Mendelova univerzita v Brně
Provozně ekonomická fakulta

Automatizované generovanie sieťovej konfigurácie v počítačovej sieti Mendelovej univerzity v Brne

Diplomová práca

Vedúci práce:
Ing. Martin Pokorný, Ph.D.

Autor práce:
Bc. Daniel Smolinský

Brno 2016

SEM VLOŽIŤ ZADANIE

Poďakovanie

Chcel by som poďakovať Ing. Martinovi Pokornému, Ph. D., môjmu vedúcemu práce za odborné vedenie, trpezlivosť a cenné rady, ktoré mi dopomohli k vytvoreniu tejto diplomovej práce. Ďalej by som chcel poďakovať UIT MENDELU za ich čas a poskytnutie dôležitých informácií, pripomienok, ktoré boli kľúčové pre realizáciu diplomovej práce. V neposlednom rade ďakujem svojej rodine, priateľke a blízkym za podporu a pozitívne myslenie, ktoré mi neustále dodávali.

Čestné prehlásenie

Prehlasujem, že som prácu: **Automatizované generovanie sieťovej konfigurácie v počítačovej sieti Mendelovej univerzity v Brne**
vypracoval/a samostatne a všetky použité zdroje a informácie uvádzam v zozname použitej literatúry. Súhlasím, aby moja práca bola zverejnená v súlade § 47b zákona č. 111/1998 Sb., o vysokých školách v znení neskorších predpisov a v súlade s platnou *Směrnicí o zveřejňování vysokoškolských závěrečných prací*.
Som si vedomý/a, že sa na moju prácu vzťahuje zákon č. 121/2000 Sb., autorský zákon, a že Mendelova univerzita v Brně má právo na uzatvorenie licenčnej zmluvy a použitie tejto práce ako školského diela podľa § 60 odst. 1 autorského zákona.
Ďalej sa zaväzujem, že pred spísaním licenčnej zmluvy o použití diela inou osobou (subjektom) si vyžiadam písomné stanovisko univerzity, že predmetná licenčná zmluva nie je v rozpore s oprávnenými záujmami univerzity a zaväzujem sa uhradiť prípadný príspevok na úhradu nákladov spojených so vznikom diela, a to až do ich skutočnej výšky.

V Brne dňa: 20. 5. 2016

.....

podpis

Abstract

SMOLINSKÝ, Daniel: Automated generation of network configuration at computer network of Mendel University in Brno. Diploma thesis. Brno. 2016.

The diploma thesis focuses on the creation backend of new network management system of Mendel University. Backend consists from automatic configuration for the DHCP server version 4 and version 6, DNS server and network access devices - switches. After specification of the requirements laid down of UIT MENDELU is designed, implemented and tested a new proposal in the area of network laboratory of MENDELU. Finally, the thesis is evaluated and discussed the results of future development opportunities.

Keywords

system network management, IPv6, automated configuration, Perl

Abstrakt

SMOLINSKÝ, Daniel: Automatizované generovanie sieťovej konfigurácie v počítačovej sieti Mendelovej univerzity v Brne. Diplomová práca. Brno. 2016.

Diplomová práca sa zameriava vytvorením backend-u nového systému pre správu siete Mendelovej univerzity. Backend pozostáva z automatickej konfigurácie pre DHCP server verzie 4 a verzie 6, DNS server a sieťové prístupové zariadenia prepínače - switch-e. Po špecifikácii požiadavkou stanovených UIT MENDELU je navrhnutý, implementovaný a testovaný nový návrh v prostredí sieťového laboratória MENDELU. Na záver je práca zhodnotená a diskutované výsledky pre možnosti ďalšieho vývoja.

Kľúčové slová

systém pre správu siete, IPv6, automatické generovanie konfigurácie, Perl

Obsah

1 Úvod

S rastúcim počtom zariadení, ktoré sa snažíme denno-denne zapojiť do internetu a rôznych lokálnych sietí v škole, v zamestnaní či na verejnosti, sa tieto siete neustále rozširujú, rastú. V tomto zmysle je čoraz ťažšie všetky tieto prvky siete, zariadenia, užívateľov spravovať a celú sieť mať pod kontrolou a preto je potrebný systém správy siete.

Systém, ktorý je riešením tejto práce, alebo súčasný, ktorý je momentálne v používaní administrátormi, sa nachádza v prostredí Mendelovej univerzity v Brne. V tejto sieti sa nachádza približne niečo okolo 10 000 užívateľov, do ktorých sa zahŕňajú samotní študenti univerzity a zamestnanci, externé subjekty, ktoré majú možnosť prenájmu miestností, osoby na internátoch univerzity a podobne. Sieť okrem užívateľov sa skladá samozrejme z počítačov, notebookov, tlačiarní, serverov, na ktoré sú napojené kontrolné čidla, bezdrôtových prvkov, switchov či routerov, diskové polia, kamery a iné, čo do správnej, veľkej siete patrí.

Prostredie univerzity sa skladá z niekoľkých hlavných budov, internátov, menz, študovní či knižníc. V tomto prostredí, ako bolo spomínané, je zasadený súčasný systém, ktorý ale momentálne už nespĺňa dnešné požiadavky.

Je potrebné spomenúť, že práca vznikla za pomoci spolupráce ústavu informačných technológií, ktorí boli aj hlavným zadávateľom tejto diplomovej práce. Hlavné zadanie pozostávalo z vytvorenia nového systému pre správu siete Mendelovej univerzity. Pre veľký rozsah zadania, bol projekt rozdelený na dve časti – frontend, ktorý rieši grafické rozhranie spolu s CRUD databázy a backend – ktorý pozostáva z funkcionality na pozadí systému. Prvá časť projektu je riešená študentom Bc. Oliverom Horečným a kapitoly analýza súčasného prostredia a návrh databázy sa môžu v určitých veciach zhodovať, keďže spolu vyvíjame jeden systém.

2 Cieľ práce

Cieľom práce je na základe požiadavkou UIT MENDELU navrhnúť, implementovať a overiť automatickú konfiguráciu pre DHCP server verzie 4 a verzie 6, DNS server a prístupové prepínače - switch-e.

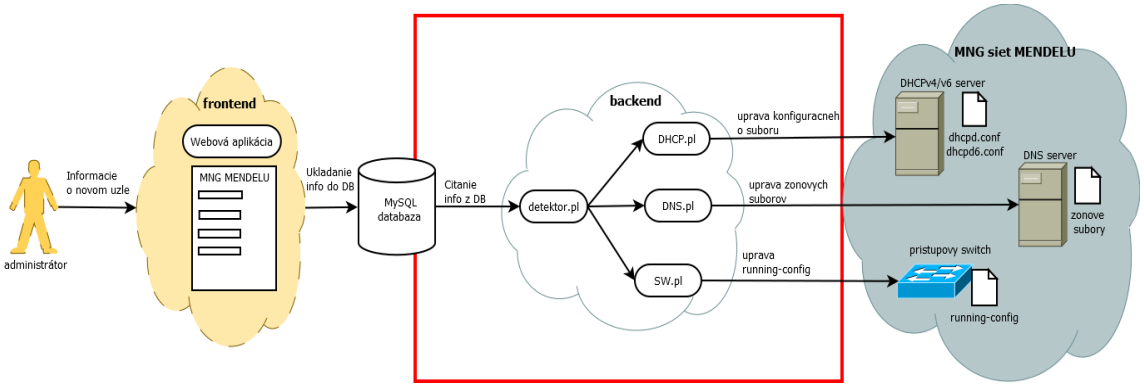
Je potrebné analyzovať súčasný stav počítačovej siete univerzity, vypracovať návrh a uskutočniť testovanie v podmienkach sieťového laboratória. Na záver zhodnotiť navrhované riešenie, kde sa nachádzajú chyby, prípadne čo je dobré v budúcnosti vylepšiť.

3 Špecifikácia požiadavkou

3.1 Súhrnný popis

Úlohou vyvíjaného systému je evidencia a konfigurácia koncových zariadení v univerzitnej sieti MENDELU. Administrátor siete by mal mať možnosť vytvárať, spravovať, respektíve upravovať tieto zariadenia v systéme pomocou webového rozhrania, ktoré je riešené v diplomovej práci študenta Bc. Olivera Horečného.

Nástroj by mal byť schopný vyčítať požiadavky vo fronte v databáze, ktoré boli zadané administrátorom. Následne systém musí rozpoznať a spustiť potrebné skripty, respektíve subpožiadavky (DHCP.pl, DNS.pl, SW.pl), ktoré majú spracovať požadovanú úlohu. Toto je hlavným cieľom tejto diplomovej práce, viď. červený obdĺžnik na Obr. 1.



Obr. 1 Nákres princípu vyvíjaného systému

3.2 Funkčné požiadavky

V tejto časti práce sú predstavené funkčné požiadavky, ktoré definujú funkcionalitu backend-u budúceho systému pre evidenciu a správu zariadení.

Automatické čítanie fronty z databázy
Musí prebiehať komunikácia medzi databázou, backend-om a externými entitami DHCP serverom, DNS serverom, switch-mi a práve skript *detektor.pl* v opakujúcom sa určitom intervale kontroluje, či sa nachádzajú nejaké požiadavky vo fronte databázy alebo nie. Na základe typu požiadavku musí spustiť požadované skripty *dhcp.pl*, *dns.pl* alebo *sw.pl*. Skript *detektor.pl* nesmie byť spustený dvakrát naraz.

Automatická úprava DHCP servera
Systém automaticky upravuje konfiguračný súbor (*dhcpd.conf*, *dhcpd6.conf*) DHCP servera. Po vyčítaní potrebných údajov z databázy, vytvorí zálohu aktuálnych súborov servera, upraví konfiguračný súbor a reštartuje službu. V prípade

neúspechu je schopný nahrať predchádzajúcu konfiguráciu pre zachovanie funkčnosti. V každom prípade ukončenia skriptu, kontaktuje *detektor.pl*, ktorému predáva informácie o dokončení požiadavku, prípadne chyby.

Automatická úprava DNS servera
Tak ako prebieha úprava konfiguračných súborov u DHCP serverov, identická úprava prebieha i pri úprave zónových súborov DNS serverov pre záznamy IPv4 i IPv6 adries. Jedná sa konkrétne o zápis A, AAAA a PTR záznamov. Po vyčítaní informácií z databázy, *dns.pl* uskutočňuje zálohu, úpravu a posiela spätnú väzbu skriptu *detektor.pl*.

Automatická konfigurácia nových sieťových prvkov – prepínačov
Pri registrácii nových sieťových prvkov do systému, respektíve sa jedná o prepínače, bude im nahratá čistá, predvolená konfigurácia. Táto predvolená konfigurácia pozostáva zo zadaných požiadavkou administrátora a zahŕňa:

a) Vytvorenie predvolenej (default) VLAN 169.
b) Priradenie sieťových rozhraní zariadenia do *mode access*.
c) Nastavenie VLAN 169 na sieťové rozhrania zariadenia.
d) Nastavenie *port-security* na sieťové rozhrania.
e) Vypnutie rozhraní zariadenia (*shutdown*).

Úprava/konfigurácia sieťového zariadenia – prepínača
Konfigurácia prebieha na princípe stiahnutia aktuálneho *running-config* zo zariadenia, z ktorého sa vytvorí kópia. Na originálnej konfigurácii sa potom urobia potrebné, požadované zmeny. Môže sa jednať o úpravu sieťového rozhrania zariadenia do zvolenej VLAN-y, zmena *port-security* daného rozhrania. Po úprave je nový *running-config* nahratý na požadované zariadenie a musí byť prevedená kontrola v podobe kontrolných súčtov konfiguračných súborov. Pri neúspechu je nahratá predchádzajúca, zálohovaná konfigurácia. Výsledok sa predáva skriptu *detektor.pl*.

3.3 Nefunkčné požiadavky

- Z dôvodov komunikácie skriptov a sieťových zariadení musí byť nastavená tzv. manažment IP adresa na každom prepínači v sieti. Bez tohto úkonu by nebolo možné realizovať automatické konfigurácie, ktoré sú jedným z cieľov práce.
- V rovnakom zmysle, komunikácia skriptov s databázou, musia byť nastavené korektné prihlasovacie údaje do databázy a predané príslušným skriptom. Prístup k databáze musí byť zabezpečený 24/7.
- Pre uskutočnenie komunikácie medzi backend-om a jednotlivými externými entitami (DHCP server, DNS server, switch-e) musí byť zrealizovaná konektivita v podobe fyzickej kabeláže.

- Pre správne fungovanie systému a správny priebeh kontroly fronty v databáze, je stanovený opakujúci sa čas pre hlavný skript *detektor.pl* = pravidelné spustenie každé 2 minúty.
- Pre úpravu konfigurácií prepínačov musí byť zriadený fungujúci TFTP server.
- Musí byť zaobstaraný počítač, respektíve server, na ktorom pobeží backend-ová časť systému.

4 Popis technologického aparátu

4.1 Základné pojmy

IPv4/v6
V dnešnej dobe sa používajú dve verzie protokolu IP, široko známy IP verzie 4 a IP verzie 6, ktorý bol navrhnutý ako náhrada za IPv4. Hlavný rozdiel prichádza v náraste počtu adries, ktorá vychádza od dĺžky danej adresy. [1]

Ďalšou výhodou a novou funkciou, s ktorou prichádza protokol verzie 6, je zmena hlavičky paketu. Tá tvorí polovicu adresy a je zarovnaná na hranicu 64 bitov, čo značne zvyšuje rýchlosť spracovania. [1]

IPv4 protokol často využíva všade smerujúce vysielanie, čo vedie k problémom, napríklad všade smerujúca búrka. Nevýhodou taktiež je, že pri takomto type vysielania, musí byť prerušená činnosť každého zariadenia v sieti a zariadenie odpovedá na správu, či je mu určená alebo nie. Protokol IPv6 všade smerujúce vysielanie nepozná a naopak používa viacsmerovú prevádzku. Zakladá sa tu taktiež ďalší nový typ a to jednosmerné vysielanie (anycast), kedy komunikácia umožňuje prideliť rovnakú adresu na viac zariadení a paket je predaný najbližšiemu hostiteľovi tejto adresy. [2]

Core, Distribution, Access layer
Firma Cisco definuje model známy ako hierarchický medzisieťový model, ktorý zjednodušuje vybudovanie spoľahlivej, škálovateľnej a čo najmenej nákladnej siete. [3], [4]

- Chrbticová vrstva (Core layer): jedná sa o vysokorýchlostnú časť siete, hierarchicky spája niekoľko vrstiev návrhu areálu a preto na spoľahlivosť je kladený vysoký dôraz. Z tohto dôvodu sú zavádzané redundantné spoje a táto vrstva umožňuje taktiež pripojenie k Internetu.

- Distribučná vrstva (Distribution layer): sa nachádza medzi chrbticovou a prístupovou vrstvou, od ktorej obdŕža a agreguje dátovú prevádzku. Dôležitou funkciou je kontrola sieťovej prevádzky (ACL[1], VLAN) a prístupová politika ku zvyšku siete.

- Prístupová vrstva (Access layer): predstavuje okraj siete a vykonáva základné sieťové funkcie ako zabezpečenie portov, kvalitu služieb (QoS) a primárnou funkciou vrstvy je poskytnúť prístup k sieti pre daných užívateľov.

[1] ACL (Access Control List) – zoznam oprávnení, ktorý určuje prístupy k danému objektu, či sa povoľuje alebo blokuje prevádzka

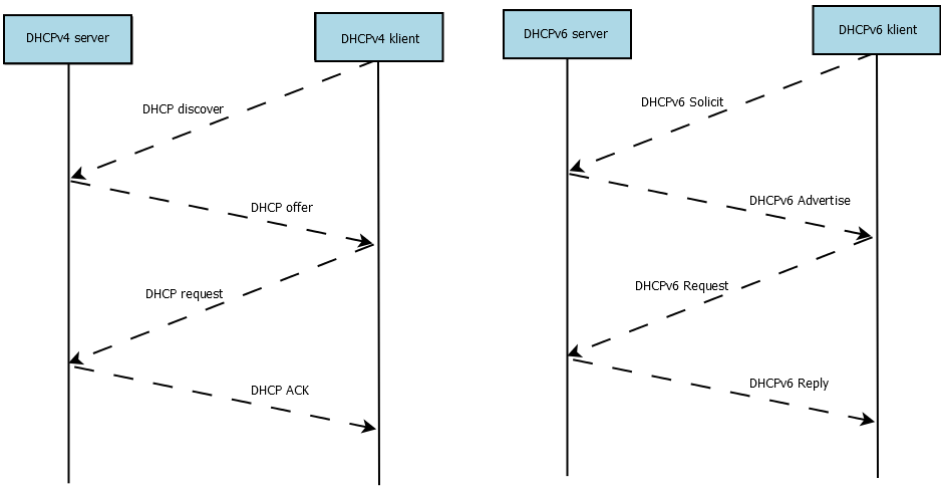
Databáza MySQL
Najpoužívanejšia opensource[2] relačná databáza, ktorá radí medzi svoje výhody jej stabilitu a rýchlosť. Pre svoju ľahkú implementovateľnosť a výkon vďačí práve tomu, že je to voľne šíriteľný software. Je to databázový systém, ktorý má širokú podporu programovacích jazykov od C, C++, Java, Perl, PHP a ďalšie. MySQL býva často spomínaná v skratke LAMP, čo označuje spojenie operačného systému Linux, webového serveru Apache, databázu MySQL a programovací jazyk. Prináša taktiež širokú škálu možností zabezpečenia a konfigurácie nad takmer všetkými operáciami, napríklad určiť si predvolený jazyk, predvolený port, umiestnenie dátového skladu, množstvo alokovanej pamäte a iné. MySQL má schopnosť ovládať ako dlho sa bude pokus o pripojenie vykonávať, či bude prerušený a je možné i nastaviť maximálnu povolenú veľkosť paketov. [5]

4.2 Sieťové služby

DHCPv4/v6 (Dynamic Host Configuration Protocol)
Manuálne prideľovanie IP adries koncovým staniciam je s rastúcim počtom pripojených zariadení samozrejme nereálne. Protokol DHCP pracuje na princípe klient-server. Klient je zariadenie, ktoré sa pripája do siete a žiada o pridelenie IP adresy a ďalších konfiguračných parametrov. Server predstavuje program, ktorý spravuje, respektíve prideľuje IP adresy a odpovedá na dotazy klientov. [6]
Rozdiel medzi verziou protokolu DHCPv4 a DHCPv6 nastáva v momente dynamickej komunikácie prideľovania IP adresy. Komunikáciu DHCPv4 i v6 možno rozdeliť na štyri kroky podľa posielaných príkazov [7]:

- DHCPv4 discover/DHCPv6 solicit – klient posiela prvý dotaz pre lokáciu serverov,
- DHCPv4 offer/DHCPv6 advertise – odpoveď servera, ktorá obsahuje ponúkanú adresu,
- DHCPv4 request/DHCPv6 request – klient odpovedá, že si zvolil konkrétnu ponúknutú adresu,
- DHCPv4 ACK/DHCPv6 reply – server potvrdzuje prijatie voľby adresy na danú dobu.

[2] Opensource – otvorený software, znamená legálny, dostupný zdrojový kód verejnosti



Obr. 2 Príklad komunikácie DHCPv4/v6 servera
Zdroj: Síťové aplikace a jejich architektura, 2014.

DHCP pool adries VS pevné (statické) adresy

DHCP server či už verzie 4 alebo verzie 6, môže pracovať v dvoch rôznych režimoch prideľovania IP adries.

Prvým spôsobom je dynamické prideľovanie z administrátorom určeného rozsahu, tzv. pool, napríklad 192.168.100.5 – 192.168.100.240. Nové zariadenie, pripojené do danej siete, sa pýta DHCP servera a ten mu pridelí jednu adresu z vyššie spomínaného rozsahu. Server sám kontroluje, ktoré adresy sú už pridelené zariadeniam alebo ktoré adresy sú k dispozícii a tak zaistí konzistenciu, že dve rôzne zariadenia nemajú rovnakú IP adresu. Nevýhodou tohto spôsobu je, že zariadenie pri ďalšom pripojení do siete môže obdŕžať úplné inú adresu z uvedeného rozsahu. [8]

Druhý spôsob je tzv. statické, fixné pridelenie IP adresy k MAC adrese daného zariadenia. Práve táto dvojica adries IP – MAC vytvoria spolu väzbu a pri každom pripojení, zariadenie obdŕža rovnakú, pridelenú adresu a je tak možné dané zariadenia v prípade potreby dohľadať. Nevýhodou je, že administrátor musí poznať MAC adresu zariadenia. [9]

Konfigurácia DHCPv6

Informácie, ktoré potrebuje koncový užívateľ:
1. Prefix
2. Interface ID
3. Prefix length
4. GW IPv6 address
5. DNS information

U verzie DHCPv6 je cieľom to, aby sa nemuselo prakticky nič konfigurovať ručne, tzv. metóda plug-and-play. Administrátor má na výber dva rôzne spôsoby: stavová a bezstavová automatická konfigurácia. [10]

Stavová konfigurácia – server odpovedá na dotazy klienta a oznámi mu všetko, čo o konkrétnej sieti sám vie. Na tomto princípe fungujú dneska siete s adresami IPv4, kde klient prijíma všetko potrebné k pripojeniu do siete. V sieti IPv6 ale nastáva problém a to s predvolenou bránou (default gateway), ktorú súčasný DHCPv6 server nedokáže poskytnúť. Dôležitým prvkom, ktorý vstupuje do komunikácie sú správy smerovačov, konkrétne nazývaných RA (Router Advertisement). Pomocou týchto správ si dokáže klient zistiť, respektíve dorátať zvyšný chýbajúci parameter, predvolenú bránu. [10]
Bezstavová konfigurácia – V tomto prípade je DHCPv6 len doplňujúcim mechanizmom a primárne informácie klient obdŕža z RA smerovačov. Konkrétne prefix siete (prefix + prefix length) a predvolený smerovač GW. Zaujímavosť nastáva i v momente samotného určenia identifikátora rozhrania (interface ID). Ten vznikne pomocou IEEE stanoveného identifikátoru *Extended Unique Identifier 64-bit* (EUI-64), ktorý pomocou klientovej MAC adresy vygeneruje jeho interface ID. Ako bolo spomínané, DHCPv6 je len doplňujúcim prostriedkom, v tomto prípade pre získanie DNS informácií. [11]

Tab. 1 Zdroj získavania sieťových informácií DHCPv6

		Druh DHCPv6 konfigurácie	
		Stavová konfigurácia	**Bezstavová konfigurácia**
sieťová informácia	**Prefix**	DHCPv6	RA
	interface ID	DHCPv6	EUI-64
	Prefix length	DHCPv6	RA
	GW address	RA	RA
	DNS information	DHCPv6	DHCPv6

Ďalšia dôležitá informácia, ktorá sa nachádza v správe RA od smerovačov, sú tzv. príznaky. Tie dávajú pokyn klientovi, či potrebné parametre obdŕža z DHCPv6 servera, alebo v sieti DHCPv6 server neposkytuje nič, respektíve nie je. Jedná sa konkrétne o príznaky *M – Managed flag* a *O – Other flag*. [10]

Tab. 2 Význam príznakov pri bezstavovej konfigurácii

M	**O**	**Význam**
1	0	Stavová konfigurácia adries => adresy a ďalšie parametre prideľuje DHCPv6
0	1	Bezstavová konfigurácia s kombináciou DHCPv6 pre DNS
0	0	DHCPv6 nie je k dispozícii

Zdroj: IPv6, 2011.

DUID

Významnú úlohu v DHCP verzie 6 hrá identifikácia klientov ale i serverov a z toho dôvodu sa zavádza pojem *DHCP Unique Identifier* (DUID). Ide o jednoznačný identifikátor účastníka v sieti a práve iba jeden unikátny má každý klient či server. Autori vytvorili tri druhy, respektíve spôsoby definovania DUID [12]:

- Výrobcom určený jedinečný identifikátor (výrobné číslo).
- Linková adresa + čas (DUID generované od dátumu 1. 1. 2000 v sekundách).
- Linková adresa (odpovedá používaniu IPv4).

