Masarykova Univerzita Fakulta Informatiky, 2007.

[4] Rob Enns. Copyright © 2006. IETF. NETCONF Configuration Protocol, http://www.faqs.org/rfcs/rfc4741.html.

[5] Margaret Wasserman and Ted Goddard. Copyright © 2006. IETF. Using the NETCONF Configuration Protocol over Secure SHell (SSH), http://www.faqs.org/rfcs/rfc4742.html.

[6] S. Chisholm and H. Trevino. Copyright © 2008. IETF. NETCONF Event Notifications, http://www.faqs.org/rfcs/rfc5277.html.

[7] Marek Žižlavský. Webové konguračné rozhrania pre sieťové zariadenia. bakalárska práce, Brno, FIT VUT v Brne, 2007.

[8] The Liberouter Project Team. Copyright © 2006, 2007, 2008 CESNET, z.s.p.o. FlowMon Probe Handbook.

[9] Pavel Satrapa. Nahradí SNMP Netconf?. 2004, http://www.lupa.cz/clanky/nahradi-snmp-netconf/.

[10] J. Schoenwaelder. Copyright © 2003. The Internet Society. Overview of the 2002 IAB Network Management Workshop, http://www.faqs.org/rfcs/rfc3535.html

[11] CESNET z.s.p.o. FlowMon probe - portál FlowMon [online, cit. 2009-01-05]. URL: http://www.flowmon.org/flowmon-probe

[12] CESNET z.s.p.o. Liberouter project web page [online, cit. 2009-01-05]. URL: http://www.liberouter.org

[13] Radej Krejčí, Ladislav Lhotka, Pavel Čeleda a Petr Špringl. Copyright © 2008. Cesnet. Secure Remote Configuration of Network Devices.

[14] J. Case, M. Fedor, M. Schoffstall, J. Davin. Copyright © 1990. Network Working Group. A Simple Network Management Protocol (SNMP), http://www.faqs.org/rfcs/rfc1157.html

[15] J. Postel, J. Reynolds. Copyright © 1983. Network Working Group. Telnet Protocol Specification, http://www.faqs.org/rfcs/rfc854.html

[16] T. Ylonen, C. Lonvick. Copyright © 2006. Network Working Group. The Secure Shell (SSH) Authentication Protocol, http://www.faqs.org/rfcs/rfc4252.html

[17] K. McCloghrie, F. Kastenholz. Copyright © 2000. Network Working Group. The Interfaces Group MIB, http://www.faqs.org/rfcs/rfc2863.html

[18] S. Waldbusser, P. Grillo. Copyright © 2000. Network Working Group. Host Resources MIB, http://www.faqs.org/rfcs/rfc2790.html

[19] World Wide Web Consortium. Copyright © 2008. Extensible Markup Language 1.0, http://www.w3.org/TR/REC-xml/

[20] Wikipedia The Free Encyclopedia [online, cit. 2010-05-24]. URL: http://en.wikipedia.org/wiki/Remote_procedure_call

[21] World Wide Web Consortium. Copyright © 2007. Simple Object Access Protocol (SOAP) 1.1, http://www.w3.org/TR/soap/

[22] M. Rose. Copyright © 2001. The Blocks Extensible Exchange Protocol Core, http://www.faqs.org/rfcs/rfc3080.html

[23] T. Berners-Lee, R. Fielding, L Masinter. Copyright © 2005. Uniform Resource Identifier (URI): Generic Syntax, http://www.faqs.org/rfcs/rfc3986.html

[24] R. Moats. Copyright © 1997. URN Syntax, http://www.faqs.org/rfcs/rfc2141.html

[25] World Wide Web Consortium. Copyright © 2007. XML Path Language (XPath) 2.0, http://www.w3.org/TR/xpath20/

[26] [online, cit. 2010-05-25]. URL: http://unittest-cpp.sourceforge.net

# Dodatok A

# Objavovanie prúdov udalostí

Príklad záznamu komunikácie objavujúcej dostupné prúdy udalostí, správa od klienta:

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get>
    <filter type="subtree">
      <netconf xmlns="urn:ietf:params:xml:ns:netmod:notification">
        <streams/>
      </netconf>
    </filter>
  </get>
</rpc>
```

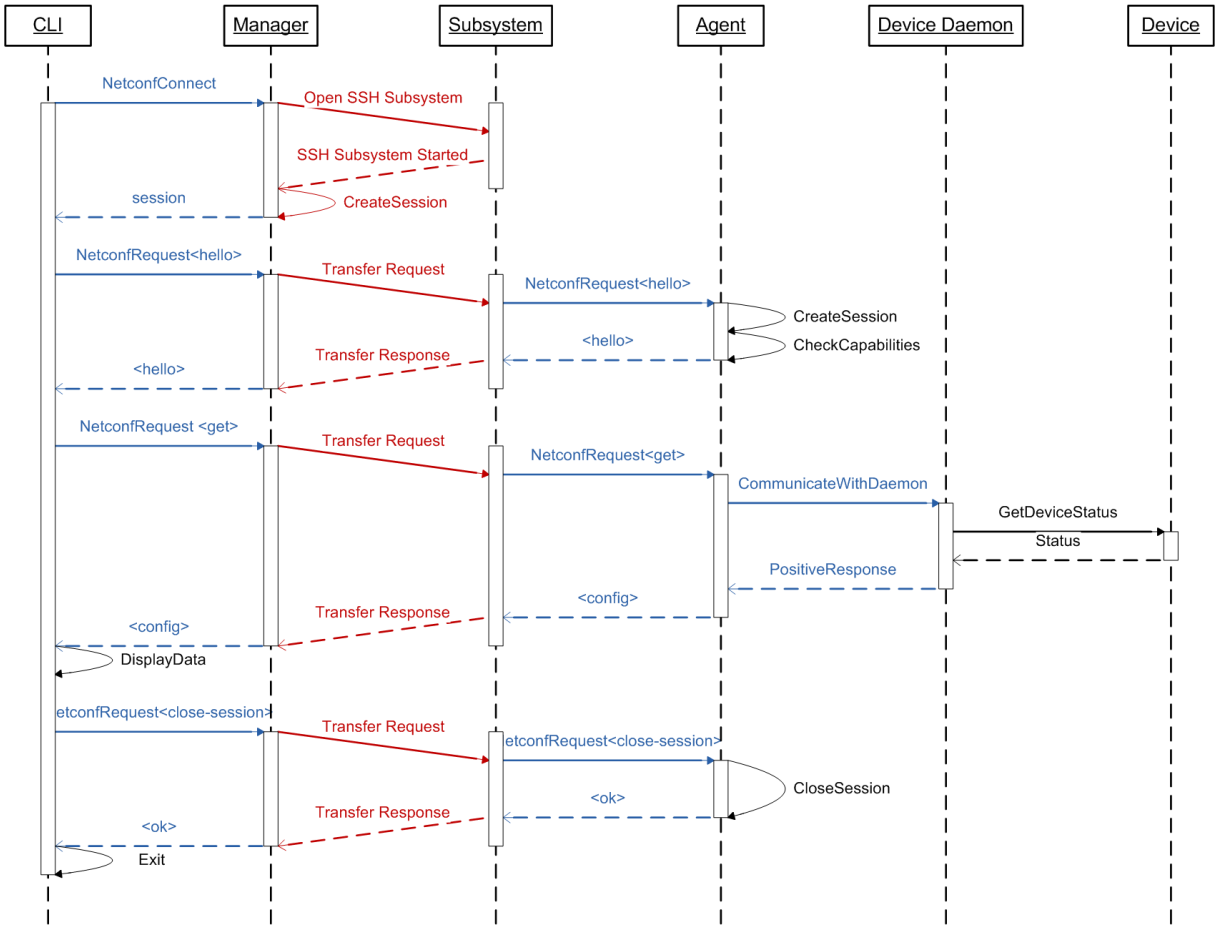
Odpoveď od serveru obsahujúca informácie o troch prúdoch - SNMP a syslog-critical a východziom prúde NETCONF:

```
<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <data>
    <netconf xmlns="urn:ietf:params:xml:ns:netmod:notification">
      <streams>
        <stream>
          <name>NETCONF</name>
          <description>default NETCONF event stream</description>
          <replaySupport>true</replaySupport>
          <replayLogCreationTime>2007-07-08T00:00:00Z</replayLogCreationTime
              >
        </stream>
        <stream>
          <name>SNMP</name>
          <description>SNMP notifications</description>
          <replaySupport>false</replaySupport>
        </stream>
        <stream>
          <name>syslog-critical</name>
          <description>Critical and higher severity</description>
          <replaySupport>true</replaySupport>
          <replayLogCreationTime>2007-07-01T00:00:00Z</replayLogCreationTime
              >
        </stream>
      </streams>
    </netconf>
  </data>
</rpc-reply>
```

# Dodatok B

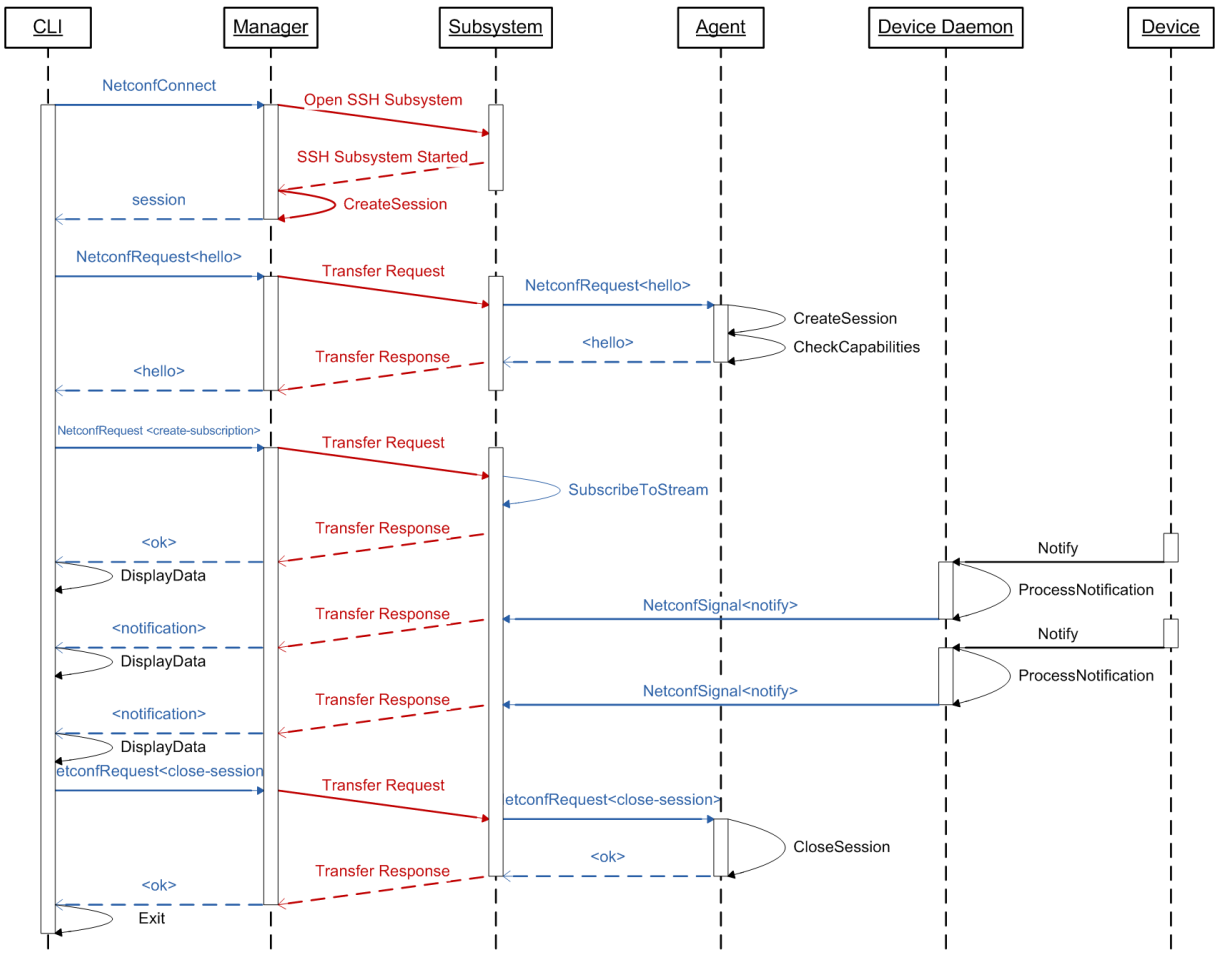
# UML Diagramy

## B.1 Diagram vykonania NETCONF operácie



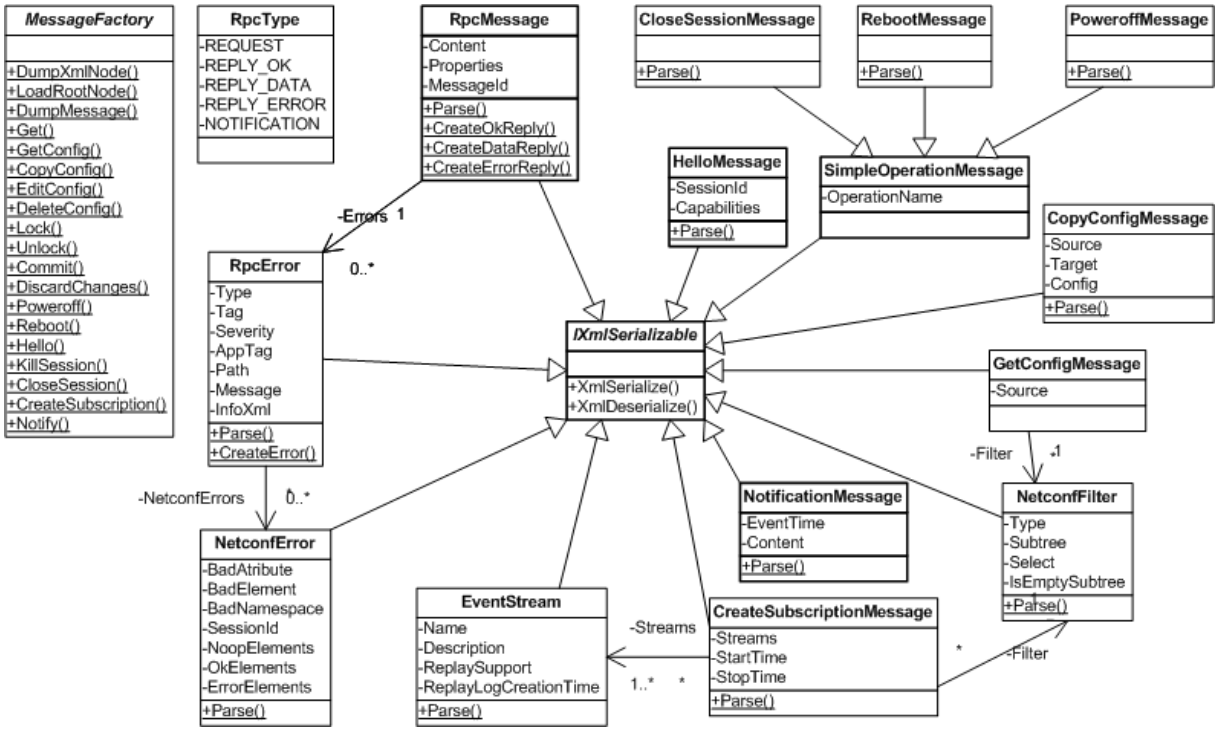

Obr. B.1: Sekvenčný diagram procesu vykonávania NETCONF operácie

## B.2 Diagram asynchrónneho doručovania upozornení




Obr. B.2: Sekvenčný diagram procesu asynchrónneho doručovania upozornení

## B.3 Diagram tried knižnice libnetconf




Obr. B.3: Diagram tried knižnice libsshwrapper