DNS (Domain Name Server)

Adresovanie a preklad adries je neodmysliteľná súčasť Internetu, bez ktorej sa žiadna komunikácia neobíde. Základnou úlohou DNS je prevod doménových adries, napríklad pc-novak.mendelu.cz na IP adresy. Služba DNS teda obsahuje databázu všetkých doménových adries a príslušných IP adries. Táto databáza je tvorená viacero počítačmi zvanými doménové servery (nameservery). [6]
Medzi základné služby DNS patrí:

- Preklad doménových adries na IP adresy.
- Preklad IP adries na doménové adresy.
- Preklad aliasov počítačov na tzv. kanonické mená.
- Určenie poštového serveru pre danú doménu.
- Delegovanie správy domén na jednotlivé subjekty.

Práca sa zameriava práve na prvé dva body spomínaných základných služieb servera DNS.

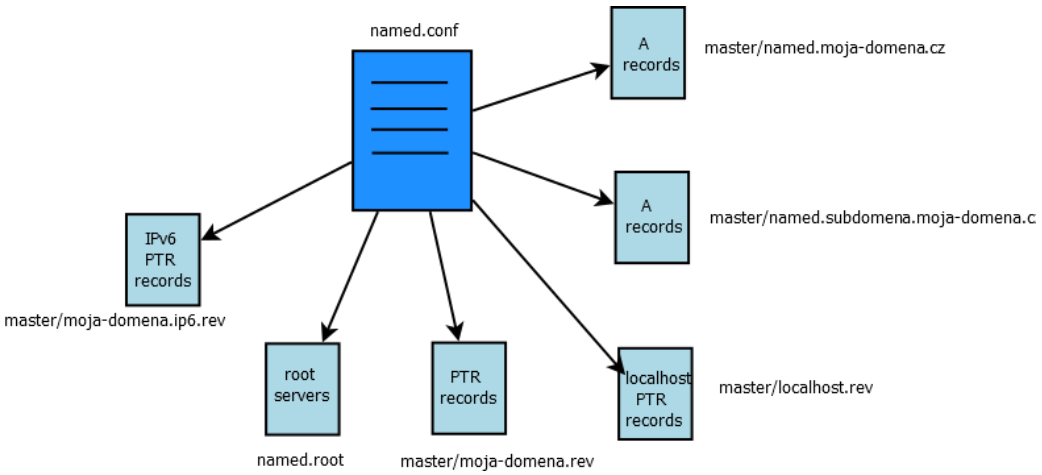
Tab. 3 Prehľad štandardov pre základné typy záznamov DNS

Typ záznamu	Názov	Štandard
SOA	Start of Authority	RFC 1034
NS	Name Server	RFC 1034
A	IPv4 Address	RFC 1034
MX	Mail Exchanger	RFC 1034
CNAME	Canonical Name	RFC 1034
PTR	Domain Name Pointer	RFC 1034
AAAA	IPv6 Address	RFC 3596

Zdroj: Síťové aplikace a jejich architektura, 2014.

Konfigurácia serveru DNS

Veľmi rozšírenou distribúciou, ktorú používa i Mendelova univerzita, na konfiguráciu DNS servera je DNS BIND (Berkeley Internet Name Domain). Základná konfigurácia je tvorená hlavným konfiguračným súborom *named.conf* a piatimi, respektíve šiestimi základnými súbormi v prípade IPv6, viď Obr. 3.



Obr. 3 Príklad konfigurácie serveru DNS BIND
Zdroj: Síťové aplikace a jejich architektura, 2014.

V tejto diplomovej práci pri manipulácii a konfigurácii DNS servera je potrebné poznať nasledovné konfiguračné súbory:
named.conf obsahuje informácie a doménach a zónových súboroch, ktoré server obsahuje, na ktorom rozhraní odpovedá na dotazy a podobne.
Priame zónové súbory obsahujú záznamy typu SOA[3], NS, A, AAAA, MX, CNAME a ďalšie.
Reverzné zónové súbory obsahujú záznamy typu SOA, NS alebo PTR.

Reverzný zónový súbor ipv6, ktorý pracuje i so záznamami AAAA, respektíve s IPv6 adresami, obsahujúci preklady pre IPv6. Pri realizácii DNS servera s IPv6 je potrebné mať vytvorené ďalší reverzný zónový súbor. [13]

VPN
Virtuálnu privátnu sieť VPN (Virtual Private Network) je možné použiť pre riešenie rôznych požiadavkou, ktoré v sebe zahŕňajú potrebu bezpečnej komunikácie. Tunelovanie môžeme uskutočniť na rôznych vrstvách architektúry. [14]

- *L2 vrstva*: tunelovanie rámcov PPP Internetom do domácej siete. Najčastejšie protokolom L2TP alebo mechanizmom GRE (Generic Route Encapsulation). Tunel môže iniciovať klient sám alebo ho vytvára prístupový server bez iniciatívy klienta.
- *L3 vrstva*: zapuzdrenie IP datagramov do iného datagramu. Všetka konfigurácia sa uskutočňuje mimo vlastnú prevádzku, predom, obvykle manuálne. Tunel na rozdiel L2 tunelov, kedy musí byť vytvorený, udržiavaný a ukončený protokolom, na L3 tunel nemusí mať udržiavaciu fázu.

[3] SOA (Start Of Authority) – Záznam, ktorý obsahuje informácie o uložení autoritatívnych dát pre danú zónu, väčšinou prvý záznam v zónovom súbore

IPsec
IPsec sa stará o bezpečný komunikačný kanál po IP medzi klientom a cieľovým systémom na sieťovej L3 vrstve. Používa autentizačné záhlavie, počíta hodnotu autentizácie cez celý datagram, vrátane zdrojovej a cieľovej IP adresy. Akákoľvek zmena IP adresy vedie k neúspešnému spojeniu. IPsec má úroveň šifrovania prednastavenú na 128bitový kľúč. [14]
IPsec využíva tri hlavné protokoly [15]:

- Authentication Header (AH): zaisťuje integritu a autentizáciu dát.
- Encapsulating Security Paylad (ESP): zaisťuje dôvernosť, autentizáciu zdroja a integritu.
- Security Association (SA): skupina algoritmov zabezpečujúcich bezpečnú komunikáciu, využíva sa framework *Internet Security Association and Key Management Protocol* (ISAKMP) s protokolom *Internet Key Exchange* (IKE) pre vyjednanie atribútov obsahujúcich zvolený šifrovací algoritmus, dobu platnosti kľúča, kompresiu a spôsob zapuzdrenia.

EtherChannel
EtherChannel je termín, ktorý primárne je používaný firmou CISCO. Obdobným pomenovaním je Link Aggregation, ktorý je stanovený v štandarde IEEE 802.1ax, predtým 802.1ad. Táto technológia umožňuje agregáciu liniek, čo znamená spojenie niekoľko fyzických ethernetových portov do jednej logickej linky. [16]

Pri konfigurácii EtherChannel-u je možné použiť buď proprietárny protokol CISCO Port Aggregation Protocol PAgP, alebo protokol práve spomínaného štandardu IEEE 802.1ax a to Link Aggregation Control Protocol LACP. Protokoly slúžia k automatickému vyjednávaniu a vytvoreniu EtherChannel-u. [17]

- PAgP: protokol vytvorený firmou CISCO a podporovaný takmer len na zariadeniach CISCO. Podporuje vytvorenie EtherChannel-u iba z rozhraní na jednom prepínači, nie v rámci stack-u[4]. Pracuje v dvoch módoch – *desirable* kedy vyjednáva spojenie a *auto*, kedy je v pasívnom postavení.
- LACP: podobne ako PAgP pracuje v dvoch módoch – *active*, kedy sa snaží o vytvorenie spojenia a *passive*, čaká na začiatok vyjednávania.

4.3 CISCO switch

Diplomová práca sa odohráva v prístupovej vrstve (access layer) hierarchického medzisieťového modelu spomínaného vyššie. Z toho dôvodu je dôležité ovládať základnú konfiguráciu prístupových prepínačov.

[4] Stack – technológia umožňujúca spojiť fyzické CISCO prepínače do jedného celku a vytvoriť jeden veľký prepínač vlastniaci všetky porty

SVI – Switch Virtual Interface
Virtuálne rozhranie je VLAN port prepínača umožňujúce smerovanie. V tomto prípade neexistuje fyzické rozhranie a zabezpečuje spracovanie paketov na tretej vrstve všetkých portov prepínača spojené s danou VLAN. Administrátor následne môže nastaviť na SVI port IP adresu a masku siete. Týmto je získaný vzdialený prístup na požadované zariadenie. [18]

TELNET
Telnet protokol umožňuje nastaviť prepojenie TCP/IP medzi dvomi zariadeniami. Užívateľ potom môže pomocou prihlasovacích údajov cieľového zariadenia nadviazať spojenie a poslať príkazy z jedného zariadenia do druhého. [19]

Switch port
Jednotlivé porty prepínača sú východiskovo switch porty zaradené do jednej alebo viacerých VLAN. Parametre, ktoré nastavujú charakteristiku a chovanie prepínača, sa konfigurujú príkazom *switchport* a môže pracovať v jednom z nasledujúcich režimov [20]:

- Access
- Trunk
- Dynamic
- Tunnel

Port security
Je metóda, ktorá umožňuje zabezpečenie prístupu do siete na základe užívateľovej MAC adresy. Nastavuje sa na konkrétny port prepínača a kontroluje, či rámce prichádzajú z povolenej adresy. V prípade, ak sa pripojí na port zariadenie s inou MAC adresou, nebude môcť komunikovať v danej sieti. [21], [22]

4.4 Jazyk Perl

Patrí k najprenositeľnejším jazykom, ktoré dnes existujú. Pôvodne bol Perl navrhnutý pre platformy s operačným systémom UNIX, ale dnes je možné ho spustiť i pod inými platformami ako Windows, Macintosh, MS DOS a iné. V súčasnosti je aktuálna verzia Perlu rady 5, ktorá umožňuje programovanie vo viacerých súbežných vláknach alebo rozširuje jazyk formou samostatných modulov. Samotný jazyk a väčšina programového vybavenia je bezplatná, ktoré je možné naliezť na celosvetovej zdieľanej sieti s názvom Comprehensive Perl Archive Network (CPAN). [23]

Veľmi silným nástrojom, ktorý dáva jazyk Perl do popredia sú regulárne výrazy. Perl dokáže pracovať na rovnakej a i vyššej výkonnosti a s viacerými možnosťami s porovnaním napríklad programom awk či grep, čo sú len jednoúčelové nástroje na spracovanie textu. [24]

Pri vytváraní programu v jazyku Perl nie je potreba žiadny špeciálny prostriedok, napríklad vývojové prostredie. Program je obyčajný text a je ho možné teda vytvárať v ľubovoľnom textovom editore. Pri programovaní je užitočné pracovať v dvoch nasledovných režimoch, ktoré pomáhajú programátorovi odhaliť prípadné chyby. Jeden z nich je *režim varovania* (warnings) s prepínačom *–w,* upozorňuje na výpis nedefinovanej hodnoty a podobne na štandardný chybový výstup. Druhým je *striktný režim*, ktorý sa zavádza modulom *strict*. Jeho úlohou je upozorniť na prácu s premennými, u ktorých nie je známy rozsah platnosti alebo manipulácia so symbolickými odkazmi. Tieto dva režimy sú veľmi nápomocné a sú taktiež používané pri vývoji jednotlivých skriptov tejto práce. [23]
Dôležitými prvkami, ktoré je potrebné poznať a mať naštudované pre realizáciu projektu a sú používané pri vývoji sú:

- Práca so spomínanými modulmi CPAN – inštalácia a následné použitie pri vývoji skriptov.
- Komunikácia s databázou – pripojenie, CRUD úkony.
- Komunikácia so zariadeniami spoločnosti CISCO – pripojenie, posielanie príkazov pre požadované akcie.
- Práca so súbormi – otvorenie, čítanie, zápis, vyhľadanie konkrétnych pozícií.
- Spracovanie a komunikácia IPC (inter process communication) medzi jednotlivými skriptami.
- Práca s dátovou štruktúrou pole.
- Vytváranie a volanie funkcií.
- Volanie systémových príkazov a spracovanie ich výstupov.
- V neposlednej rade, práca so spomínanými regulárnymi výrazmi.

5 Analýza súčasného stavu

5.1 Sieť MENDELU

Pod pojmom sieť Mendelovej univerzity je potreba si predstaviť v prvom rade lokality, ktoré ju tvoria. Skladá sa z nasledujúcich častí:

- Hlavný areál Černá pole
- Areál akadémie FRRMS
- Internát Jána Amosa Komenského (JAK)
- Internát Jozefa Taufera (TAK)
- Internát Lednice na Moravě

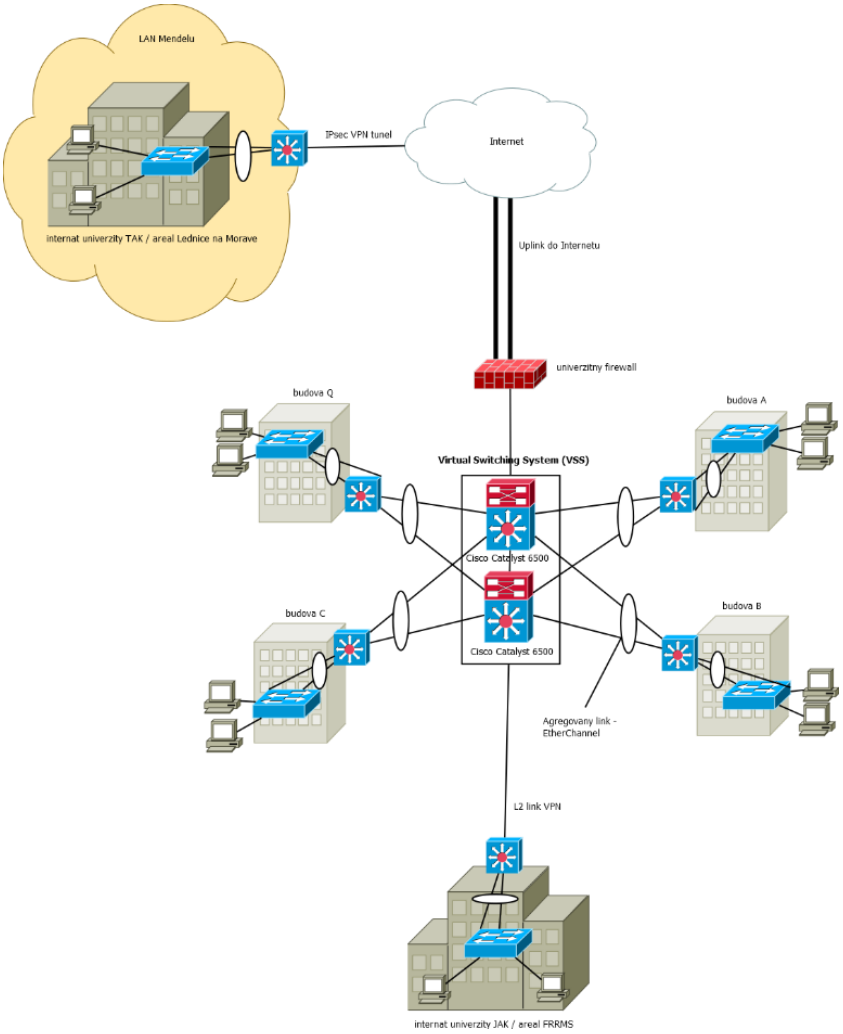
Dokopy všetky lokality v sebe zahrňujú okolo 10000 užívateľov. Na nasledujúcom obrázku Obr. 4 je vyobrazený hlavný areál Černá pole (vrchná časť obrázku) spolu s areálom akadémie FRRMS (budova s písmenom Z). Internáty sa samozrejme nachádzajú mimo hlavnú časť Mendelovej univerzity v Brne.

Obr. 4 Plán areálu MENDELU vrátane FRRMS
Zdroj: http://ipm.af.mendelu.cz/

Hlavná štruktúra siete pozostáva z dvoch CORE prvkov, a to presne Cisco Catalyst 6500, ktoré sú zapojené v tandeme a tvoria jeden virtuálny prepínač. Toto zariadenie sa nachádza, podľa obrázku vyššie, v budove s písmenom X. Z tohto virtuálneho prepínaču takmer vždy vedú dva optické linky spojené v EtherChan-

nel-u a smerujú na distribučné prvky v jednotlivých budovách. Z nich sa dostávame na následné prístupové prepínače, pod ktorými si je možné predstaviť zariadenia Cisco Catalyst 2960 48-portový alebo 28-portový prepínač. K prístupovým prepínačom už vedú potom koncové zariadenia ako sú stolné počítače, notebooky, tlačiarne, telefóny, kamery a podobne. Práve spomínaná prístupová časť je riešená v tejto diplomovej práci.

Ak sa pozrie na ďalšie časti univerzity, tým sú spomínané internáty vo vzdialených lokalitách, topológia je tu podobná. Nie je samozrejme v takom veľkom rozsahu a nenachádza sa tu CORE vrstva, ale prístupová a distribučná, z ktorej vedie linka priamo do areálu Černá pole. Tento prepoj, medzi hlavným areálom a internátmi je riešený buď technológiou IPsec VPN tunelom v prípade TAK či Lednice na Moravě, alebo priamou L2 linkou VPN v prípade JAK, alebo taktiež k areálu FRRMS, ktoré nie sú ešte priveľmi vzdialené od Černých polí.
Čo sa týka prepojenia univerzitnej siete so sieťou Internet, tak z časti CORE sa ide cez univerzitný firewall a následné pripojenie zabezpečuje spoločnosť CESNET



Obr. 5 Nákres súčasnej siete MENDELU

pomocou dvoch 10 Gigabitových uplink-ov. Jeden spoj je vždy aktívny a ide cez neho všetka prevádzka a druhý slúži ako záložný spoj, v prípade výpadku hlavného.

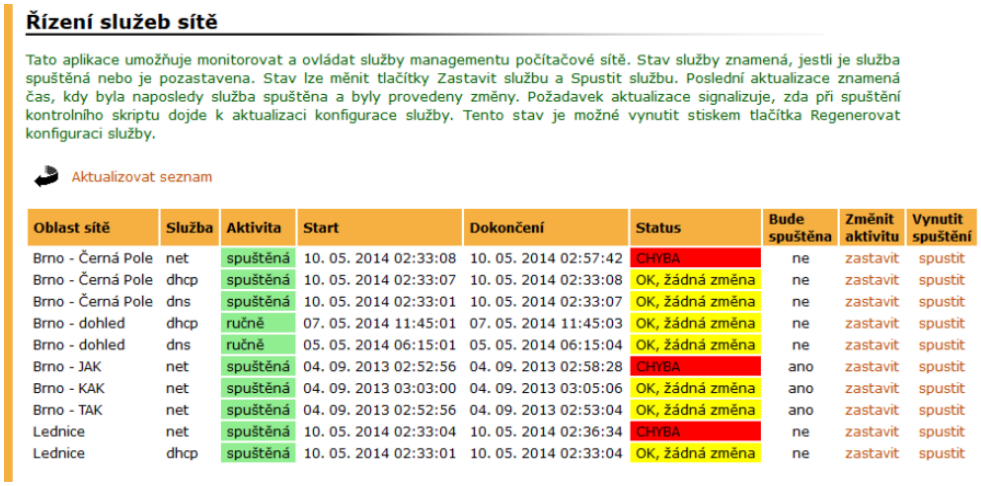
Diplomová práca spracováva a pracuje taktiež s DNS (Domain Name System) a DHCP (Dynamic Host Configuration Protocol) servermi, preto je dôležité ich spomenúť pri analýze súčasného prostredia. Obidva servery sú spustené na počítačoch s operačným systémom Linux. DHCP server sa používa od ISC (Internet Software Consortium), ktorý je pre Linux najbežnejší. Pre DNS sa používa taktiež open source software s názvom BIND (Berkeley Internet Name Domain).

5.2 Aktuálny systém správy siete MENDELU

V neposlednej rade je potrebné opísať i súčasný systém pre manažment siete. Systém je dostupný užívateľom s rôznym typom oprávnení. Systém zahŕňa v sebe aplikáciu na všetky povolené domény DNS v rámci siete Mendelovej univerzity, spracováva širokú radu číselníkov, od rôznych typov zariadení v sieti, spojov, licencie software-ov pre zariadenia, sieťové služby, protokoly, aplikácie a iné. Dôležitou a významnou časťou súčasného systému je evidencia zariadení v sieti, ktorá umožňuje vytváranie, editáciu, vyhľadávanie, či kontrolu zodpovedných užívateľov za zariadenia. [25]

Systém obsahuje v sebe aplikáciu na riadenie služieb siete, ktorá umožňuje ovládať služby manažmentu počítačovej siete. Jedná sa o služby DNS a DHCP pre verziu IPv4 adries, služba net, ktorá má zaisťovať automatickú konfiguráciu. Pod konfiguráciou sa myslí zmena na prepínači a to konkrétne [25]:

- Nastavenie *mode access* na rozhranie.
- Zmena požadovanej VLAN na rozhraní.
- Zmena *port-security* a priradenie MAC adresy.



Řízení služeb sítě

Tato aplikace umožňuje monitorovat a ovládat služby managementu počítačové sítě. Stav služby znamená, jestli je služba spuštěná nebo je pozastavena. Stav lze měnit tlačítky Zastavit službu a Spustit službu. Poslední aktualizace znamená čas, kdy byla naposledy služba spuštěna a byly provedeny změny. Požadavek aktualizace signalizuje, zda při spuštění kontrolního skriptu dojde k aktualizaci konfigurace služby. Tento stav je možné vynutit stiskem tlačítka Regenerovat konfiguraci služby.

Aktualizovat seznam

Oblast sítě	Služba	Aktivita	Start	Dokončení	Status	Bude spuštěna	Změnit aktivitu	Vynutit spuštění
Brno - Černá Pole	net	spuštěná	10. 05. 2014 02:33:08	10. 05. 2014 02:57:42	CHYBA	ne	zastavit	spustit
Brno - Černá Pole	dhcp	spuštěná	10. 05. 2014 02:33:07	10. 05. 2014 02:33:08	OK, žádná změna	ne	zastavit	spustit
Brno - Černá Pole	dns	spuštěná	10. 05. 2014 02:33:01	10. 05. 2014 02:33:07	OK, žádná změna	ne	zastavit	spustit
Brno - dohled	dhcp	ručně	07. 05. 2014 11:45:01	07. 05. 2014 11:45:03	OK, žádná změna	ne	zastavit	spustit
Brno - dohled	dns	ručně	05. 05. 2014 06:15:01	05. 05. 2014 06:15:04	OK, žádná změna	ne	zastavit	spustit
Brno - JAK	net	spuštěná	04. 09. 2013 02:52:56	04. 09. 2013 02:58:28	CHYBA	ano	zastavit	spustit
Brno - KAK	net	spuštěná	04. 09. 2013 03:03:00	04. 09. 2013 03:05:06	OK, žádná změna	ano	zastavit	spustit
Brno - TAK	net	spuštěná	04. 09. 2013 02:52:56	04. 09. 2013 02:53:04	OK, žádná změna	ano	zastavit	spustit
Lednice	net	spuštěná	10. 05. 2014 02:33:04	10. 05. 2014 02:36:34	CHYBA	ne	zastavit	spustit
Lednice	dhcp	spuštěná	10. 05. 2014 02:33:01	10. 05. 2014 02:33:04	OK, žádná změna	ne	zastavit	spustit

Obr. 6 Riadenie služieb siete v súčasnom systéme
Zdroj: Dokumentácia UIS – zväzok 21, 2016.

Chybou súčasného systému je, že nemá vôbec podporu IPv6 adries a s tým spojené služby DNS a DHCPv6. Univerzita v blízkej budúcnosti chce prejsť na adresáciu sieťového priestoru pomocou IPv6 adries a práve toto bol hlavný impulz zadania pre vytvorenie nového systému.

6 Publikačná rešerš

Dneska existuje značný počet systémov, ktoré využívajú mnohí poskytovatelia internetu pre správu a evidenciu zákazníkov a zariadení. Niektorí z nich majú vlastné, iní používajú komerčné systémy, zakúpené od firiem alebo je tu možnosť open source systémov, alebo inak povedané systémy s otvorenou licenciou. Všetky tieto systémy majú za úlohu uľahčiť prácu správcom sietí.

Stanovené kritéria
Kritériá projektu sú dané zákazníkom, v tomto zmysle administrátormi siete MENDELU, ktorí budú priamo pracovať a využívať všetky funkcie tohto systému. Tieto požiadavky sú vypísané v kapitole špecifikácia požiadavkou.

Do projektu je taktiež potrebné zahrnúť záverečné práce študentov vysokých škôl, či niekto nepracoval na podobnom projekte. Publikácie sa hľadali podľa kľúčových slov súvisiacich s danou problematikou. Vyhľadávanie prebiehalo na portáloch theses.cz, www.vutbr.cz, portal.zcu.cz, dk.upce.cz/, is.muni.cz/thesis/ a cez vyhľadávač google.cz.
Všetky vyhľadávané zdroje, aby sa mohli považovať za stále platné a aktuálne, nemohli byť staršie ako 5 až 6 rokov.

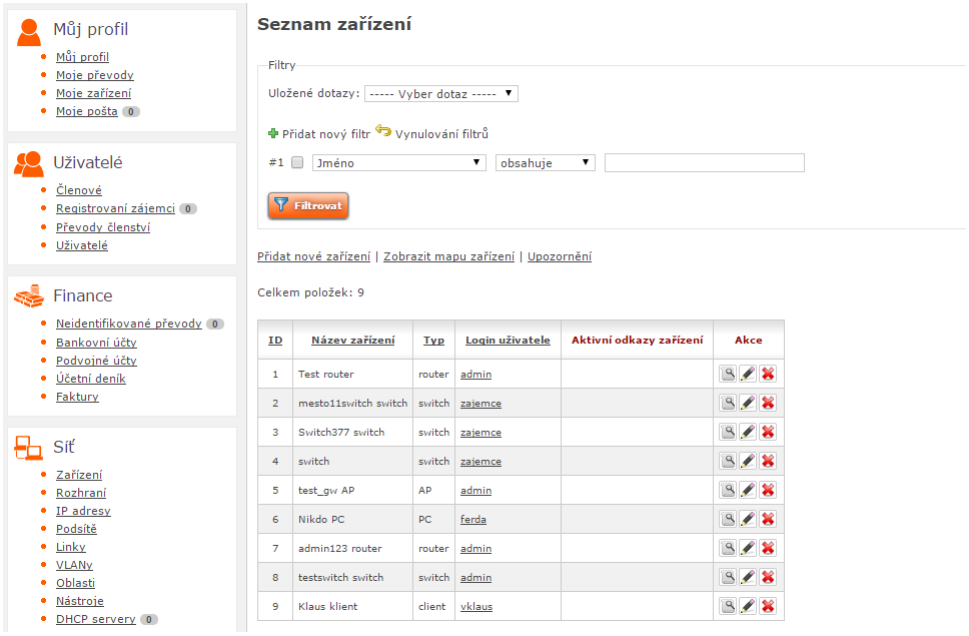
Kľúčové slova: systém pro správu sítě, IPv6 systémy, systém síťové konfigurace, generování konfigurace, automatická konfigurace.

6.1 Open source systémy

6.1.1 FreeNetIS

Informačný systém, ktorý sa zameriava na neziskové organizácie. Dokáže evidovať užívateľov a zariadenia, ktoré k danému užívateľovi patria. Ďalej eviduje umiestnenie a IP adresy jednotlivých zariadení. Má v sebe zabudovaný monitoring v podobe nástroja fping (daemon bežiaci na pozadí linuxového stroja), ktorý upozorňuje na dostupnosť zariadení či DHCP serverov. Tento systém je napísaný v jazyku PHP a používa databázu MySQL. [26]

Oproti požiadavkám na systém, navrhovaný pre MENDELU, FreeNetIS nepodporuje adresy IPv6 a taktiež nedokáže automaticky konfigurovať zariadenia. Administrátor musí všetky IP adresy pre zariadenia, VLAN-y vypĺňať ručne. Neexistuje automatický generátor, ktorý by upozorňoval na obsadené alebo voľné IP adresy, ktoré by automaticky dopĺňal. Ukážka systému je na Obr. 7.

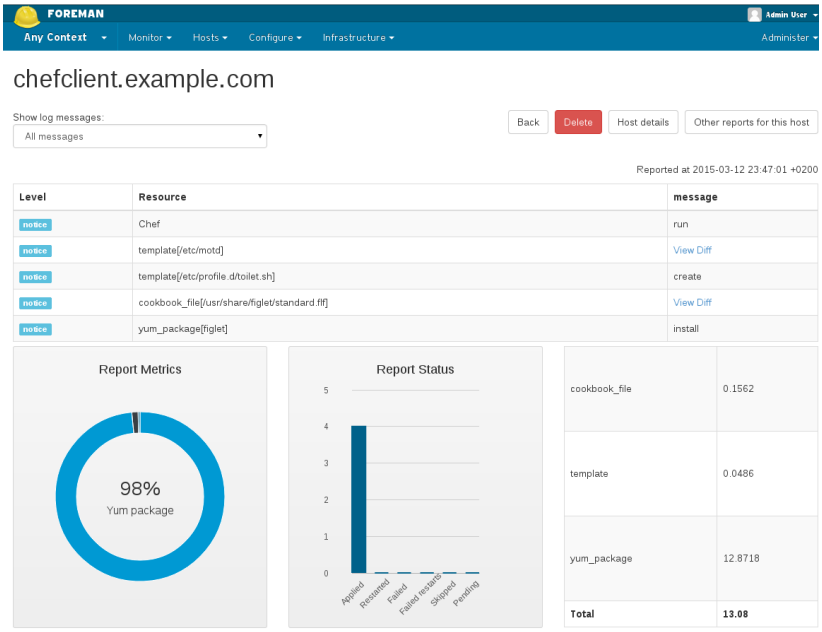


Obr. 7 Ukážka systému FreeNetIS

6.1.2 TheForeman

Jedná sa o ďalší open source projekt, ktorý pomáha administrátorom so správou siete, serverov a iných zariadení. Používa značné množstvo pluginov napísaných v jazyku Ruby. Tieto pluginy sa pripájajú do systému pomocou Smart-Proxy, čo je funkcia umožňujúca poskytnúť API rozhranie na zladenie nástrojov pre systém a poskytnúť tak spôsob pre pridávanie ďalších subsystémov. V súčasnosti podporuje DHCP, DNS či TFTP servery. Pre prístup do systému používa webové rozhranie. [27]

TheForeman je veľmi rozsiahly systém, ktorý podporuje značné množstvo funkcií. Na druhú stranu, ani tento systém nepodporuje IPv6 adresy. So zariadeniami dokáže komunikovať, ale nevie automaticky generovať konfigurácie pre prepínače (switch). Neuchováva taktiež žiadne evidencie užívateľov alebo zariadení, ktoré k nim náležia. Ukážka systému je na Obr. 8.

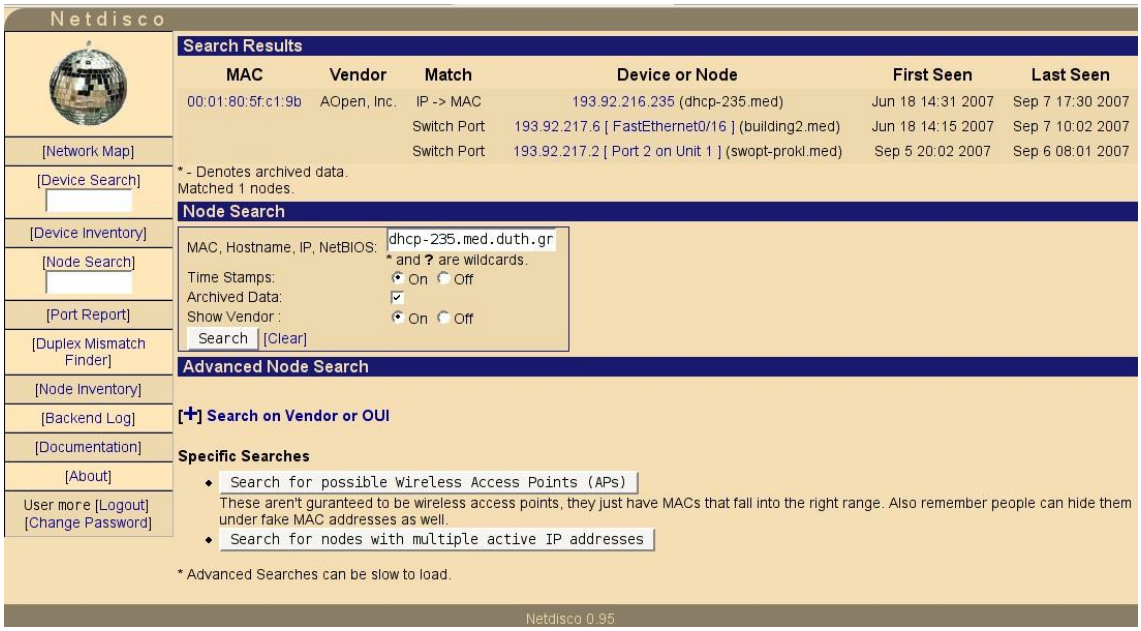


Obr. 8 Ukážka systému TheForeman
Zdroj: http://theforeman.org/

6.1.3 Netdisco

Netdisco je systém, ktorý sa začal vyvíjať v roku 2003. Systém je voľne dostupný aj s jeho zdrojovým kódom, takže si ho užívateľ môže aj individuálne prispôsobiť podľa svojich potrieb. Tento nástroj sa zameral na správu siete veľkých firiem a univerzít. Dáta zo siete sú zbierané prostredníctvom SNMP. Prvotné verzie sa teda ako už napovedá protokol, ktorý systém používa, zamerali na monitoring siete a lokalizáciu počítača v sieti a určenie portu prepínača, ku ktorému je pripojený. To funguje vďaka mapovaniu MAC adries na IP adresy, ktoré si cez SNMP načíta z ARP tabuliek z router-ov a MAC tabuliek zo switch-ov. Takže v tomto prípade nemusí užívateľ pristupovať do zariadenia cez konzolu, aby sa tieto informácie dozvedel. Ako sa postupne systém vyvíjal, tak bol postupne doplnený aj o inventarizáciu sieťového hardwaru, kde si užívateľ môže do systému vložiť ľubovoľný prvok a definovať mu základné informácie napr. názov, výrobcu, model, operačný systém, atď. Tieto zhromaždené údaje sú ukladané do databázy PostgreSQL. [28]

Netdisco má pomerne zaujímavé a rozsiahle využitie, avšak nepodporuje autokonfiguráciu. Takže je to nástroj, ktorý sa ideálne hodí na monitoring, nakoľko využíva spomínaný SNMP protokol. Výhodou je aj jeho inventarizácia sieťových prvkov, no tiež obmedzená len na základné údaje zariadenia. Ukážka systému je znázornená na obrázku Obr. 9.



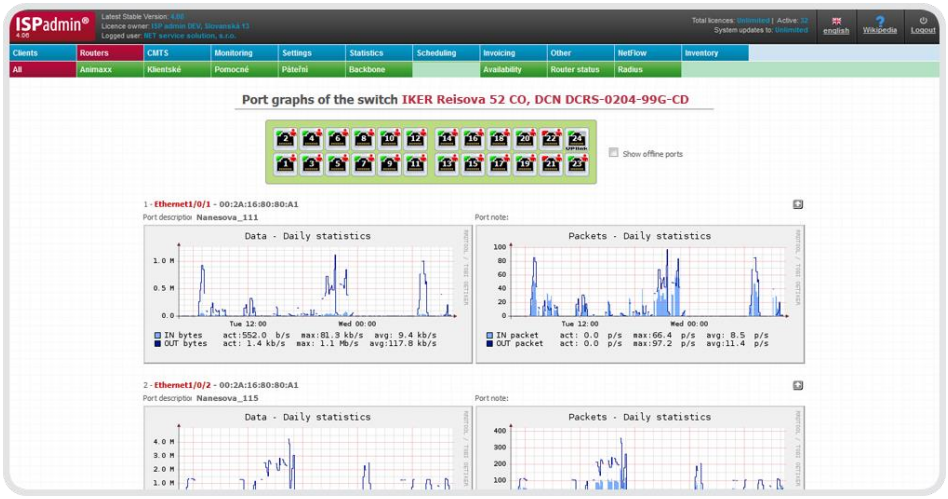
Obr. 9 Ukážka systému Netdisco
Zdroj: https://commons.wikimedia.org

6.2 Komerčné systémy

6.2.1 ISP Admin

Jeden z komerčných systémov, ktorý sa zameriava hlavne na poskytovateľov internetu. Má integrované funkcie pre fakturácie, plánovanie, klientsky portál a iné, pre lepšiu správu a prácu providerov. V súčasnosti ISP Admin podporuje smerovače s operačným systémom MikroTik, Linux, Ubiquity, Motorola Canopy, pre ktoré má v sebe zabudované prívetivé grafické prostredie pre ich správu. Taktiež by mal mať podporu IPv6 adries. Na monitoring siete využíva systém NAGIOS, ktorý monitoruje nielen jednotlivé zariadenia v sieti, ale sleduje aj služby a procesy, ktoré bežia na týchto zariadeniach. [29]

Oproti navrhovanému systému v diplomovej práci, tento nemá podporu pre typy zariadení od firmy Cisco, čo nevýhoda. Neeviduje zapojenie jednotlivých fyzických portov a administrátor nemá tak prehľad o topológii sieti. Neumožňuje porovnávanie ani zálohovanie konfigurácií prvkov, nedokáže tiež generovať tieto konfigurácie a zasielať do zariadení. Obrázok Obr. 10 znázorňuje ukážku systému ISP admin.

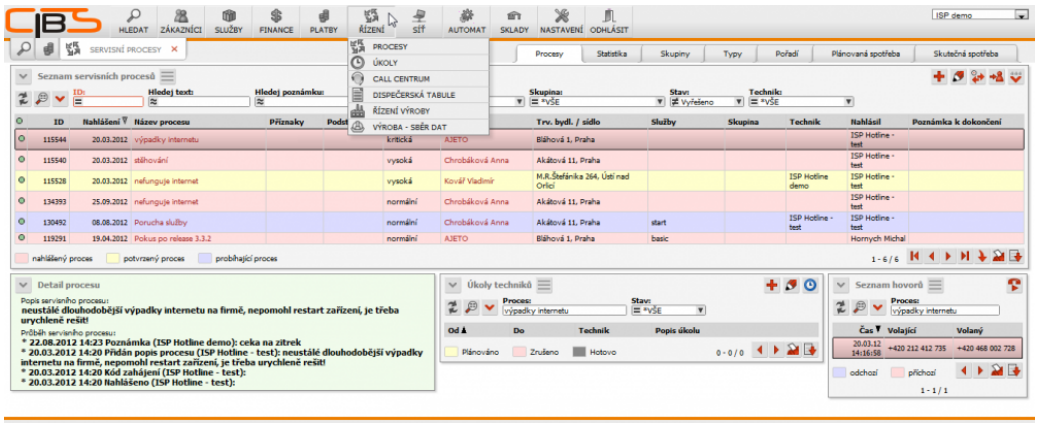


Obr. 10 Ukážka systému ISP admin
Zdroj: http://www.ispadmin.eu/

6.2.2 CIBS

CIBS je taktiež systém zo skupiny pre poskytovateľov internetu. No na rozdiel od vyššie spomínaného ISP Admin, má podporu pre správu prepínačov či smerovačov od rôznych firiem. Má veľmi veľký manažment účtovacích služieb, generovanie a archivácie dokumentov. Umožňuje topológie siete jednotlivých zariadení a kompletný monitoring a ako jeden z mála aj automatickú konfiguráciu zariadení. Je to veľmi rozsiahly systém určený pre veľké siete a obsahuje veľké množstvo modulov. [30]

Jedná sa o komerčný systém a jeho kód nie je voľne dostupný. Nakoľko je veľmi rozsiahly, nie je užívateľsky jednoduchý a ľahko ovládateľný. Patrí taktiež k systémom, ktoré nepodporujú IPv6 adresy. Ukážka systému je na obrázku Obr. 11.



Obr. 11 Ukážka systému CIBS
Zdroj: https://is.cibs.cz/

6.3 Záverečné práce študentov

Vilém Šlesinger (2014)
Práca s názvom Informačný systém pre správcov siete sa zameriava hlavne na evidencie zariadení, komponentov, licencie alebo multilicencie zariadení, správa užívateľov, miestností a iné. Je to skorej komplexný systém s vytvorenou relačnou databázou pre vedenie agendy ako systém pre správu celej siete, konfigurácie a monitoringu. [31]

Jiří Leskovec (2013)
Táto práca spracováva návrh a implementáciu webovej aplikácie pre konfiguráciu a kontrolu zariadení. Aplikácia je schopná spracovať výstup zo zariadení od firmy Cisco, urobiť užívateľskú kontrolu a zobraziť výstup. Pôvodne autor zamýšľal i správu zariadení, ale táto myšlienka ostala neimplementovaná. [31]

Ján Doleček (2010)
Jedná sa o komplexný systém vytváraný pre správu a evidenciu zákazníkov a sieťových zariadení pre poskytovateľov sieťových pripojení. Systém eviduje stav a počet jednotlivých sieťových zariadení. Ako jeden z mála umožňuje základnú autokonfiguráciu zariadení, ale dokáže automatizovať len prvotné nastavenia nových prvkov. Nedokáže komunikovať so servermi DHCP alebo DNS. Systém je napísaný v jazyku PHP a používa relačnú databázu MySQL. Taktiež nepodporuje IPv6 adresy. [32]

Petr Hromádko (2009)
Diplomová práca sa zaoberá návrhom webového informačného systému pre správu klientov v bezdrôtovej sieti. Jedná sa síce len o bezdrôtovú sieť, ale táto práca je zaujímavá práve tým, že využíva zariadenia od firmy MIKROTIK. Informačný systém poskytuje trvalý prehľad o všetkých klientoch siete a zároveň umožňuje nastavovať parametre spomínaných sieťových prvkov MIKROTIK. Systém zahŕňa i jednoduchý monitoring v podobe zisťovania množstva prenesených dát alebo kontrola stavu signálu.

Práca naopak neobsahuje podporu IPv6 adries, alebo evidenciu jednotlivých zariadení, prípadne užívateľov. Negeneruje žiadne konfigurácie pre DHCP alebo DNS servery. Nevýhodou, v zmysle našich požiadaviek, je taktiež, že nepracuje so zariadeniami firmy CISCO a systém sa nezameriava na celú sieť, ale len na bezdrôtovú. [33]

Roman Tomášek (2013)
Autor rieši návrh informačného systému, ktorý slúži hlavne k zhromažďovaní informácií o zariadeniach pripojených do počítačovej siete. Tieto dáta sú získané pomocou protokolu SNMP, ktoré sú ukladané do MySQL databázy. Systém pojednáva možnosť dohľadania zariadení a port na ktorom je pripojený na základe MAC adries, alebo dokonca IPv6 adries. V tomto smere ale využiteľnosť systému

končí. Je zaujímavé, že pracuje práve aj s IPv6 adresami, no systém ako bolo spomínané, slúži viac na zhromažďovanie informácií ako nejaké konfigurácie či nastavenia, prípadne evidencie. [33]

6.4 Porovnanie funkcionalít vybraných systémov

Tab. 4 Porovnanie funkcionalít existujúcich systémov

	Názov produktu				
Funkcia - požiadavka	OpenSource systémy			Komerčné systémy	
	FreeNet IS	The Foreman	Netdisko	ISP Admin	CIBS
podpora IPv6				áno	
komunikácia so sieťovými prvkami		áno	áno	áno (obmedzene)	áno
generovanie konfigurácií pre sieťové prvky					áno
zálohovanie konfigurácií sieťových prvkov					
generovanie konfigurácií pre službu DHCP		áno			
generovanie konfigurácií pre službu DNS		áno			
monitoring siete	áno	áno	áno	áno	áno
evidencia užívateľov	áno				áno
evidencia sieťových zariadení	áno		áno	áno	
GUI sieťových prvkov (interface-y, zapojenia)				áno	
GUI pre správu IS	áno	áno	áno	áno	áno

Tab. 5 Porovnanie funkcionalít systémov zo záverečných prací študentov

	Záverečné práce				
Funkcia - požiadavka	Vilém Šlesinger (2014)	Jiří Leskovec (2013)	Ján Doleček (2010)	Petr Hromádko (2009)	Roman Tomášek (2013)
podpora IPv6					áno
komunikácia so sieťovými prvkami		áno	áno	áno	áno
generovanie konfigurácií pre sieťové prvky		áno	áno (prvotné nastavenia		

zálohovanie konfigurácií sieťových prvkov					
generovanie konfigurácií pre službu DHCP					
generovanie konfigurácií pre službu DNS					
monitoring siete		áno		áno	áno
evidencia užívateľov	áno		áno	áno	
evidencia sieťových zariadení	áno		áno	áno	
GUI sieťových prvkov (interface-y, zapojenia)					
GUI pre správu IS	áno	áno		áno	áno

6.5 Záver publikačnej rešerše

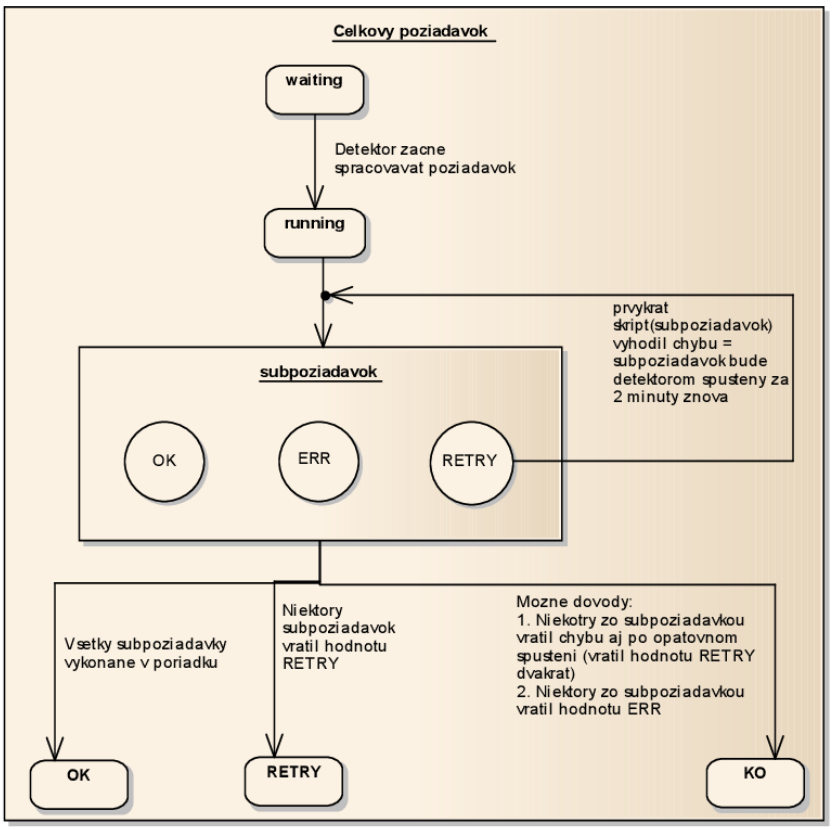
Väčšina systémov, ktoré sú opísané vyššie, sa zameriava na systémy pre poskytovateľov internetových pripojení. Majú obsiahle evidencie užívateľov, účtovacie a fakturačné doplnky a máloktoré komunikujú priamo so zariadeniami pre potrebnú konfiguráciu a s tým spojené i zálohovanie konfigurácií. Niektoré z nich nepodporujú sieťové prvky od firmy ako je CISCO, ktoré sa v sieti MENDELU nachádzajú. Čo je ale tiež veľmi dôležité spomenúť je fakt, že systémy, okrem jedného, ktoré boli vyhľadané a spĺňali aspoň niektoré zo zadaných požiadaviek na vyvíjaný systém pre MENDELU, nepodporujú IPv6 adresy. Práve podpora IPv6 adries je hlavným dôvodom vývoja nového systému. Na tabuľke porovnania funkcionalít jednotlivých systémov je možné vidieť, že iba ISP Admin podporuje prácu s IPv6. Bohužiaľ, tento systém nie je open source a ku zdrojovému kódu sa nedá dostať. V tomto zmysle je taktiež pre školu nepoužiteľný.

Podobný problém nastal i pri prehľadávaní záverečných prací. Všetky sa zameriavali na niektoré časti nášho vyvíjaného systému, či kvalitná evidencia, alebo základná konfigurácia, ale nenašiel sa systém, ktorý by dokázal všetko v jednom a naviac pracovať s IPv6 adresami. Nehovoriac o ďalších, špecifických požiadavkách pre konkrétnu sieť MENDELU.

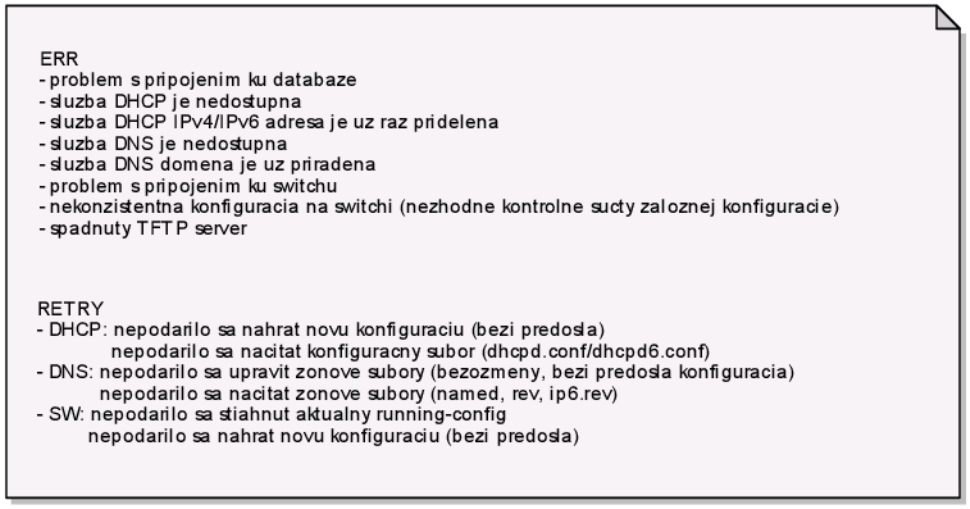
7 Návrh nového riešenia

Táto časť práce opisuje diagramy návrhu nového systému z backend-ového pohľadu. Jedná sa o diagramy, ktoré sa snažia zachytiť základný návrh databáze, ktorá bola dielom spolu so študentom Oliverom Horečným, nakoľko sa jedná o časť systému, kde sa stretávajú naše diplomové práce. Ďalej sú zahrnuté vývojové diagramy jednotlivých skriptov, ktoré opisujú funkčnosť a uľahčujú následnú implementáciu. Konkrétne sa jedná o skripty *detektor.pl, dhcp.pl, dns.pl a sw.pl*.

Pre pochopenie následných diagramov, scenárov či tabuliek, je potrebné stanoviť terminológiu požiadavok a subpožiadavok. To, čo zadáva administrátor do systému a nachádza sa v tabuľke *change*, sa označuje ako celkový **požiadavok**. Ten v sebe podľa jeho typu obsahuje príslušné **subpožiadavky**, ktoré majú byť spustené, respektíve skripty zápis do DHCP, zápis do DNS, konfigurácia SW. Stavy, ktoré môžu nadobúdať požiadavky alebo subpožiadavky, opisuje nasledujúci diagram (Obr. 12) ich životný cyklus, respektíve jednotlivé prechody. K diagramu náleží i doplňujúca poznámka (Obr. 13), ktorá taktiež obsahuje v sebe príklady, kedy jednotlivé stavy subpožiadavkou môžu nastať.



Obr. 12 Alternatívne stavy požiadavkou a subpožiadavkou



Obr. 13 Príklady pre možné stavy subpožiadavkou

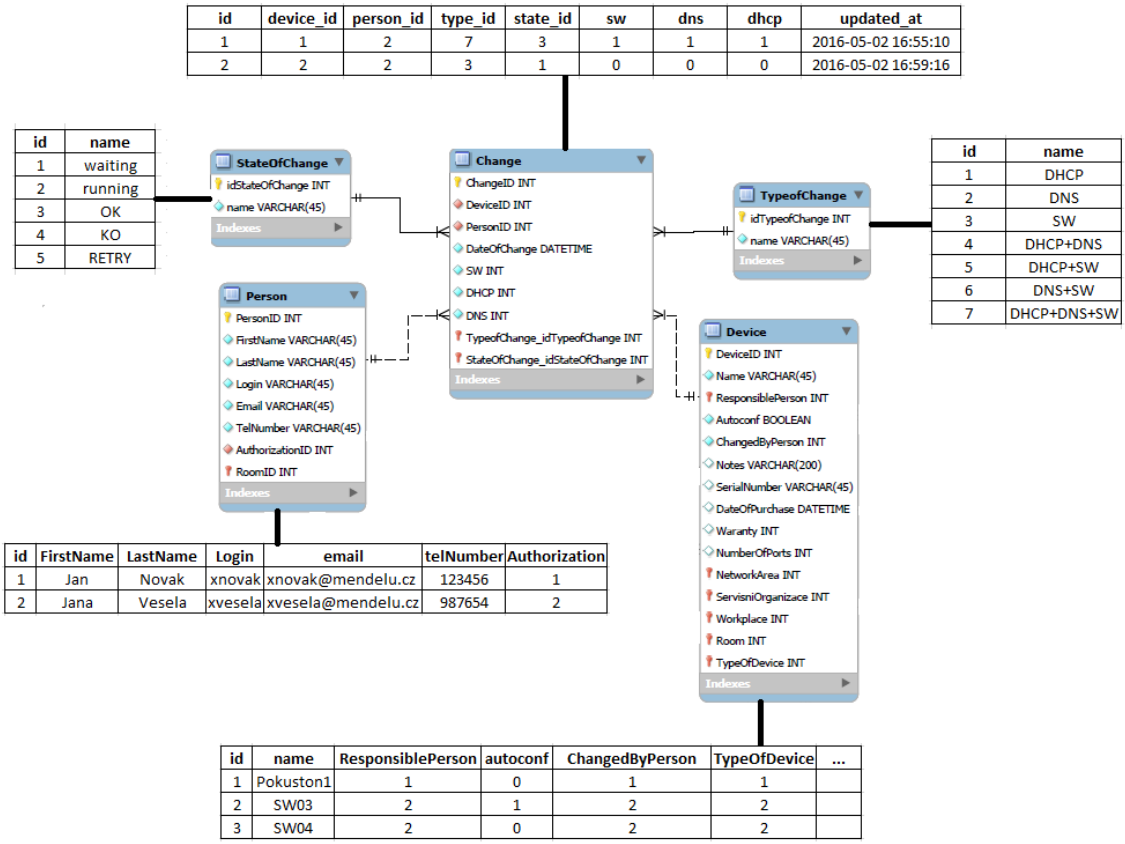
7.1 Entity-relationship diagram (ERD)

Obrázok Obr. 14 znázorňuje výrez diagramu ERD. Zobrazuje hlavnú časť, ktorá je podstatná práve pre backend systému. Celý diagram sa nachádza v kapitole prílohy. [Príloha A]

Hlavný skript *detektor.pl* komunikuje s tabuľkou *change*. V tej sú vždy uložené a aktualizované požiadavky, ktoré sa majú v systéme previesť. Stĺpce *device_id, person_id, type_id, state_id* sú priamo prepojené s ďalšími tabuľkami v databáze podľa obrázku.
Stĺpce *sw, dns, dhcp* ukladajú hodnoty podľa výsledku jednotlivých subpožiadavkou, respektíve skriptov, ktoré spúšťa detektor. Môžu obsahovať hodnoty:

- 0 = žiadna akcia nebola prevedená
- 1 = subpožiadavok skončil v poriadku (OK)
- 2 = subpožiadavok vrátil chybu (RETRY)
- 3 = subpožiadavok vrátil ERROR

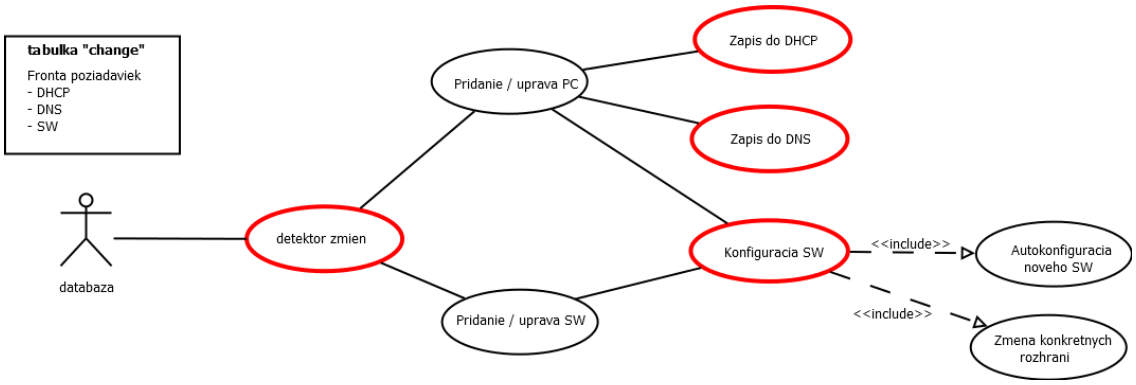
Situácie a rozdiel medzi hodnotou 2 a 3 sú taktiež vyobrazené v obrázku Obr. 13 vyššie. Chyba ERROR je spojená so službami, či pripojením ku zariadeniam a kedy hlavnému skriptu *detektor.pl* je vynútené ukončenie, nakoľko nemá zmysel spúšťať ďalšie požiadavky, keď nefunguje niektorá z hlavných služieb a je potrebný zásah administrátora. Chyba RETRY je spojená s konfiguráciami, v neúspešnom prípade sú nahraté záložné konfigurácie a subpožiadavok je opätovne skúšaný/spustený, nie je nutné ukončenie detektoru. Práve podľa hodnôt subpožiadavkou a typu konkrétneho požiadavku *type_id* detektor vyhodnotí celý požiadavok, t.j. *state_id*. Ak sa teda jedná o *type_id* s hodnotou 7 (viď. Obr. 14), detektor kontroluje, ako dopadli všetky tri subpožiadavky. Ak je *type_id* s hodnotou 3, zisťuje sa hodnota iba subpožiadavku *sw*.



Obr. 14 Výrez diagramu ERD

7.2 USE CASE model

Nasledujúci obrázok Obr. 15 zobrazuje USE CASE model backend-ovej časti navrhovaného systému. Červeno vyznačené prípady užitia, nižšie i opísané v jednotlivých scenároch, sú hlavnými časťami zadania.



Obr. 15 USE CASE model backend-ovej časti navrhovaného systému

Scenár pre USE CASE detektor zmien

Detektor zmien, je hlavný skript, mohlo by sa povedať tzv. manažér, ktorý rozdáva prácu podriadeným. Kontroluje a prechádza tabuľku *change* v databáze, či sa vo fronte nachádza nejaká požiadavka. Ak áno, tak podľa typu požiadavky spúšťa subpožiadavky, v tomto prípade skripty *dhcp.pl, dns.pl, sw.pl*. Tieto subpožiadavky posielajú ich výsledok naspäť detektoru a ten aktualizuje databázu, v krajnom prípade informuje administrátora o vyskytnutej chybe.

Tab. 6 Scenár pre USE CASE detektor zmien

USE CASE	Detektor zmien
Krátky popis	Čítanie fronty z databázy a následné spustenie príslušných skriptov (subpožiadavkou).
Aktéri	Databáza, skripty (DHCP.pl, DNS.pl, SW.pl).
Podmienky pre spustenie	Fungujúca databáza a prístup k nej. Nastavené opakujúce sa spustenie skriptu *detektor.pl* každé 2 minúty.
Základný scenár	1. Detektor kontroluje, či je už raz spustený 2. Pripojenie do databázy 3. Prezeranie tabuľky *change*, či sa nachádza vo fronte požiadavok so stavom „waiting“ 4. Zmení sa pre daný požiadavok príznak „running“ 5. Podľa „*type_id*“ sa spustia potrebné subpožiadavky 6. Detektor čaká na návratovú hodnotu subpožiadavkou 7. Aktualizácia stavu subpožiadavkou v databáze 8. Vyhodnotenie celkového požiadavku podľa stavov subpožiadavkou v databáze 9. Prechod na bod číslo 3
Alternatívny scenár	1a – Detektor zistil, že už je raz spustený 1a.1 – Notifikácia administrátorovi prostredníctvom email-u 1a.2 – Ukončenie skriptu 2a – Detektor sa nemôže pripojiť ku databáze 2a.1 – Notifikácia administrátorovi prostredníctvom email-u 2a.2 – Ukončenie skriptu 3a – V tabuľke *change* sa vo fronte nachádza požiadavok so stavom „retry“ 3a.1 – Detektor podľa stavov subpožiadavkou v DB spúšťa opätovne skripty, ktoré skončili prvýkrát neúspešne 3a.2 – Detektor čaká na návratovú hodnotu subpožiadavkou 3a.3 – Aktualizácia stavu subpožiadavkou v databáze 3a.4 – Vyhodnotenie celkového požiadavku podľa stavov subpožiadavkou v databáze 3a.5 – Kontrola, či sa nachádza vo fronte požiadavok so stavom „retry“ 3a.5a – Prechod na bod 3a 3a.5b – Prechod na bod 3 3b – V tabuľke *change* sa nenachádza požiadavok so stavom „waiting“ alebo „retry“ 3b.1 – Ukončenie skriptu 8a – Všetky požadované subpožiadavky vrátili hodnotu OK 8a.1 – Aktualizácia celkového požiadavku vo fronte v databáze „OK“ 8a.2 – Prechod na bod číslo 3 8b – Niektorý zo subpožiadavkou vrátil hodnotu RETRY i po opätovnom spustení 8b.1 – Aktualizácia celkového požiadavku vo fronte v databáze „KO“

	8b.2 – Notifikácia administrátorovi prostredníctvom email-u 8b.3 – Prechod na bod číslo 3 8c – Niektorý zo subpožiadavkou vrátil hodnotu ERROR 8c.1 – Aktualizácia celkového požiadavku vo fronte v databáze „KO“ 8c.2 – Notifikácia administrátorovi prostredníctvom email-u 8c.3 – Ukončenie skriptu
Podmienky ukončenia	V tabuľke *change* sa nenachádza žiadny záznam so stavom „waiting“ alebo „retry“. Detektor nesmie byť spustený dvakrát naraz. Nie je možné sa pripojiť ku databáze. Niektorý zo subpožiadavkov vrátil hodnotu ERROR.

Scenár pre USE CASE zápis do DHCP

Úlohou tohto prípadu užitia je úprava servera DHCPv4/v6. Po vyčítaní potrebných informácií z databáze si otvorí konfiguračné súbory serverov, upraví podľa požiadaviek a následne reštartuje služby. V prípade neúspešného štartu serverov, nahráva záložné konfigurácie. Výsledok odovzdáva detektoru, ktorý aktualizuje databázu.

Tab. 7 Scenár pre USE CASE zápis do DHCP

USE CASE	Zápis do DHCP
Krátky popis	Úprava konfigurácií DHCPv4/v6 servera.
Aktéri	Detektor zmien, databáza, konfiguračné súbory DHCPv4/v6 servera.
Podmienky pre spustenie	Fungujúca databáza a prístup k nej. Fungujúci DHCPv4/v6 server.
Základný scenár	1. Kontrola služieb *dhcpd/dhcpd6* 2. Pripojenie do databázy 3. SQL SELECT potrebných informácií (IPv4 adresa zariadenia, IPv4 adresa siete, MAC adresa zariadenia, IPv6 ID zariadenia, IPv6 adresa siete, DUID zariadenia, názov zariadenia) 4. Vytvorenie zálohy konfiguračných súborov *dhcpd.conf* a *dhcpd6.conf* 5. Čítanie konfiguračných súborov *dhcpd.conf* a *dhcpd6.conf* 6. Úprava *dhcpd.conf/dhcpd6.conf* 7. Reštart služieb dhcpd/dhcpd6 8. Zaslanie návratovej hodnoty detektoru 9. Ukončenie skriptu
Alternatívny scenár	1a – Služba dhcpd/dhcpd6 nie je spustená 1a.1 – Zaslanie návratovej hodnoty detektoru o chybe služby 1a.2 – Ukončenie skriptu 2a – Skript dhcp.pl sa nemôže pripojiť ku databáze 2a.1 – Zaslanie návratovej hodnoty detektoru o chybe služby 2a.2 – Ukončenie skriptu 5a – Chyba pri otváraní konfiguračných súborov *dhcpd.conf/dhcpd6.conf* 5a.1 – Zaslanie návratovej hodnoty detektoru o chybe súborov 5a.2 – Ukončenie skriptu 5b – Požadovaná IP adresa je už pridelená (používaná) 5b.1 – Zaslanie návratovej hodnoty detektoru o chybe IP adresy

	5b.2 – Ukončenie skriptu 8a – Služba dhcpd/dhcpd6 nemôže naštartovať (je spadnutá) 8a.1 – Skript dhcp.pl nahráva záložnú konfiguráciu súborov *dhcpd.conf/dhcpd6.conf* 8a.2 – Reštart služieb dhcpd/dhcpd6 8a.3 – Kontrola služieb, či sú spustená 8a.3a – Zaslanie návratovej hodnoty detektoru „predošlá konfigurácia“ 8a.3b – Ukončenie skriptu 8b.3a – Prechod na krok číslo 1a
Podmienky ukončenia	Konfiguračný súbor *dhcpd.conf/dhcpd6.conf* je úspešne upravený podľa požiadavkou. Konfiguračný súbor nie je upravený, je nahratá predošlá/záložná konfigurácia. Služba dhcpd/dhcpd6 nie je spustená. Problém s konfiguračnými súbormi *dhcpd.conf/dhcpd6.conf*.

Scenár pre USE CASE zápis do DNS

Zápis do DNS je podobný prípad užitia ako zápis do DHCP s tým rozdielom, že komunikuje s DNS serverom. Vyčíta potrebné informácie, mení zónové súbory servera pre príslušnú doménu a reštartuje službu. V neúspešnom prípade nahráva predchádzajúcu, funkčnú konfiguráciu a výsledok opäť posiela detektoru.

Tab. 8 Scenár pre USE CASE zápis do DNS

USE CASE	Zápis do DNS
Krátky popis	Úprava konfigurácií DNS servera.
Aktéri	Detektor zmien, databáza, konfiguračné/zónové súbory DNS servera.
Podmienky pre spustenie	Fungujúca databáza a prístup k nej. Fungujúci DNS server.
Základný scenár	1. Kontrola služby *dns* 2. Pripojenie do databázy 3. SQL SELECT potrebných informácií (IPv4 adresa zariadenia, IPv6 ID zariadenia, IPv6 adresa siete, doménové meno) 4. Čítanie konfiguračných/zónových súborov domény *named/rev/ip6.rev* 5. Vytvorenie zálohy súborov domény *named/rev/ip6.rev* 6. Úprava súborov domény *named/rev/ip6.rev* 7. Reštart služby *dns* 8. Zaslanie návratovej hodnoty detektoru 9. Ukončenie skriptu
Alternatívny scenár	1a – Služba *dns* nie je spustená 1a.1 – Zaslanie návratovej hodnoty detektoru o chybe služby 1a.2 – Ukončenie skriptu 2a – Skript *dns.pl* sa nemôže pripojiť ku databáze 2a.1 – Zaslanie návratovej hodnoty detektoru o chybe služby 2a.2 – Ukončenie skriptu 4a – Chyba pri otváraní konfiguračných súborov domény *named/rev/ip6.rev* 4a.1 – Zaslanie návratovej hodnoty detektoru o chybe súborov 4a.2 – Ukončenie skriptu 4b – Požadované doménové meno je už pridelené (používané)

	4b.1 – Zaslanie návratovej hodnoty detektoru o chybe domény 4b.2 – Ukončenie skriptu 7a – Služba *dns* nemôže naštartovať (je spadnutá) 7a.1 – Skript dns.pl nahráva záložnú konfiguráciu súborov *named/rev/ip6.rev* 7a.2 – Reštart služby *dns* 7a.3 – Kontrola služby, či je spustená 7a.3a – Zaslanie návratovej hodnoty detektoru „predošlá konfigurácia“ 7a.3b – Ukončenie skriptu 7b.3a – Prechod na krok číslo 1a
Podmienky ukončenia	Zónové súbory danej domény *named/rev/ip6.rev* sú úspešne upravené podľa požiadavkou. Zónové súbory nie sú upravené, je nahratá predošlá/záložná konfigurácia. Služba *dns* nie je spustená. Problém so zónovými súbormi danej domény *named/rev/ip6.rev*.

Scenár pre USE CASE konfigurácia SW

V prípade užitia konfigurácia SW, sa môže jednať o konfiguráciu úplne nového prepínača, alebo už iba o konkrétnu zmenu fyzických rozhraní prepínača. Najprv sa teda zistí, o aký typ konfigurácie sa jedná a po vyčítaní potrebných informácií z databázy, sa upravuje aktuálny *running-config* zariadenia. Ten sa najprv stiahne na TFTP server, urobí sa záloha, jeho zmena a je nahratý späť do zariadenia. Ak kontrolné súčty konfigurácií na zariadení a TFTP servery sú rovnaké, kladný výsledok je odoslaný detektoru. V opačnom prípade je opäť nahratá záložná konfigurácia a záporný výsledok je odovzdaný detektoru.

Tab. 9 Scenár pre USE CASE konfigurácia SW

USE CASE	Konfigurácia SW
Krátky popis	Úprava running-config zariadenia SW.
Aktéri	Detektor zmien, zariadenie SW, TFTP server, databáza.
Podmienky pre spustenie	Fungujúca databáza a prístup k nej. Nastavená manažment IP adresa a úspešné pripojenie na cieľové zariadenie SW. Fungujúci TFTP server.
Základný scenár	1. Pripojenie do databázy 2. SQL SELECT potrebných informácií (príznak autokonfigurácie, IP adresa SW, názov rozhraní, VLAN-y pre rozhrania, MAC adresy zariadení) 3. Pripojenie ku zariadeniu SW 4. Kontrola TFTP servera 5. Poslanie príkazu do SW na stiahnutie aktuálneho *running-config* na TFTP server 6. Vytvorenie zálohy stiahnutého *running-config* 7. Úprava stiahnutého *running-config* podľa požiadaviek 8. Poslanie príkazu do SW na nahratie nového *running-config* z TFTP serveru 9. Porovnanie kontrolného súčtu upravenej konfigurácie na SW vs TFTP server

	10. Zaslanie návratovej hodnoty detektoru 11. Ukončenie skriptu
Alternatívny scenár	1a – Skript SW.pl sa nemôže pripojiť ku databáze 1a.1 – Zaslanie návratovej hodnoty detektoru o chybe služby 1a.2 – Ukončenie skriptu 3a – Skript SW.pl sa nemôže pripojiť ku SW 3a.1 – Zaslanie návratovej hodnoty detektoru o chybe pripojenia ku SW 3a.2 – Ukončenie skriptu 4a – Skript SW.pl sa nemôže pripojiť ku TFTP serveru 4a.1 – Zaslanie návratovej hodnoty detektoru o chybe služby 4a.2 – Ukončenie skriptu 5a – Neúspešné stiahnutie *running-config* z SW 5a.1 – Zaslanie návratovej hodnoty detektoru o chybe stiahnutia 5a.2 – Ukončenie skriptu 9a – Nezhodný kontrolný súčet konfigurácií 9a.1 – Poslanie príkazu do SW na nahratie záložného *running-config* z TFTP serveru 9a.2 – Kontrola kontrolných súčtov nahratej konfigurácie na SW vs na TFTP servery 9a.2a – Zaslanie návratovej hodnoty detektoru „predchádzajúca konfigurácia“ 9a.2b – Ukončenie skriptu 9b.2a – Zaslanie návratovej hodnoty detektoru „nekonzistentná konfigurácia“ 9b.2b – Ukončenie skriptu
Podmienky ukončenia	*Running-config* úspešne nahraný na cieľový SW podľa požiadavkou. Nahratý záložný r*unning-config* na cieľový SW (predchádzajúca konfigurácia). Nekonzistentná konfigurácia na cieľovom SW. Problém s pripojením ku SW. Problém s pripojením ku databáze. Problém s pripojením ku TFTP serveru.

7.3 Vývojové diagramy skriptov

Kapitola pojednáva a znázorňuje vývojové diagramy jednotlivých skriptov, ich postupné akcie, čo robia, s kým komunikujú, čo upravujú. Pred tým je ale potrebné zmieniť návratové hodnoty, ktoré skripty vracajú a komunikujú tak s detektorom, ktorý podľa toho aktualizuje databázu. Je to inak povedané vnútorná komunikácia medzi procesmi. Znázorňuje to tabuľka Tab. 10.

Stĺpec návratová hodnota, sú čísla, ktoré posielajú skripty, inak povedané subpožiadavky *dhcp.pl, dns.pl, sw.pl* hlavnému detektoru a vyjadrujú udalosť, ako skončili.

Stĺpec subpožiadavok vyjadruje, ako budú aktualizované subpožiadavky detektorom v databáze, na základe návratovej hodnoty. Jedná sa konkrétne o položky *sw, dns, dhcp* v tabuľke *change* spomínané vyššie pri ERD diagrame.
Stĺpec RETRY vyjadruje farbami, v ktorých prípadoch sú skripty spustené opätovne. Žltá farba znázorňuje, že subpožiadavok skončil s chybou (hodnota 2) a bude spustený znova. Modrá farba znázorňuje, že subpožiadavok skončil s chybou, prípadne opätovne s chybou a nebude už spustený ďalší krát (hodnota 3).

Stĺpec zastav detektor poukazuje, v ktorých prípadoch bude vyslaný signál pre stopnutie chodu detektoru, aby neprechádzal na ďalšie požiadavky vo fronte.

Ako bolo uvedené, je to z dôvodu, ak by boli niektoré zo služieb spadnuté, je zbytočné sa pokúšať o zmenu konfigurácií a je potrebné informovať administrátora.

Legenda

- (hodnota 1) subpožiadavok skončil v poriadku (OK)
- (hodnota 2) subpožiadavok skončil s chybou, opätovné spustenie (RETRY)
- (hodnota 3) subpožiadavok skončil s chybou, nie je spustený ďalší krát (KO)
- (hodnota 3) subpožiadavok skončil s chybou + signál pre zastavenie detektoru (ERR)

Tab. 10 Návratové hodnoty subpožiadavkou

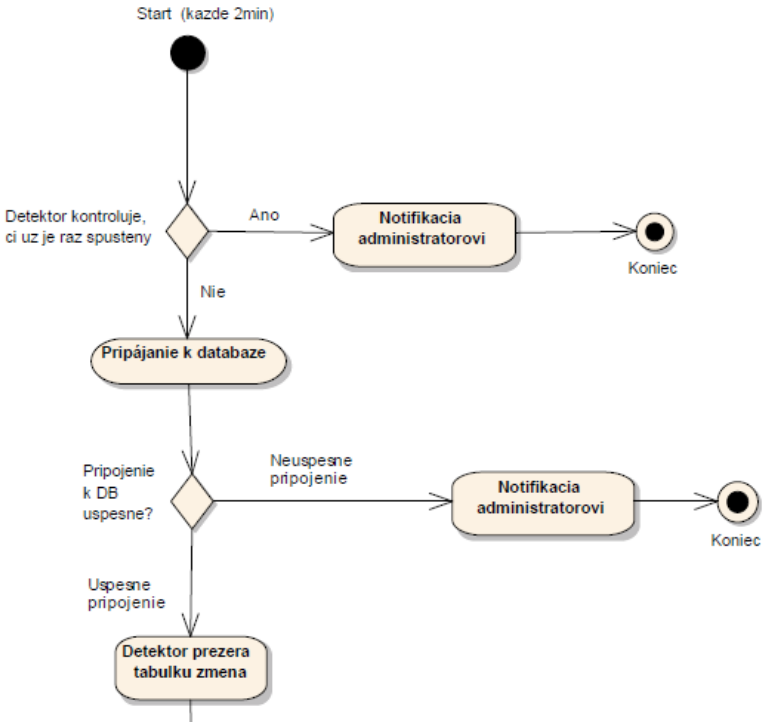
Návratová hodnota	Udalosť	Popis	Subpožiadavok	RETRY	Zastav detektor
1	OK	všetko v poriadku, nová konfigurácia nahratá	1		
2	RETRY -> KO	skript vrátil chybu, nahratá predchádzajúca konfigurácia, skript sa spúšťa znova ak vráti opäť chybu je KO	2/3		
3	DB	chyba pripojenia k databáze	3		
4	DHCPv4	DHCPv4 server neodpovedá (is down)	3		
5	DHCPv6	DHCPv6 server neodpovedá (is down)	3		
6	DHCPv4addr	chyba zápisu novej adresy do DHCPv4 servera, adresa je už priradená	3		
7	DHCPv6addr	chyba zápisu novej adresy do DHCPv6 servera, adresa je už priradená	3		
8	DHCPv4conf	nepodarilo sa načítať konfiguračný súbor *dhcpd.conf*	2/3		
9	DHCPv6conf	nepodarilo sa načítať konfiguračný súbor *dhcpd6.conf*	2/3		
10	DHCPv4RETRY	skript vrátil chybu pri zápise DHCPv4, nahratá predchádzajúca konfigurácia, skript sa spúšťa znova ak vráti opäť chybu je KO	2/3		
11	DHCPv6RETRY	skript vrátil chybu pri zápise DHCPv6, nahratá predchádzajúca konfigurácia, skript sa spúšťa znova ak vráti opäť chybu je KO	2/3		

12	DNS	DNS server neodpovedá (is down)	3		
13	DNSconf	nepodarilo sa načítať konfiguračný súbor DNS servera	2/3		
14	DNSdomain	chyba pri úprave zónových súborov = doména je už priradená	3		
15	SW	chyba pripojenia ku zariadeniu SW	2/3		
16	SWconf	na zariadení SW nie je konzistentná konfigurácia	3		
17	TFTP	server TFTP nedopovedá (is down)	3		
18	timeout	skript nevrátil hodnotu počas 15s	2/3		

7.3.1 Detektor

Nasledujúce obrázky znázorňujú časti vývojového diagramu pre *detektor.pl*. Celý diagram sa nachádza v prílohe diplomovej práce. [Príloha B]
Na obrázku Obr. 16 je možné vidieť prvú časť detektoru. Po automatickom štarte, ktorý je nastavený na každé 2 minúty, sa ako prvé kontroluje či skript *detektor.pl* nie je už raz spustený. Je to z toho dôvodu, aby sa predišlo situácii, ak by sa skript náhodou zasekol na riešení nejakého požiadavku. Nesmie nastať situácia, kedy by jeden požiadavok spracovávali dva skripty, respektíve detektory. Ak by nastalo, že by detektor zistil, že beží proces spusteného detektoru, ihneď uvedomí administrátora.

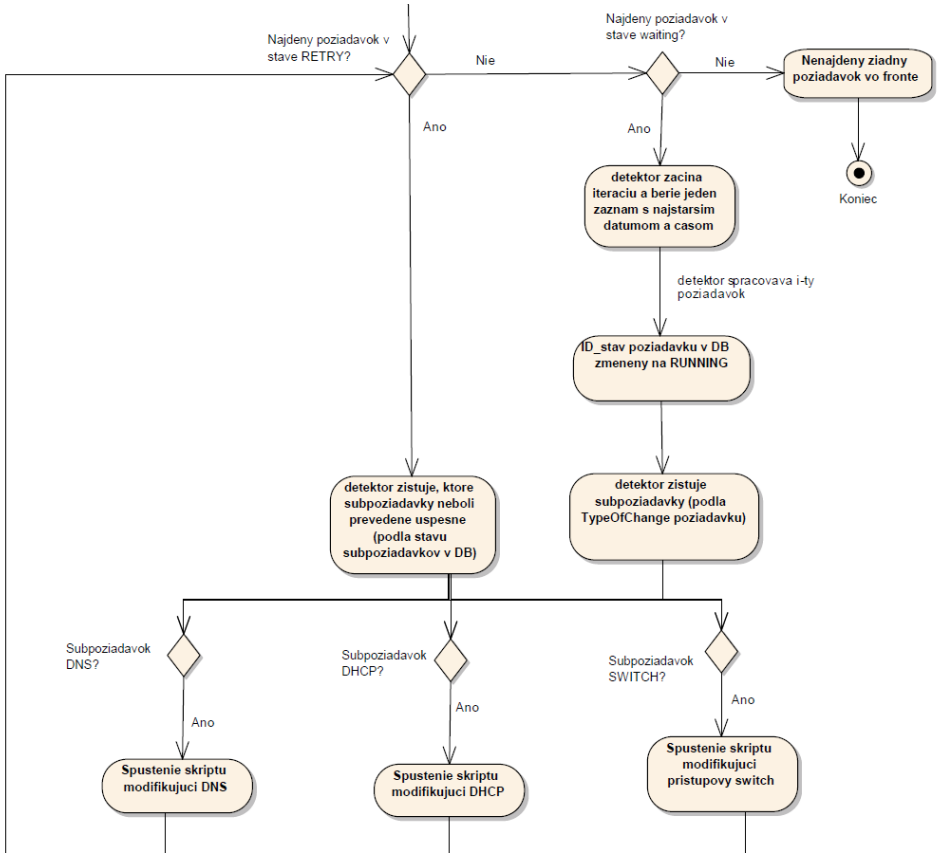
Ak detektor nezistí prítomnosť iného detektoru, pokračuje kontrolou pripojenia databázy. Opäť, pri neúspechu je administrátor informovaný a skript *detektor.pl* ukončený. Naopak, pri úspešnom pripojení ku databáze, nastáva prezeranie tabuľky *change*, či sa v nej nachádzajú nové požiadavky.



Obr. 16 Vývojový diagram *detektor.pl* (časť 1)

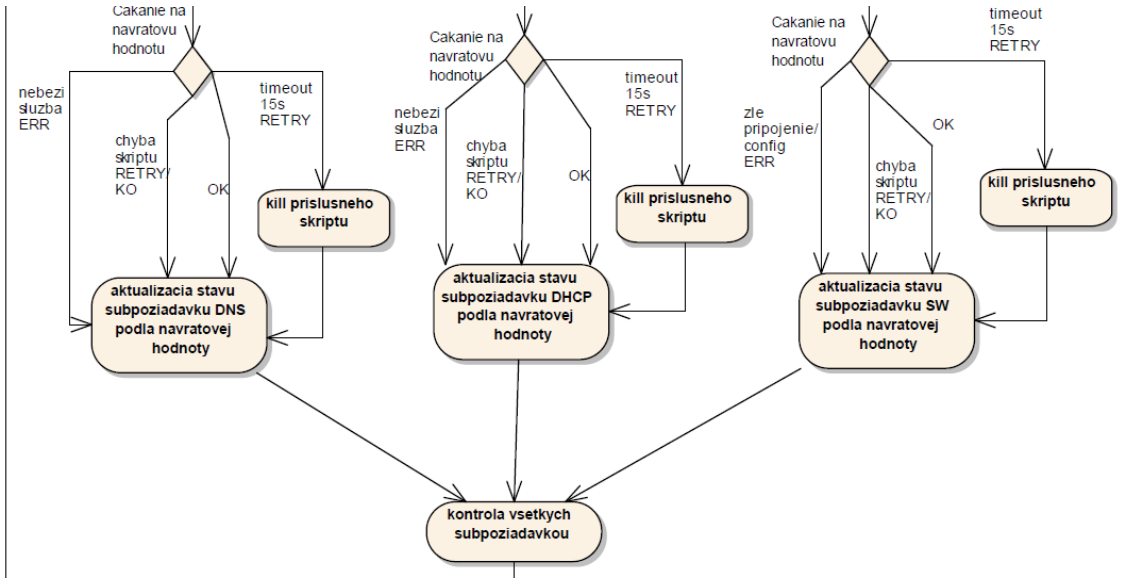
Obrázok Obr. 17 zobrazuje druhú časť detektoru, kde skript prehľadáva tabuľku *change*. Detektor najprv kontroluje, či sa nachádzajú v tabuľke nejaké požiadavky so stavom *retry*. Ak áno, znamená to, že pri spustenom predchádzajúcom detektore niektorý zo subpožiadavkou vrátil chybu a je potrebné ho spustiť znova. Mohlo sa jednať o dočasnú chybu, napríklad malého výpadku siete a preto je potrebné vyskúšať spustiť požiadavok znova. Detektor v tomto prípade zistí, ktorý subpožiadavok vrátil chybu (hodnota 2 - RETRY) a je spustený iba konkrétny skript. Takto sa predíde problému, ak by sa jednalo o požiadavok, kde sa upravuje DHCP server i prepínač a jeden z nich iba skončil s chybou, nebude sa prepisovať a znova spúšťať úspešná úprava. Takto sa prejdú všetky požiadavky so stavom *retry*.

Ak by detektor nenašiel žiadne záznamy so stavom *retry*, prezerá tabuľku change, či sa tu nachádzajú požiadavky so stavom *waiting*. Znamená to, že je nový záznam vo fronte a začína iterácia s najstarším dátumom a časom, nakoľko v tabuľke sa môže nachádzať viac položiek so stavom *waiting*. Pre konkrétne spracovávaný požiadavok sa mení jeho stav z *waiting* na *running*. Detektor zistí, aký je TypeOfChange požiadavku (type_id) a podľa toho spustí príslušné skripty, respektíve subpožiadavky (*dhcp.pl, dns.pl, sw.pl*).



Obr. 17 Vývojový diagram *detektor.pl* (časť 2)

Na obrázku Obr. 18 je možné vidieť tretiu časť vývojového diagramu detektoru. Detektor čaká na návratovú hodnotu spustených skriptov (subpožiadavkou). Tie mu vracajú niektorú z návratových hodnôt podľa tabuľky Tab. 10 spomínanej vyššie. V prípade, ak by neprichádzala návratová hodnota do 15 sekúnd zo strany subpožiadavku smerom k detektoru, tak je automaticky tento proces ukončený a subpožiadavok označený hodnotou 2 – RETRY. Po vrátení hodnôt zo všetkých subpožiadavkou, detektor kontroluje ich hodnoty.



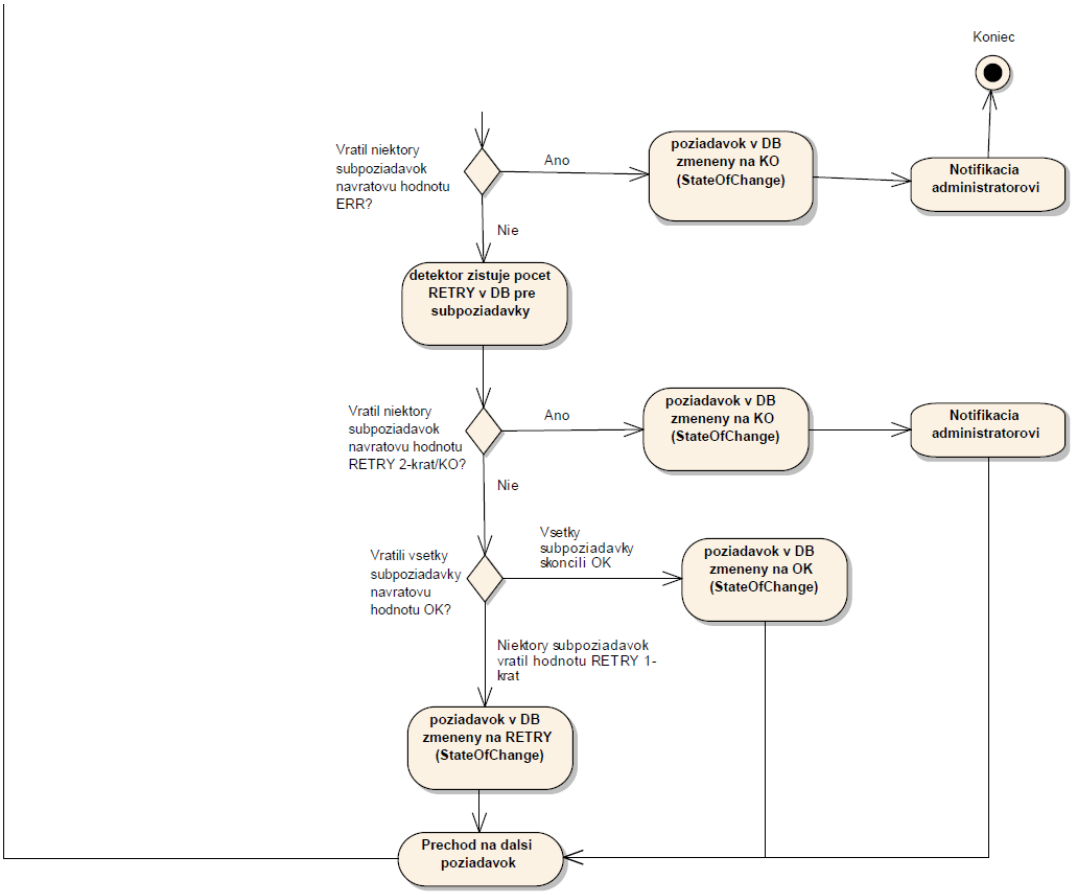
Obr. 18 Vývojový diagram *detektor.pl* (časť 3)

Záverečná časť detektoru, zobrazená na obrázku Obr. 19 znázorňuje vyhodnocovanie celkového požiadavku podľa subpožiadavkou. V prípade, ak niektorý subpožiadavok vrátil hodnotu ERR, ktorá vyjadruje chybu z niektorých služieb DHCP, DNS alebo TFTP, prípadne chybu pripojenia ku databáze či zariadeniu, celkový požiadavok je vyhodnotený ako neúspešný – KO. Odosiela sa email administrátorovi o chybe ktorá nastala a následne je detektor ukončený (dôvod spadnutá služba).

Ak nebol vrátený žiadny ERROR, detektor vyhodnocuje podľa počtov vrátených RETRY pre subpožiadavok ďalšie kroky. Ak niektorý subpožiadavok vrátil hodnotu RETRY dvakrát, celkový požiadavok je označený ako neúspešný – KO. V tomto prípade je opäť potrebné upozorniť administrátora o vyskytnutom probléme, a že neprebehla požadovaná úprava. Nie je ale potrebné ukončiť celý detektor, nakoľko sa môže jednať len o individuálnu chybu skriptu v podobe napríklad rozdielneho kontrolného súčtu konfigurácií na prepínači.

V prípade, že všetky subpožiadavky vrátili hodnotu OK, t. j. skončili v poriadku, je celkový požiadavok zmenený taktiež na stav OK a prechádza sa späť na krok prehľadávania tabuľky *change* v rámci iterácie, či sa nachádzajú nejaké záznamy so stavom *waiting*.

Ak nenastal žiadny ERROR a nevrátili všetky subpožiadavky kladnú hodnotu úspešnej úpravy, je jasné že niektoré subpožiadavky skončili s chybou. V tomto prípade prechádza celkový požiadavok do stavu *retry*, aby pri ďalšom spustení detektoru bol spustený znova.



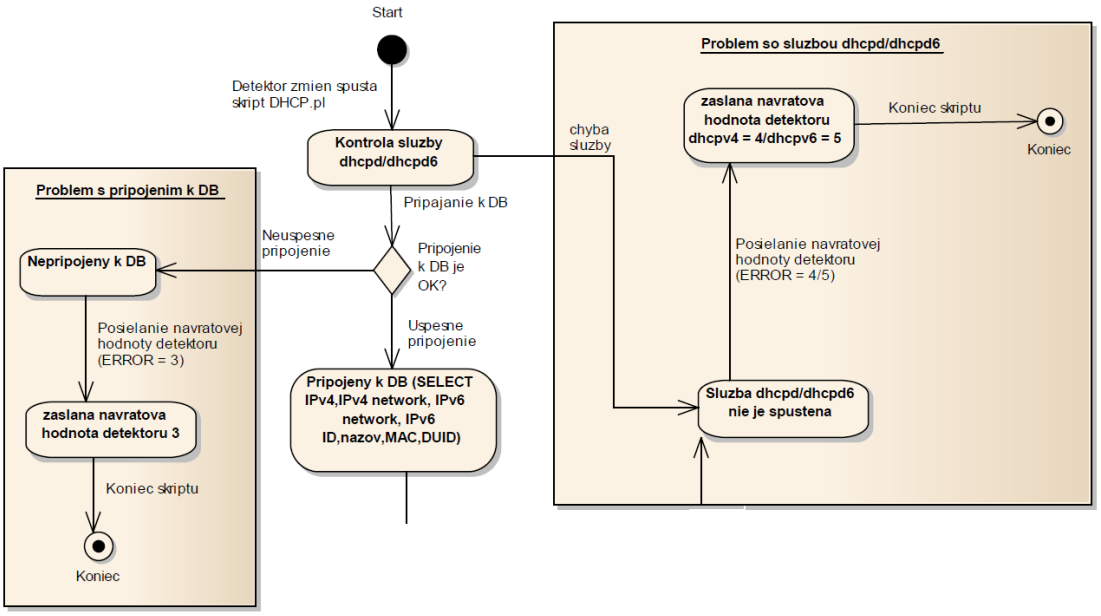
Obr. 19 Vývojový diagram *detektor.pl* (časť 4)

7.3.2 DHCP

Skript *dhcp.pl* je prvý z troch skriptov, ktoré sú spúšťané detektorom. Jeho návrh je zobrazený na nasledujúcich obrázkoch. Celý diagram sa nachádza v kapitole prílohy. [Príloha C]

Obrázok Obr. 20 vykresľuje prvú časť vývojového diagramu. Na začiatku je potrebné najprv skontrolovať, či služba nie je spadnutá, aby zbytočne sa neupravoval nefungujúci konfiguračný súbor servera. Ak skript zistí, že služba nie je aktívna, odosiela návratovú hodnotu detektoru s konkrétnou chybou, aby následne potom bol administrátor informovaný, čo presne sa udialo.

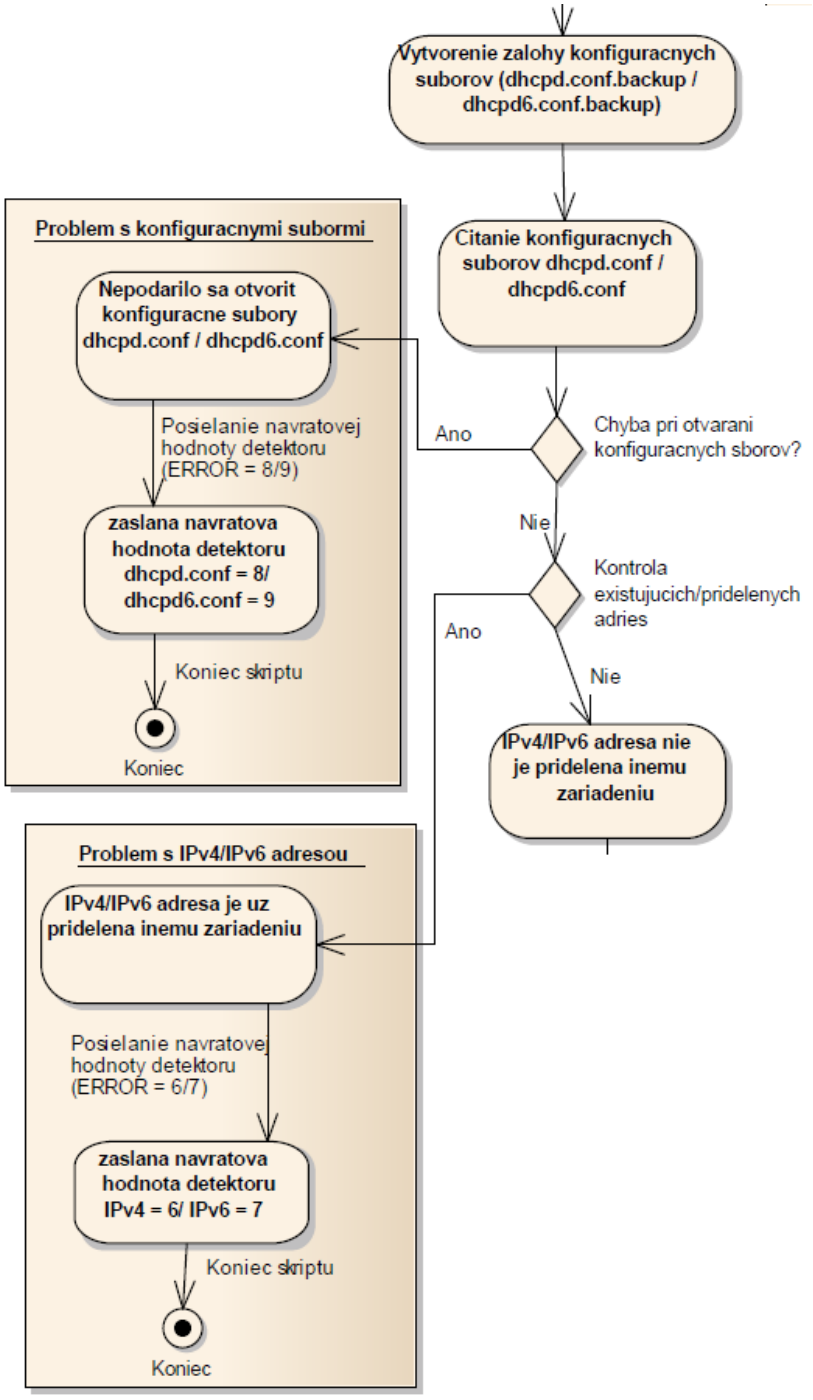
Druhý krok predstavuje pripojenie ku databáze. V prípade neúspechu je opäť odoslaná patričná návratová hodnota detektoru a skript ukončený. Ak pripojenie prebehne v poriadku, vytvára sa SQL dotaz pre zistenie IPv4 adresy, IPv4 adresy siete, IPv6 adresy siete, IPv6 ID pre konkrétne zariadenie, jeho názov, MAC adresa a príslušné DUID.



Obr. 20 Vývojový diagram *dhcp.pl* (časť 1)

Druhá časť diagramu vyobrazená na obrázku Obr. 21 začína vytvorením zálohy konfiguračných súborov ako pre DHCPv4 server tak pre DHCPv6. Tieto súbory sa v prípade neúspechu požadovanej úpravy nahrávajú späť ako aktívne.

Pri následnom prečítaní súčasných konfiguračných súborov, môže nastať chyba pri ich otváraní, a táto chyba je ihneď odoslaná detektoru. Ak čítanie prebehlo v poriadku, kontroluje sa, či požadované nové adresy nie sú už náhodou použité. Táto chyba by teoreticky nemala nastať, nakoľko by sa nemali vyskytovať duplicity už v samotnej databáze. Je to ale kontrolný bod, ak by náhodou niekto manipuloval priamo na servery s konfiguračnými súbormi. V prípade, že sa našla zhoda, nastáva upozornenie v podobe návratovej hodnoty smerom k detektoru, v opačnom prípade skript pokračuje ďalej.



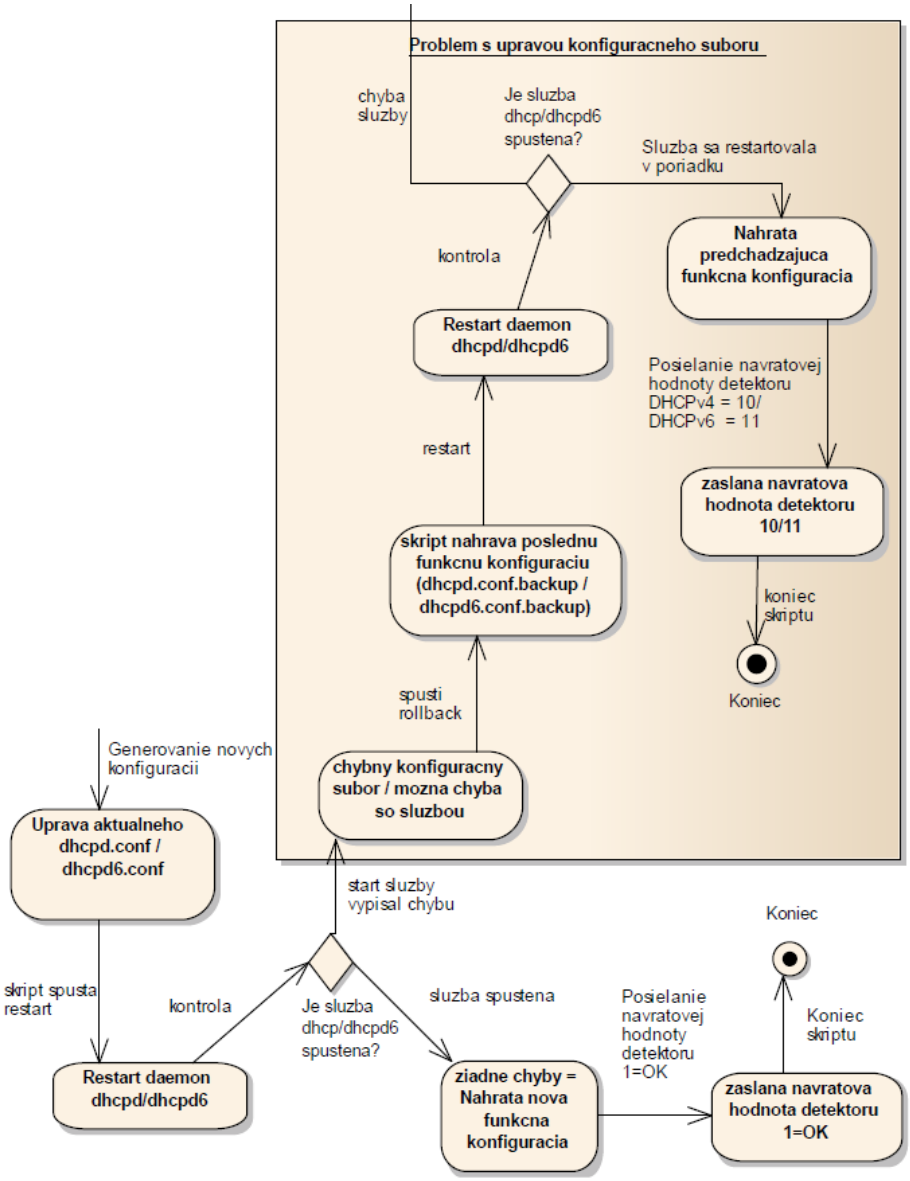
Obr. 21 Vývojový diagram *dhcp.pl* (časť 2)

Ďalším krokom skriptu je už priame generovanie a úprava súčasných konfiguračných súborov (viď. Obr. 22) podľa požiadavkou a hodnôt vybraných pomocou SQL dotazu na začiatku diagramu. Po úprave skript posiela príkaz systému na reštart služieb *dhcpd* a *dhcpd6*. Ak reštart prebehol v poriadku, nevrátil

žiadnu chybu a obidve služby bežia, je odoslaná kladná návratová hodnota detektoru, že požiadavok na úpravu serverov DHCPv4/v6 skončil v poriadku.

Pri neúspešnom reštarte služieb, sú nahraté záložné konfigurácie vytvorené pred úpravou súčasných. Skript opäť posiela príkaz systému na reštart služieb a v kladnom prípade je odoslaná návratová hodnota detektoru, že server DHCPv4/v6 beží, ale konfigurácia nebola upravená podľa požiadavkou.

V najhoršom prípade, že by sa služba nenaštartovala ani po nahratí záložných konfigurácií, sa prechádza na časť problém so službou *dhcpd/dhcpd6* v časti obrázku Obr. 20. V tomto prípade sa zasiela ERROR kód detektoru a je potrebný zásah administrátora.

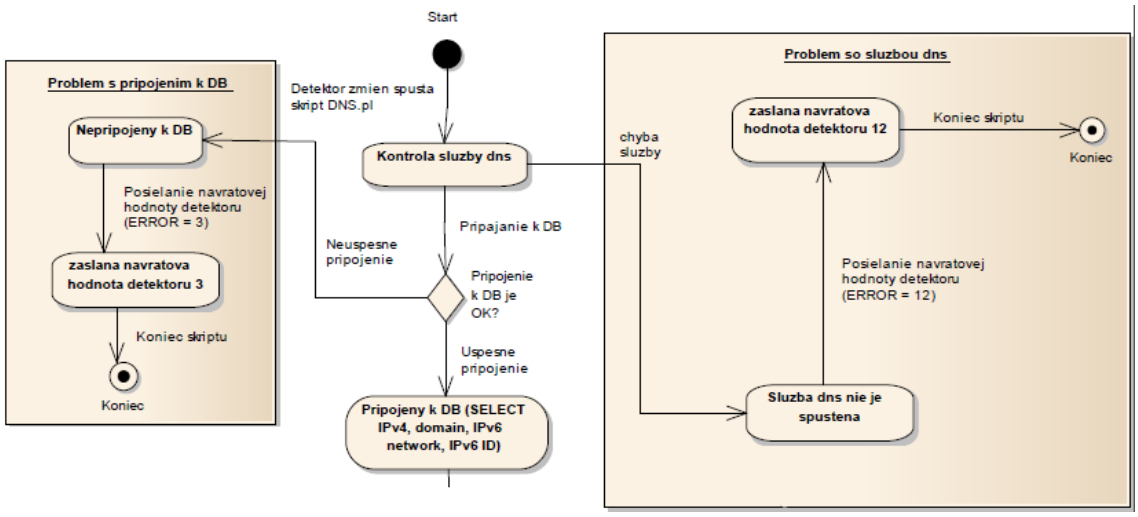


Obr. 22 Vývojový diagram *dhcp.pl* (časť 3)

7.3.3 DNS

Druhý skript, respektíve subpožiadavok, ktorý je spúšťaný detektorom je *dns.pl*. Diagram sa nachádza v kapitole prílohy. [Príloha D]
Nasledujúci obrázok Obr. 23 stvárňuje prvú časť z celého vývojového diagramu. Tak ako bolo v prípade *dhcp.pl* tak i *dns.pl* ako prvé kontroluje, či je služba aktívna. Bolo by zbytočné upravovať nefungujúci DNS server. V tomto prípade je odoslaná príslušná návratová hodnota detektoru o nájdenom probléme služby.

V kladnom prípade, že služba *dns* beží, nastáva kontrola pripojenia k databáze a po úspešnom pripojení vytvorený SQL dotaz pre výber IPv4 adresy, IPv6 adresy siete + IPv6 ID príslušného zariadenia a požadované doménové meno.



Obr. 23 Vývojový diagram *dns.pl* (časť 1)

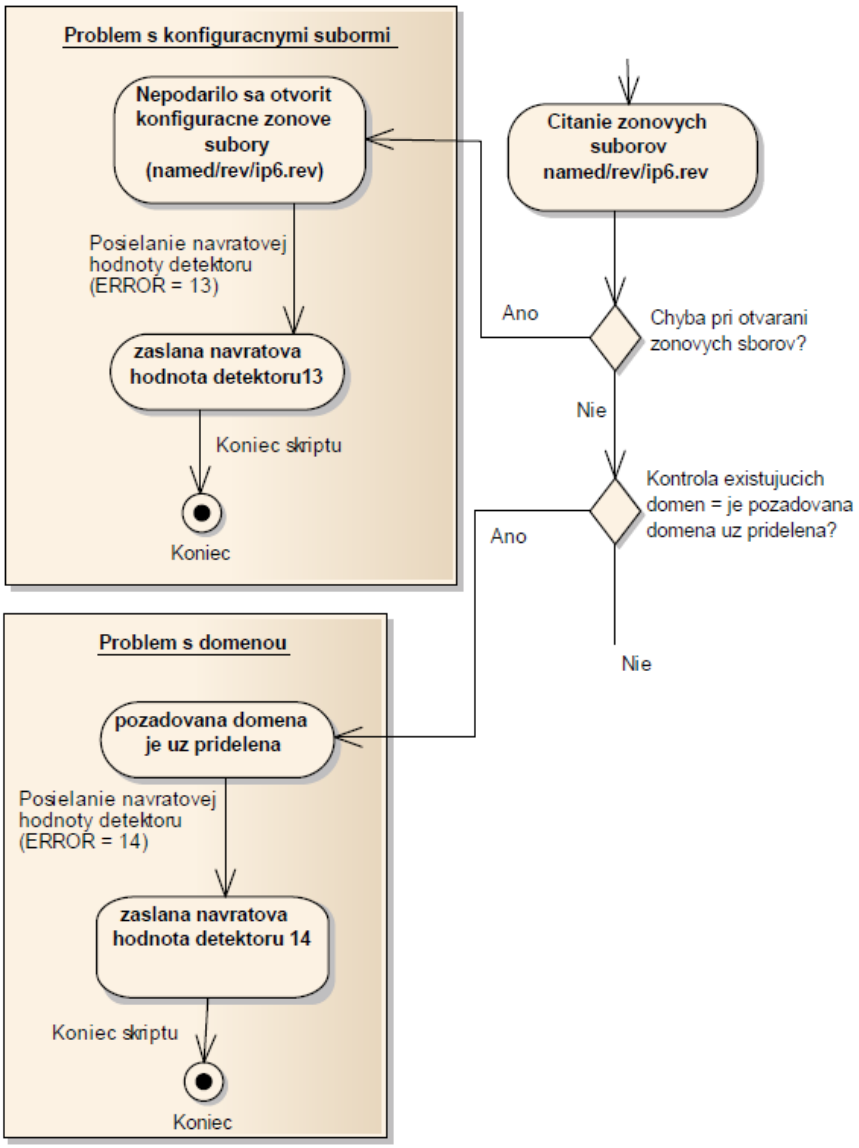
Skript pokračuje obrázkom Obr. 24, kde nasledujúcim krokom je čítanie konfiguračných súborov DNS servera. Jedná sa o konkrétne tri súbory, kde je potrebné mať zachovanú hierarchiu názvoslovia, aby skript bol schopný upravovať správne zónové súbory podľa získaného doménového mena z databáze. Pre lepšie pochopenie je uvedený nasledujúci príklad:
Máme doménu *mendelu.cz* a zariadenie, ktorému chceme priradiť doménové meno *pokus.mendelu.cz*.
Zónové súbory sa musia volať nasledovne:

- named.**mendelu.cz**
- **mendelu**.rev
- **mendelu**.ip6.rev

Skript v tomto prípade dokáže vyhľadať a upraviť správne zónové súbory.

Ak by skript zlyhal pri čítaní týchto súborov, respektíve nedokázal ich nájsť, je odoslaná návratová hodnota detektoru o príslušnom probléme. Ak čítanie prebehlo bez problémov, kontroluje sa, či doména už nebola pred tým použitá pre iné zariadenie. Opäť, tento problém by nemal teoreticky nastať, nakoľko by sa

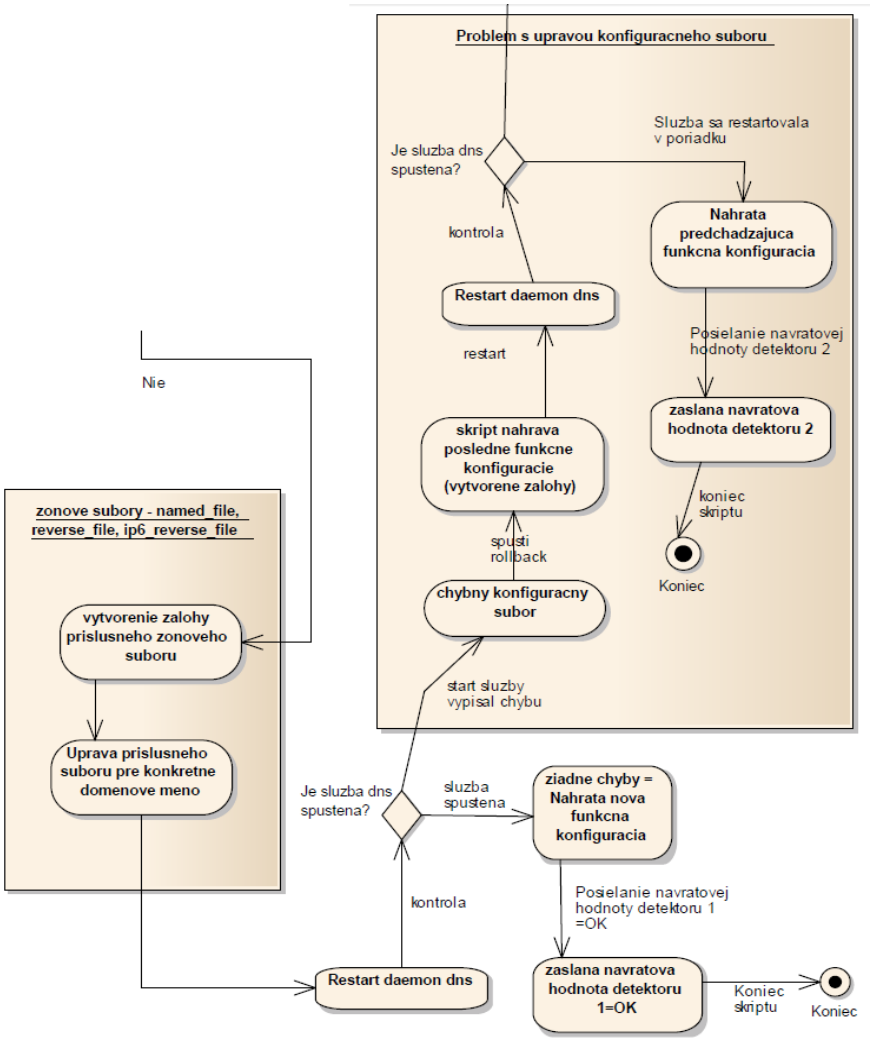
nemali vyskytovať duplicity už priamo v databáze, ale je to pre istotu z kontrolných dôvodov. Ak všetky tieto kontroly prebehnú v poriadku, skript pokračuje už úpravou jednotlivých, spomínaných, zónových súborov.



Obr. 24 Vývojový diagram *dns.pl* (časť 2)

Obrázok Obr. 25 vykresľuje poslednú časť vývojového diagramu *dns.pl*. Ako bolo spomínané vyššie, skript pokračuje už úpravou konkrétnych zónových súborov. Pri každom vytvára najprv zálohu a potom až následnú zmenu. Takto prejde postupne všetky tri typy súborov. Skript posiela príkaz systému pre reštart služby *dns*. V kladnom prípade, respektíve bez žiadnych vyskytnutých chýb je úprava dokončená a odoslaná návratová hodnota detektoru, že je všetko v poriadku. Ak by

reštart neprebehol v poriadku, sú nahraté záložné konfiguračné súbory, vytvárané pred úpravou. Nasleduje opätovný reštart služby *dns* a ak nabehne, odosiela sa návratová hodnota detektoru, ktorá vyjadruje, že DNS server beží, ale nie s požadovanou, novou úpravou. Ak by server nenabehol ani po nahratí záložných súborov, nastáva problém so serverom (viď. Obr. 23), kde sa posiela okamžitý signál detektoru, že DNS server je spadnutý. V tomto prípade je už potrebný zásah administrátora.



Obr. 25 Vývojový diagram *dns.pl* (časť 3)

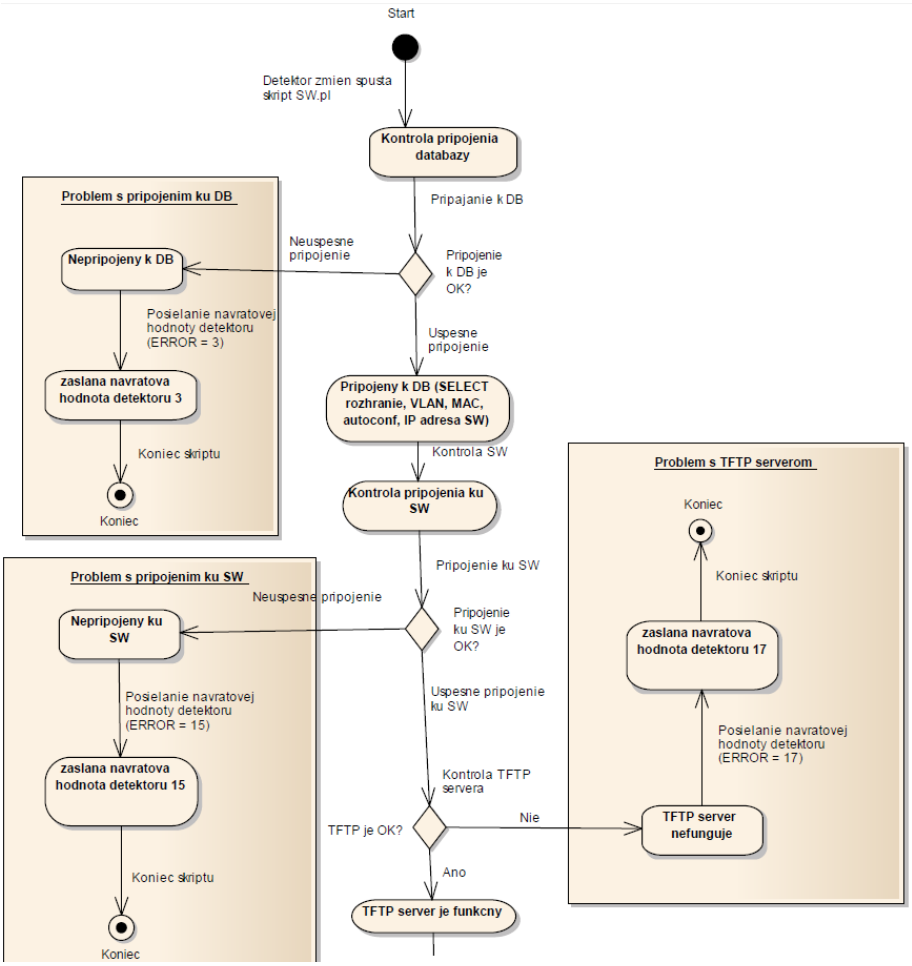
7.3.4 SW

Posledným skriptom, ktorý je spúšťaný opäť detektorom, je *sw.pl*. Tento skript komunikuje a upravuje CISCO zariadenia. Celý diagram je možné nájsť v kapitole prílohy. [Príloha E]

Obrázok Obr. 26 znázorňuje prvú časť skriptu *sw.pl*, ktorá sa venuje kontrolám. Zisťuje sa, či je možné sa pripojiť ku databáze a odoslať SQL dotaz pre

získanie informácií, konkrétne sa najprv zistí, či sa jedná o autokonfiguráciu nového zariadenia alebo len o úpravu niektorých rozhraní prepínača. Táto informácia je zistená z databázy a to hodnotou *autoconf*. Zadávateľ tejto práce, t. j. UIT MENDELU požadoval, aby pri registrácii nových zariadení bolo automaticky nahrané do týchto prepínačov čistá konfigurácia, viď. kapitola funkčné požiadavky. Ďalšie hodnoty, ktoré je potrebné získať z databázy, sú rozhrania, ktoré chceme upraviť, VLAN-a do ktorej rozhranie je priradené, MAC adresa zariadenia, ktorá sa nastaví ako *port-security* na dané rozhranie a samozrejme IP adresa, ktorá predstavuje manažment IP, cez ktorú sa pripojí skript ku zariadeniu.

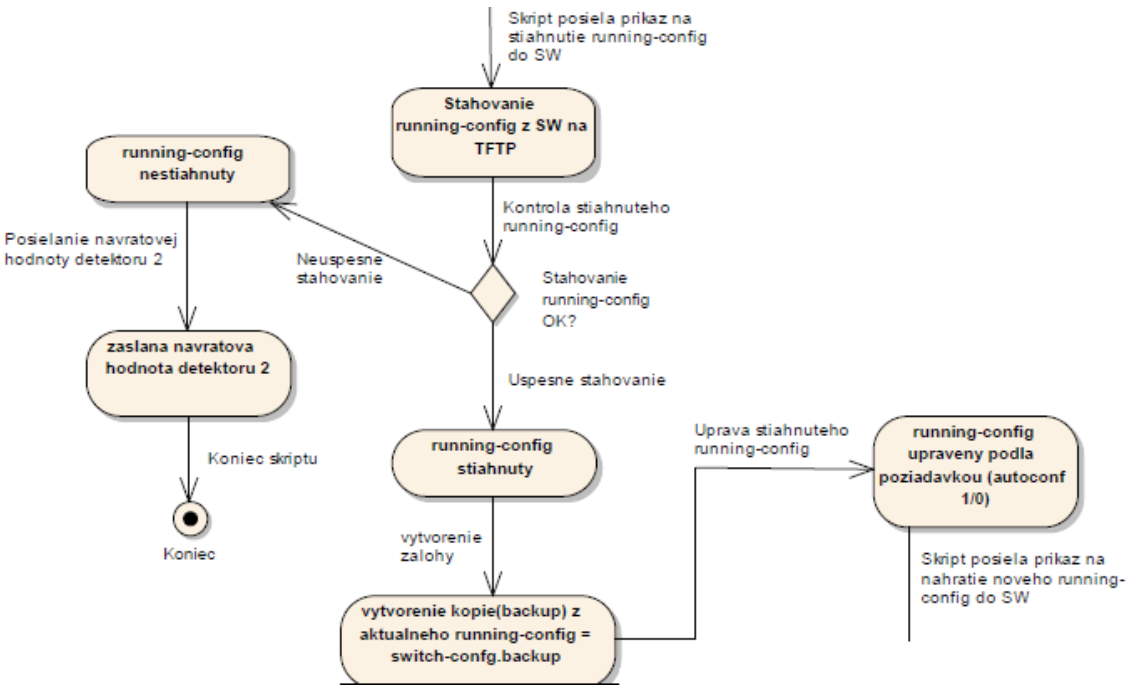
Skript po zistení dôležitých hodnôt pokračuje kontrolou pripojenia ku konkrétnemu prepínaču, pri neúspechu je odoslaná návratová hodnota detektoru. Ak je pripojenie úspešné, je potrebné ešte skontrolovať, či server TFTP je aktívny. Ak by nebol, nie je možné stiahnuť aktuálny *running-config* zo zariadenia a je odoslaná patričná návratová hodnota o chybe detektoru. Ak všetky kontroly skončili v poriadku, *sw.pl* pristupuje ku hlavnému bodu, a tým je úprava konfiguračného súboru zariadenia.



Obr. 26 Vývojový diagram *sw.pl* (časť 1)

Spomínaný TFTP server, kde sa ku zariadeniu pripája pomocou služby Telnet, je len jednou z variant, ako sa dá komunikovať a pracovať so sieťovými zariadeniami. Ďalšie alternatívne možnosti sú napríklad pomocou SSH alebo SNMP protokolu. SSH je dneska veľmi populárny, nakoľko umožňuje šifrovanie správ a pre budúcnosť by mohol byť tou správnou cestou. Je potrebné ale, aby na zariadeniach bola táto služba povolená, čo v dnešnej sieti MENDELU nie je. V protokole SNMP má každá hodnota jednoznačný identifikátor OID, ktorý je súčasťou stromovej štruktúry MIB, no nie je veľmi výhodný, nakoľko s príchodom rôznych verzií IOSov, sa jednotlivé identifikátori OID líšia. Viac o tom pojednáva taktiež prípadová štúdia odkazovaná v literatúre. [34]

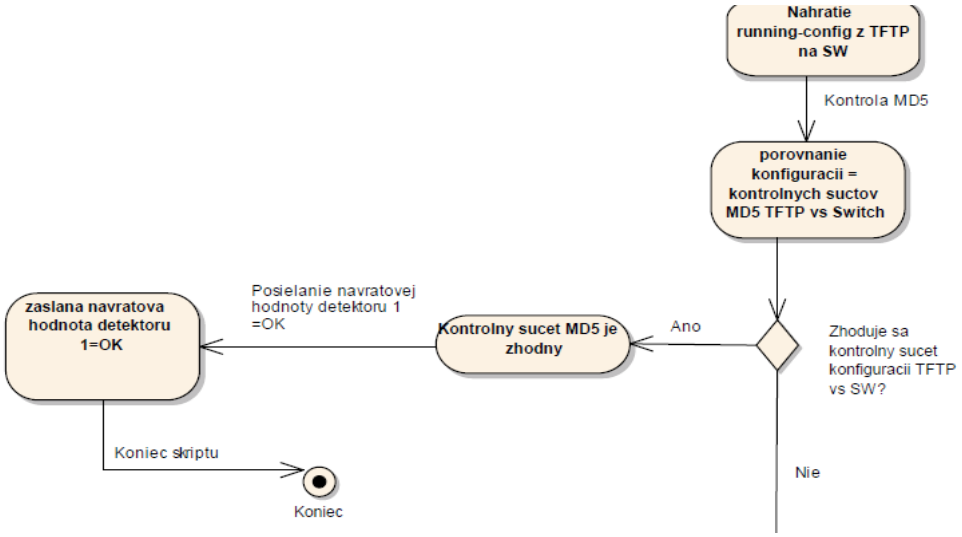
TFTP server bol taktiež zvolený z toho dôvodu, nakoľko nie je dobré posielať jednotlivé príkazy do zariadenia na upravenie konfigurácie, pretože môže dôjsť k strate niektorého paketu, a preto je potrebné aktuálny *running-config* zo zariadenia stiahnuť na náš TFTP server, viď. obrázok Obr. 27. Ak stiahnutie *running-config*-u sa nepodarí, je odoslaná návratová hodnota o chybe detektoru. Ak stiahnutie je úspešné, vytvára sa kópia, respektíve záloha konfiguračného súboru a nasleduje úprava podľa požiadaviek. Táto úprava závisí od spomínanej autokonfigurácie, či sa jedná o nové zariadenie alebo nie. Po úprave sa posiela príkaz na nahratie nového *running-config*-u do zariadenia.



Obr. 27 Vývojový diagram *sw.pl* (časť 2)

Obrázok Obr. 28 vyobrazuje kontrolu nahratej novej, upravenej konfigurácie do prepínača. Po nahratí je dôležité zistiť, či bola celá konfigurácia odoslaná úspešné. Tento úkon je realizovaný pomocou výpočtov MD5. Je vyrátaný kontrolný súčet

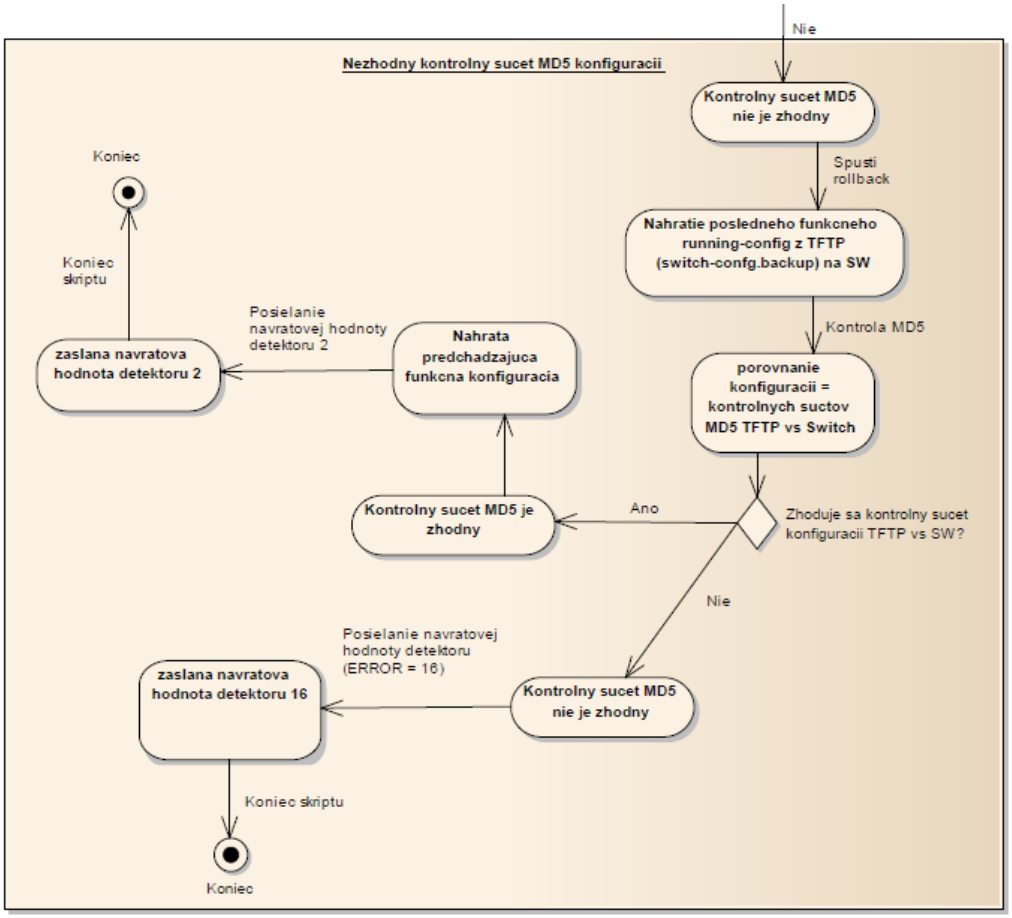
MD5 konfigurácie na TFTP servery a porovnaný s kontrolným súčtom konfigurácie nahratej na cieľovom zariadení. Ak sa zhodujú, všetko prebehlo v poriadku a je odoslaná kladná návratová hodnota detektoru. Ak sa kontrolné súčty rozchádzajú, je jasné, že došlo niekde k chybe a je potrebné, pre zachovanie funkčnosti zariadenia, nahrať záložnú, predchádzajúcu konfiguráciu prepínača.



Obr. 28 Vývojový diagram *sw.pl* (časť 3)

Ako bolo spomínané, pri nezhode kontrolných súčtov je potrebné nahrať predchádzajúcu konfiguráciu. Toto znázorňuje obrázok Obr. 29. Skript teda odosiela príkaz do zariadenia pre nahratie záložného *running-config*-u. Nasleduje opätovná kontrola pomocou súčtu MD5 konfigurácií a v kladnom prípade je odoslaná návratová hodnota detektoru, ktorá informuje, že úprava neprebehla, ale na zariadení je nahratá predchádzajúca, funkčná konfigurácia.

Ak by ani v druhom prípade, t. j. pri nahratí zálohy, sa nezhodoval kontrolný súčet, na zariadení je nekonzistentná aktuálna konfigurácia. Zariadenie v tomto prípade nemôže fungovať správne a je potrebné odoslať návratovú hodnotu detektoru a informovať tak administrátora.



Obr. 29 Vývojový diagram *sw.pl* (časť 4)

8 Implementácia

8.1 Popis implementácie

Implementácia začínala inštaláciou programového vybavenia jazyka perl pod operačným systémom Linux Centos7. Potrebné bolo taktiež nainštalovať LAMP[5] server, umožňujúci ďalšiu manipuláciu s webovým serverom, či databázou. V neposlednej rade, pre správnu funkcionalitu skriptov, je potrebné mať nainštalované moduly z celosvetovej siete CPAN. Konkrétne sa jedná o nasledovné moduly:

- **strict + warnings** – pomáhajú pri hľadaní chýb a ladení programu
- **DBI** – modul, ktorý umožňuje komunikáciu s databázou
- **Switch** – programový modul, ktorým je možné použiť štruktúru prepínača
- **Time::Out qw(timeout)** – zabezpečuje prerušenie volaného, druhého skriptu
- **MIME::Lite** – zabezpečuje posielanie emailovej komunikácie pomocou programu
- **Net::Telnet::Cisco** – modul, ktorý umožňuje komunikáciu s CISCO zariadeniami
- **Term::ANSIColor qw(:constants)** – doplnkový modul, ak je potrebné mať výpisy programu a informačný priebeh prehľadne, respektíve rôznymi farbami

Ďalším podstatným krokom je spomínaná databáza, bez ktorej by nebolo možné implementáciu uskutočniť. Návrh prebiehal spôsobom spoločných stretnutí spolu s UIT MENDELU a študentom Bc. Oliverom Horečným, ktorý ju aj vytvoril. Pre účely tejto práce bol prevzatý, exportovaný súbor, obsahujúci SQL dotazy pre vytvorenie konkrétnej databázy a nahratý na lokálny LAMP server.

Nasledovné podkapitoly opisujú len hlavné časti kostry jednotlivých kódov, respektíve skriptov. Celé kódy implementácie je možné nájsť na CD priloženom ku diplomovej práci a to konkrétne:

- Skript detektor.pl (1004 riadkov)
- Skript dhcp.pl (260 riadkov)
- Skript dns.pl (265 riadkov)
- Skript sw.pl (472 riadkov)

[5] LAMP - Linux, Apache, MySQL, Perl/Python/PHP

8.2 Implementácia skriptu detektor.pl

Pripojenie ku databáze

Pripojenie prebieha pomocou spomínaného modulu DBI. Je volaná metóda *connect*, ktorá vytvára hlavné pripojenie ku databáze. V tomto prípade sa jedná o databázu *sprava-site*, *host* predstavuje cieľovú adresu umiestnenia databázy a potrebné sú samozrejme prihlasovacie údaje – meno *root* a heslo *****. V prípade neúspechu pripojenia sa volá naprogramovaná funkcia *errorDB()*, ktorá kontaktuje administrátora pomocou emailu.

```
my $dbh = DBI->connect('DBI:mysql:database=sprava-site;host=localhost',
'root','*****') or die errorDB();
my $sth = $dbh->prepare('SELECT COUNT(state_id) as pocet FROM change WHERE
state_id=1');
       $sth->execute();

       while (my $ref = $sth->fetchrow_hashref()){
         $test = $ref->{'pocet'};
         }
$sth->finish();
```

Metoda *prepare* modulu DBI pripravuje a spracováva už konkrétny SQL dotaz, ktorý v tomto prípade hľadá, či sa v databáze vyskytuje nejaký nový záznam, t. j. so stavom *waiting*. Tento dotaz je potom odoslaný pomocou metódy *execute()*. Následný cyklus *while* umožňuje získané hodnoty SQL dotazom predať do premennej *$test*, s ktorou je potom možné pracovať v ďalších častiach programu.

Komunikácia pomocou emailu

Nasledujúci príklad, funkcia *errorDB()* umožňuje informovať administrátora prostredníctvom emailu. Používa spomínaný modul zo siete CPAN MIME::Lite. Je potrebné určiť programátorom obsah a predmet správy. Následne je cez modul vytvorený objekt, ktorý v sebe zahŕňa okrem obsahu a predmetu správy, taktiež od koho a komu je email adresovaný. Nakoniec pomocou príkazu *send* je odoslaný príkaz na zaslanie emailu.

```
sub errorDB{
       my $subject = "databaza";
         my $message = "neuspesne pripojenie ku DB";
         my $msg = MIME::Lite->new(
         From    => from,
         To      => to,
         Subject => $subject,
         Data    => $message
       );
         $msg->send;
}
```

Iterácia pre požiadavky so stavom *retry*

Nasledujúci kód vychádza z opisovaného vývojového diagramu vyššie (Obr. 17), konkrétne rozhodová časť (diamant) s názvom „Nájdený požiadavok v stave

retry?“, kedy detektor prehľadáva tabuľku *change*. Ako prvé, hlavný skript *detektor.pl* kontroluje, či sa nachádzajú v tabuľke *change* nejaké požiadavky so stavom *retry*, teda tie, čo skončili prvýkrát s chybou. SQL dotazom sa vyberú z tabuľky *change* všetky *id*, ktoré sú v stave *retry*, respektíve so *state_id* = 5. Všetky tieto záznamy sú ukladané do poľa *array*.

```
#zistenie ID poziadavku, ktore su RETRY
my $sth = $dbh->prepare('SELECT id FROM change WHERE state_id=5');
$sth->execute();
my @array;
my $i=0;
while(my $ref = $sth->fetchrow_hashref){
                $array[$i] = $ref->{'id'};
                $i++;
}


...
...
...
```

V premennej *$pocet* sa nachádza číslo v závislosti výskytov požiadaviek v stave *retry*. Napríklad, ak v tabuľke *change* sú tri požiadavky so stavom *retry*, hodnota premennej *$pocet* = 3 a následný cyklus pôjde trikrát, pre každý požiadavok. Premenná *$id* nadobúda v každej novej iterácii cyklu *while* hodnotu *id* z poľa *array*, opísaného v predchádzajúcom príklade. Následný SQL SELECT vyberá z tabuľky *change* stavy subpožiadavkou *dhcp, dns, sw* pre požiadavok s daným *id* a ukladá do premenných *$DNS, $DHCP, $SW*.

Následná časť kódu potom kontroluje, v ktorých z premenných sa nachádza hodnota 2, ktorá symbolizuje chybu, pri predchádzajúcom spustení subpožiadavku. Ak nastane zhoda, daný subpožiadavok je spustený znova, respektíve druhýkrát. Sú volané funkcie – *skriptDHCP(), checkDHCPsub(), checkFULLRequest()* opisované v ďalších častiach, podľa toho, ktorý subpožiadavok skončil *retry*.

```
while ($pocet>0){
       $id = $array[$pocet-1];
         $sth = $dbh->prepare('SELECT dns,dhcp,sw FROM change WHERE id='.$id);
         $sth->execute;
         while(my $ref = $sth->fetchrow_hashref){
                    $DNS = $ref->{'dns'};
                    $DHCP = $ref->{'dhcp'};
                    $SW = $ref->{'sw'};
         }

#kontrola stavov subpoziadavkou,ktore su RETRY
       if ($DNS==2){
#zavola funkciu na skriptDNS druhykrat
       $nh = skriptDNS($id);
#update v DB pre subpoziadavok
       my $return = checkDNSsub($nh);
       print $return;
#update celkoveho poziadavku
```

```
        checkFULLRequest($return);
        }

        if ($DHCP==2){
#zavola funkciu na skriptDHCP druhykrat
        $nh = skriptDHCP($id);
#update v DB pre subpoziadavok
        my $return = checkDHCPsub($nh);
        print $return;
#update celkoveho poziadavku
        checkFULLRequest($return);
        }
        ...
        ...
        ...
```

Iterácia pre požiadavky so stavom *waiting*

Skript *detektor.pl* po absolvovaní iterácie pre požiadavky v stave *retry* zisťuje, či sa nachádzajú v tabuľke *change* nové požiadavky so stavom *waiting,* druhá vetva v diagrame na obrázku Obr. 17.

Funkcia *zaznamNEW()* vracia hodnotu, ktorá obsahuje v sebe počet požiadavkou v stave *waiting*. Ak sa v tabuľke nachádzajú takéto záznamy, detektor berie požiadavok vo fronte s najstarším dátumom a hlavne časom. Pre tento požiadavok sa následne mení jeho stav z *waiting* na *running* – *state_id*=2.

```
while (zaznamNEW()>0){
        #detektor berie jeden zaznam s najstarsim datumom a casom
        my $sth = $dbh->prepare('SELECT id,device_id,type_id FROM change WHERE
         state_id=1 ORDER BY updated_at LIMIT 1');
        $sth->execute();
        while (my $ref = $sth->fetchrow_hashref()){
                $id = $ref->{'id'};
                $id_zarizeni = $ref->{'device_id'};
                $id_zmeny = $ref->{'type_id'};
        }
        $sth->finish();

        #update poziadavku z waiting na running
        $sth = $dbh->prepare("UPDATE change SET state_id=2"." WHERE id=".$id);
          $sth->execute();
        $sth->finish();
...
...
```

Hlavný switch podľa typu požiadavku

Predchádzajúci kód pokračuje iteráciou pre požiadavky so stavom *waiting*. Ten podľa typu požiadavku *type_id* uloženého v premennej *$id_zmeny* spúšťa danú časť switchu. Postup v jednotlivých prípadoch je podobný ako bolo v predchádzajúcom prípade pre požiadavky typu *retry*. Rozdiel nastáva, ak je potrebné spustiť viac subpožiadavkou, napríklad v prípade *case 6*, kedy je požadované pustiť

DNS+SW subpožiadavok. V tom prípade sa ukladajú ich výsledky do dvoch premenných, konkrétne *$return* a *$return2* a celkový požiadavok sa vyhodnocuje na základe ich hodnôt.

```
switch($id_zmeny){
        case ('1') {
                    #_____DHCP_____
                #volanie funkcie na spustenie skriptu
                  $nh = skriptDHCP($id);
                #update v DB pre subpoziadavok
                  my $return = checkDHCPsub($nh);
                #update celeho poziadavku v DB podla subpoziadavkou
                  checkFULLRequest($return);
         }
        case ('2'){
              ...
         }
        case ('3'){
              ...
         }
       ...
       ...
        case ('6') {
                    #_____DNS+SW_____
                #volanie funkcie na spustenie skriptu DNS
                    $nh = skriptDNS($id);
                    my $return = checkDNSsub($nh);
                    ...

                #volanie funkcie na spustenie skriptu SW
                    $nh2 = skriptSW($id);
                    my $return2 = checkSWsub($nh2);
                    ...

               #update celeho poziadavku v DB podla subpoziadavkou
               #najprv kontrola return a return2
                    if (($return == 1) and ($return2 == 1)){
                          checkFULLRequest($return);
                    }

                     if ($return == 3){
                           checkFULLRequest($return);
                    }

                     if ($return2 == 3){
                           $nh = $nh2;
                           checkFULLRequest($return2);
                    }
                    }
       ...
       ...
```

Volanie/spustenie iného skriptu detektorom

Funkcia *skriptDHCP()* je jedna z troch funkcií, ktorá volá spustenie iného skriptu, v našom prípade subpožiadavku (časť vývojového diagramu zobrazeného na obrázku Obr. 18). Funkcia, pri jej volaní, prevezme *id* do premennej *$skript_id[0]*.

Premenná $nb_secs určuje počet sekúnd, po dobu ktorých musí volaný subpožiadavok vrátiť hodnotu.

Následne je volaný modul TIMEOUT, kde v prvej časti pomocou príkazu *require* je načítaný skript dhcp.pl (náš subpožiadavok) a volaná jeho funkcia DHCP, ktorej sa odovzdáva prevzaté *id* v premennej *$skript_id[0]*. Táto funkcia potom vracia návratovú hodnotu, podľa jej úspechu či neúspechu, a ukladá ju do premennej *$skript_nh*. Ak nevráti žiadnu návratovú hodnotu po dobu spomínaných 15 sekúnd, funkcia je ukončená, a návratová hodnota je nastavená do premennej *$skript_nh* konkrétne na hodnotu 18 (viď. tabuľka Tab. 10). Uložená hodnota v premennej *$skript_nh* je vrátená späť detektoru, ktorý ju podľa jej obsahu ďalej vyhodnocuje.
Ostatné dve funkcie *skriptDNS()* a *skriptSW()*, pre volanie ostatných subpožiadavkou vyzerajú identicky, samozrejme až na ich názov a volanie funkcií.

```
sub skriptDHCP{
       my @skript_id = @_;
       my $skript_nh;
       #TIMEOUT
       my $nb_secs = 15;
         timeout $nb_secs => sub{
         #spusta prikaz pre spustenie DHCP skriptu
         require("/home/dany/Plocha/testovanie/dhcp.pl");
         $skript_nh= (DHCP($skript_id[0]));
       };

         if ($@){
               # ked vyprsi timeout = co sa stane
                  print "timeout vyprsal\n";
                  print "zastavenie procesu\n";
                  $skript_nh =18;
         }
         return $skript_nh;
}
...
...

sub skriptDNS{
...
}

sub skriptSW{
...
}
```

Funkcia pre aktualizáciu subpožiadavkou v databáze podľa návratových hodnôt

Nasledovná funkcia *checkDHCPsub()* je opäť jedna z troch funkcií pre vyhodnotenie subpožiadavkou, respektíve aktualizáciu ich stavu v databáze. Vychádza z poslednej časti opisovaného vývojového diagramu na obrázku Obr. 19.

Základom funkcie je štruktúra switch, ktorá podľa hodnoty (hodnota funkcie *skriptDHCP()*) prevzatej do premennej *$hodnota[0]* pri volaní opisovanej

funkcie, pokračuje v príslušnej časti prepínača. Ak subpožiadavok skončil v poriadku a vrátil návratovú hodnotu 1, je vytvorený SQL dotaz pre aktualizáciu stĺpca *dhcp* v tabuľke *change* a funkcia vracia detektoru rovnakú hodnotu. Ak vráti niektorú z hodnôt 8-11, čo predstavujú chybu pri čítaní konfiguračných súborov, alebo chybu pri zápise (viď. Tab. 10), zisťuje sa najprv z databázy, či sa to udialo už druhýkrát. To je uskutočnené pomocou SQL dotazu a hodnota je uložená do premennej *$retry*. Ak sa v premennej nenachádza žiadna hodnota, respektíve hodnota 0, jedná sa o chybu, ktorá sa stala prvýkrát a subpožiadavok je potrebné v ďalšom cykle detektoru opätovne spustiť. Zapíše sa preto do databáze pod stĺpec *dhcp* ku danému požiadavku hodnota 2 a funkcia vracia detektoru stav RETRY pre daný subpožiadavok.

Ak sa v premennej *$retry* nachádza hodnota 2, signalizuje, že sa jedná už o druhé spustenie a rovnakú chybu. Tým pádom vracia funkcia *checkDHCPsub()* detektoru hodnotu 3, reprezentujúcu stav KO.

V rovnakom princípe sa pracuje i pri stave TIMEOUT daného subpožiadavku. Musí dvakrát nastať a až potom je vyhodnotený ako KO.

Čo sa týka hodnôt 3 až 7, signalizujúce problémy so službami *dhcpd* alebo *dhcpd6*, v tomto prípade je hneď subpožiadavok vyhodnotený ako KO hodnotou 3. Ako bolo už spomínané, je zbytočné spúšťať subpožiadavok znova, ak je problém so serverom.

```
sub checkDHCPsub{
       my @hodnota = @_;
switch($hodnota[0]){
       case('1'){
       #skript je OK
              my $sth = $dbh->prepare("UPDATE change SET dhcp=1 WHERE id=".$id);
              ...
              ...
              return 1;
       }

       case[8..11]{
       #skript havaroval RETRY
       #havaroval uz viackrat?
             my $retry;
             my $sth = $dbh->prepare("SELECT dhcp FROM change WHERE id=".$id);
                ...
                ...
             if ($retry==2){
             #havaroval uz 2-krat
              my $sth = $dbh->prepare("UPDATE change SET dhcp=3 WHERE id=".$id);
               ...
               ...
             return 3;
             }else{
              #havaroval iba raz -> RETRY
              my $sth = $dbh->prepare("UPDATE change SET dhcp=2 WHERE id=".$id);
                     ...
                     ...
              return 2;
             }
       }
```

```
        case[3,4,5,6,7]{
                #skript vyhodil ERR
                ...
                ...
                return 3;
        }

        case ('18'){
                #skript timeout-oval 15s -> RETRY 1-krat
                #timeoutoval uz viackrat
                                ...
                        return 3;
                   }else{
                   #timeout len raz
                                ...
                                return 2;
                      }
        }
}
}

sub checkDNSsub{
...
}

sub checkSWsub{
...
}
```

Funkcia pre aktualizáciu celkového požiadavku

Podľa návratových hodnôt, ktoré vracajú funkcie *checkDHCPsub(), checkDNSsub(), checkSWsub()*, vyhodnocuje nasledovná funkcia *checkFULLRequest()* celkový požiadavok v databáze. Táto časť kódu sa odkazuje na rozhodovú časť(diamant) v diagrame na obrázku Obr. 19 s názvom „Vrátili všetky subpožiadavky návratovú hodnotu OK?“. Pracuje na podobnom princípe ako predchádzajúce funkcie na vyhodnotenie subpožiadavkou s tým rozdielom, že aktualizuje celkový stav – *state_id*. Hodnota 1 signalizuje, že všetky subpožiadavky skončili v poriadku – OK, hodnota 2 – RETRY, hodnota 3 – KO. Naviac sa tu nachádza vetva v prípade hodnoty 3 – KO, kedy sa zisťuje, akú konkrétne chybu vrátil subpožiadavok a informuje sa administrátor. Ak sa jedná o chyby typu spadnutý DHCPv4/v6 server alebo spadnutý TFTP server a podobne, detektor je zastavený pre zbytočné spúšťanie ďalších požiadavkou.

```
sub checkFULLRequest{
        my @hodnota = @_;
switch($hodnota[0]){
        case('1'){
        #celkovy poziadavok je OK
         my $sth = $dbh->prepare("UPDATE change SET state_id=3 WHERE id=".$id);
                $sth->execute();
                $sth->finish();
        }

        case[2]{
```

```
        #chyba subpoziadavku = RETRY
        my $sth = $dbh->prepare("UPDATE change SET state_id=5 WHERE id=".$id);
                $sth->execute();
                $sth->finish();
        }

        case[3]{
        #celkovy poziadavok je KO (bud skript alebo sluzba ERR)
        my $sth = $dbh->prepare("UPDATE change SET state_id=4 WHERE id=".$id);
                $sth->execute();
                $sth->finish();
        #informovanie administratora podla druhu chyby
        error($nh);

        #zastavenie detektoru ak je niektora zo sluzieb spadnuta
        if ( ((2<$nh)and($nh<6)) or ($nh == 12) or ($nh == 15) or ($nh == 17) ){
                my $pid = qx/ps -ef | grep -v grep | grep detektor  | awk '
{print \$2}'/;
                system 'kill '.$pid;
        }
}

}
}
```

8.3 Implementácia skriptu dhcp.pl

Úprava DHCPv4 servera

Na začiatku skriptu sa odosiela SQL dotaz pre získanie informácií z databázy – IPv4 adresa, MAC adresa, meno zariadenia, adresa siete a maska siete.

Ak sa vychádza podľa vývojového diagramu na obrázku Obr. 22, začína úprava konfigurácií servera. Do poľa *buffer4* je načítaný konfiguračný súbor *dhcpd.conf* a nasleduje jeho postupné prehľadávanie riadok po riadku, kde sa zisťuje a porovnáva požadovaný, respektíve hľadaný subnet, do ktorého má byť zaradený. Po jeho nájdení je potrebné nájsť pozíciu slova group, vyjadrujúce uplatnenie nastavení (gateway, broadcast address, ...) pre všetky adresy subnetu. Práve na túto pozíciu je potrebné vkladať informácie pre nové zariadenie.

Na podobnom princípe pracuje i funkcia pre úpravu DHCPv6 konfiguračného súboru. Rozdiel je v tom, že pri verzii 6 nie je potrebné hľadať konkrétnu pozíciu subnet-u a group-y, ale stačí pridávanie požadovaného, nového zariadenia na koniec konfiguračného súboru *dhcpd6.conf*.

```
for $line (@buffer4) {
                #porovnavam so subnetom
                if (($line =~ $adresa_site) and ($line =~ $maska_siteIPv4)){
                        $find++;
                }
                        #kontroluje sa najblizsia group za subnetom
                        if ($line =~ "group" and $find==1){
                        #vkladam novu IP
                        splice @buffer4, $i+1, 0,
"host $name {
hardware ethernet $MAC;
fixed-address $IP;
```

```
}\n";

                    $find++;
                    print BOLD GREEN, "new IP added to subnet\n", RESET;
                    $restart4++;
                }
            $i++;

        }
...
...
```

Reštart služieb/nahratie záložných konfigurácií

Po úprave konfigurácií súborov *dhcpd.conf* a *dhcpd6.conf* , ako znázorňuje ďalšia časť diagramu Obr. 22, nasleduje reštart služieb. Skript posiela príkaz do systému a je mu vrátená hodnota 0 alebo 1. V prípade návratu hodnoty 0, beží všetko v poriadku a úprava je úspešná. V opačnom prípade sa prepisujú upravené konfiguračné súbory záložnými, ktoré sa vytvorili pred ich úpravou. Opakuje sa reštart služieb a podľa výsledku je potom odoslaná návratová hodnota detektoru.

```
#restart sluzby DHCPv4
my $cmd4 = system "service dhcpd restart";

#restart sluzby DHCPv6
my $cmd6 = system "service dhcpd6 restart";

if (($cmd4 == 0)and($cmd6 == 0)){
        #vsetko bezi ok
        print BOLD GREEN, "service dhcpdv4/v6 restarted succesfully\n", RESET;
                $navrat = 1;
        }else{
                #nahratie predchadzajucej konfiguracie
                my $oprava4 = system "cp /etc/dhcp/dhcpd.conf.backup
/etc/dhcp/dhcpd.conf";
                my $oprava6 = system "cp /etc/dhcp/dhcpd6.conf.backup
/etc/dhcp/dhcpd6.conf";
...
...
```

8.4 Implementácia skriptu dns.pl

Úprava zónového súboru named.doména.cz

Ako prvý z troch zónových súborov sa upravuje named.doména.cz. Skript si z požadovaného doménového mena, ktoré sa pridáva, napríklad *pokus.mendelu.cz*, vyberie názov domény *mendelu* a podľa toho hľadá príslušné súbory. Práve tento názov domény *mendelu* je obsahom premennej *$hladaj* v nasledujúcej ukážke-časti kódu.

Po načítaní príslušného súboru, sa kontroluje duplicita doménového mena (diagram Obr. 24). Ak kontrola prebehne v poriadku, vytvára sa záloha súboru a pokračuje sa v požadovanej úprave (diagram Obr. 25 časť zónové súbory). Tá je ukončená pridaním doménového mena na koniec súboru.

```
#nacitanie named
my $named_file = "/var/named/named.$hladaj";
open (IN, $named_file) || die $navrat = 13;
my @buffer_named = <IN>;
close IN;
...
...
#kontrola, ci uz domena je priradena
if (grep (/$meno/, @buffer_named)){
       print "Domain name is assigned already\n";
       $navrat = 14;
}else{
#pokracuje sa v uprave konfiguracie
...
...
#zalohovanie
my $backup_named = system "cp $named_file $named_file.backup";

#zapis named
open(OUT,">>", $named_file);
print OUT "$meno           IN     A      $IP\n";
print OUT "         IN     AAAA   $IPv6\n";
close OUT;
}
```

Úprava zónového súboru doména.ip6.rev

Úpravy zónových súborov prebiehajú rovnakým spôsobom ako je spomínané vyššie. Špeciálnosť prichádza v prípade konfigurácie pre verziu ipv6, kedy je potrebné ipv6 adresu upraviť do požadovaného tvaru. Tá je potom uložená do premennej *$dns_adresa*.

Ak sa jedná o zmenu súčasného doménového mena pre zariadenie, postupuje sa podobným spôsobom s tým rozdielom, že sa aktuálne doménové meno zariadenia najprv vymaže a potom sa pokračuje v požadovanej úprave.

```
#pocitanie velkosti IPv6 ID
my $pocet = 0;
$pocet = length $IPv6ID;

my $dns_adresa = join(".",split(//,$IPv6ID));

print "$dns_adresa\n";
#doplnenie priestoru 0-mi
while ($pocet < 16){
       $dns_adresa = join "",$dns_adresa, ".0";
       $pocet++;
}

#nacitanie suboru ip6.rev
my $rev_ip6_file = "/var/named/$revers.ip6.rev";
open (IN, $rev_ip6_file) || die $navrat = 13;
my @buffer_ip6_rev = <IN>;
close IN;
...
```

8.5 Implementácia skriptu sw.pl

Pripojenie ku zariadeniu CISCO

Pre pripojenie ku zariadeniu sa využíva modul *Net::Telnet::Cisco*. Funkcia *connectSW* prevezme ako parameter IP adresu zariadenia, ku ktorému sa skript pripája. Pre úspešné prihlásenie je potrebné nastaviť prihlasovacie údaje, ako pre pripojenie telnet tak i pre prepnutie z užívateľského módu zariadenia do priviligovaného režimu. Ak je pripojenie úspešné, funkcia vracia kód 1, a skript *sw.pl* potom môže pokračovať ďalej.

```
sub connectSW{
my @connect_IP = @_;
#pripojenie ku zariadeniu

my $session = Net::Telnet::Cisco->new(Host => $connect_IP[0], Errmode => 're-
turn');
if(!$session || !$session->login('cisco','cisco123')){
	return 15;
}else{
	#enable mode => Privileged EXEC Mode
	if ($session->enable("cisco123") ) {
		print BOLD, "Connection to the SW successfully\n\n", RESET;
		return 1;
	}else {
		return 15;
	}
}
}
```

Úprava *running-config*

Vychádza z diagramu zobrazeného na obrázku Obr. 27. Po odoslaní príkazu do zariadenia na stiahnutie aktuálneho *running-config*-u nasleduje jeho úprava. Konfigurácia je načítaná v poli *buffer* a postupne sa prechádza každý riadok, ktorý je načítaný do premennej *$line*. Premenná *$string* obsahuje názov rozhrania, respektíve kľúčové slovo *interface*. Keď cyklus foreach narazí na riadok, kde sa toto kľúčové slovo nachádza, tak sa na nasledujúci riadok vkladajú požadované príkazy.

Pri úprave konfigurácie zariadenia, kedy sa menia niektoré rozhrania, je najprv potrebné vybrať z databázy informácie o všetkých rozhraniach zariadenia. Konkrétne sa jedná o názov rozhrania, VLAN-a do ktorej rozhranie má byť priradené a MAC adresa zariadenia, ktoré bude na rozhranie pripojené. Tieto údaje sú načítané do polí, ako je napríklad *@pole_mac_DB*. Práve MAC adresu pripájaného zariadenia, je potrebné najprv upraviť z tvaru, v ktorom je uložená v databáze do tvaru, ktorý prijímajú CISCO zariadenia. Následná úprava postupuje podobným spôsobom, ako nového zariadenia, v premennej *$string* sa v tomto prípade ale nachádza už konkrétny názov rozhrania (napr. FastEthernet1/0) a pridávajú sa údaje vybrané z databáze.

```
...
...
#uprava noveho zariadenia
foreach my $line (@buffer){
       $if = 0;
       $if = grep (/$string/, $line);

       if( $if == 1 ){
              $end = $buffer[ $i+1 ];
              if ($end eq "!\n"){
       #zapis prikazov
       splice @buffer, $i+1, 0,
" switchport access vlan 169
 switchport mode access
 switchport port-security mac-address sticky
 shutdown\n";
              }
       }
$i++;
}
...
...
#uprava konkretneho rozhrania
my $mac_ad = lc $pole_mac_DB[$p];
$mac_ad =~ s/[:]//g;
substr($mac_ad,4,0)='.';
substr($mac_ad,9,0)='.';
...
...
#zapise nove udaje
splice @buffer, $i+1, 0,
" switchport access vlan ".$upr_vlan."
 switchport mode access
 switchport port-security mac-address ".$mac_ad."
 no shutdown\n";
...
...
```

Funkcia na overenie *running-config* konfigurácií

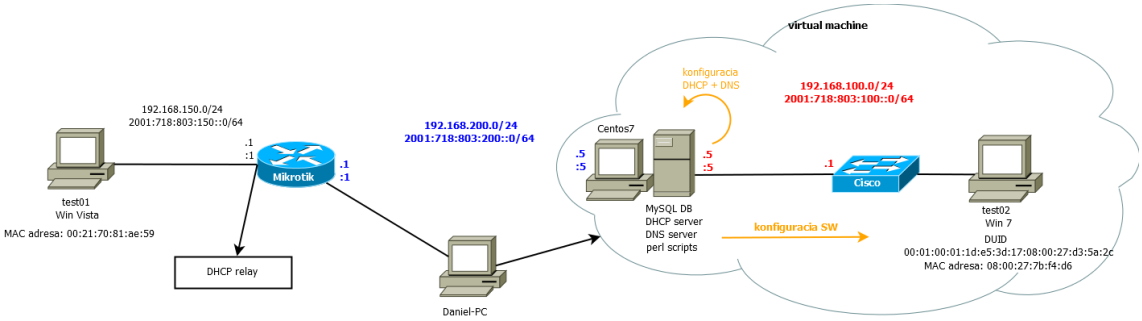
Dôležitým prvkom skriptu je funkcia *md5sum()*, ktorá kontroluje správnosť nahranej novej konfigurácie do zariadenia (diagram Obr. 28, konkrétne časť s názvom „porovnanie konfigurácií = kontrolných súčtov MD5“). Funkcia sa pripája do zariadenia a odosiela príkaz (`verify /md5 system:running-config`) pre zistenie súčtu md5 aktuálneho *running-config*-u. Výsledok je spracovaný, regulárnymi výrazmi upravený a uložený do premennej *$md5*. Do druhej premennej *$systemmd5* je uložený výsledok súčtu md5 konfigurácie uloženej na TFTP server, ktorá sa odosielala do zariadenia. Nasleduje porovnanie týchto dvoch premenných a v pozitívnom prípade, respektíve zhode, znamená, že konfigurácia bola nahratá na zariadenie v poriadku a je možné ju zapísať do *startup-config*-urácie zariadenia. V opačnom prípade sa posiela návratová hodnota skriptu a nasleduje nahratie záložných konfigurácií.

```
sub md5sum {
       my $session = Net::Telnet::Cisco->new(Host =>  $host_IP, Errmode => 're-
turn');
       if(!$session || !$session->login('cisco','cisco123')){
              return 15;
       }else{
          if ($session->enable("cisco123") ) {
             print BOLD, "prihlasenie ku SW uspesne\n\n", RESET;
             my $string = "system:running-config";
              my @buffer = $session->cmd('verify /md5 system:running-config');

              my @line = grep (/$string/, @buffer);
              my $md5 = $line[0];

             $md5 =~ s/.*["= "]//;
             print $md5;
             my $systemmd5 = qx(md5sum /tftpfiles/switch-confg | cut -f1 -d" ");
              print $systemmd5;
              if ($md5 eq $systemmd5){
                  print "\nOK\n";
                  $session->cmd("copy running-config startup-config\n\n\n");
                   return 1;
              }else{
                  print  "\nKO\n";
                   return 2;
             }
       }else {
          return 15;
             }
       }
}
```

9 Testovanie

9.1 Testovacie prostredie

Na obrázku Obr. 30 je vyobrazené testovacie prostredie. Ako bolo opisované v kapitole implementácia, LAMP server, testovacie skripty, DHCP a DNS server bežia na virtuálnom stroji s operačným systémom Linux Centos7. Ako virtuálny stroj bol taktiež vytvorený aj CISCO prepínač spolu s jednou testovacou stanicou test02 s operačným systémom Windows7. Druhá testovacia stanica test01 bol fyzický počítač nachádzajúci sa v inej časti siete. Testovanie skriptu *sw.pl* bolo taktiež uskutočnené nielen vo virtuálnom prostredí, ale aj v sieťovom laboratóriu univerzity na fyzických prvkoch, nakoľko bolo potrebné overiť rôzne verzie IOS-u CISCO prepínačov.



Obr. 30 Nákres testovacieho prostredia

9.2 Testovacie príklady

Pre otestovanie jednotlivých skriptov bol vytvorený príklad, kedy boli zadané požiadavky do databáze pre pridanie/úpravu DHCP + DNS konfigurácie pre klientov test01 a test02, znázornených na nákresu testovacieho prostredia. Avšak, aby mohol klient test02 komunikovať a vyžiadať si IP adresu od DHCP servera, musel byť najprv vytvorený ďalší požiadavok, ktorý nastaví prístupový port na zariadení CISCO podľa MAC adresy klienta test02 a priradí príslušnú VLAN.

Nasledujúce podkapitoly opisujú a znázorňujú jednotlivé situácie, kedy boli úspešne prevedené všetky požadované zmeny, ale aj situácie záporné, ktoré môžu nastať v alternatívnych scenároch. Príklady boli nasimulované chronologicky tak, ako postupuje skript, respektíve od prvotných kontrol až po konečné, úspešné úpravy.

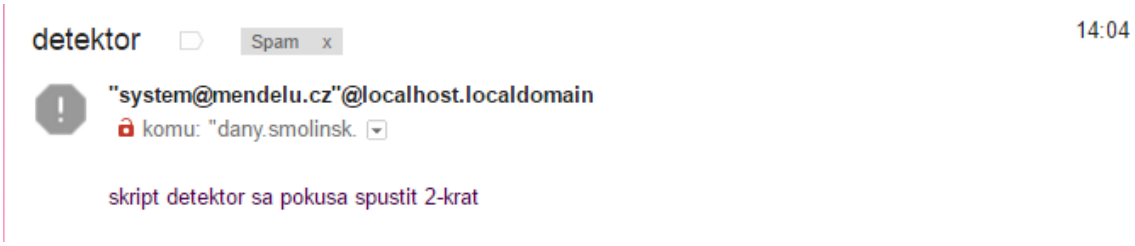
9.2.1 Testovanie detektoru

Kontrola duplikácie detektoru

Ako prvé, detektor kontroluje duplicitu v spustení seba samého. Ak zistí, že je už raz spustený, je odoslaný email administrátorovi. Obrázok Obr. 31 vyobrazuje výpis procesov s názvom *detektor.pl*. Z výpisu je možné vidieť, že jeden detektor práve beží a pri druhom spustení skriptu, nie je umožnené detektoru sa spustiť a je odoslaný email administrátorovi, viď. obrázok Obr. 32.

```
[root@localhost testovanie]#
[root@localhost testovanie]# date
Čt kvě 12 14:04:03 CEST 2016
[root@localhost testovanie]# ps -ef | grep -v grep | grep detektor.pl
root      7750  5716  0 14:04 pts/1    00:00:00 perl detektor.pl
[root@localhost testovanie]# perl detektor.pl
detektor.pl je uz raz spusteny
kontaktujem administratora
[root@localhost testovanie]# _
```

Obr. 31 Výpis procesov s názvom detektor.pl



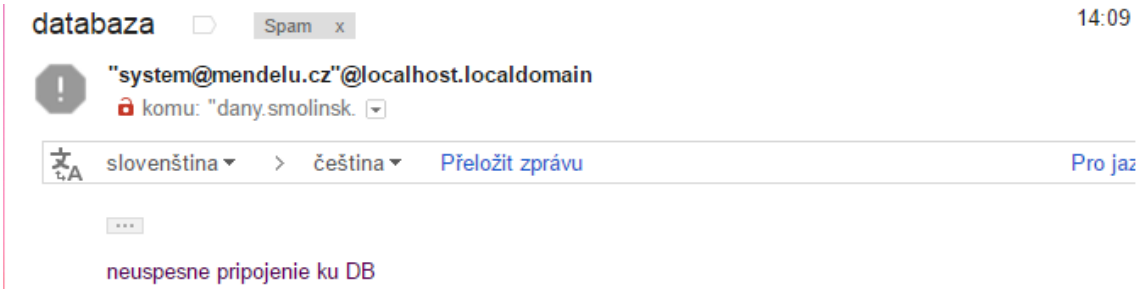
Obr. 32 Emailová komunikácia – chyba detektoru

Chyba pri pripojení ku databáze

Rovnako ako v predchádzajúcom prípade, bolo otestované taktiež pripojenie ku databáze, kedy pri chybe pripojenia bol obdŕžaný email s danou chybou.

```
[root@localhost testovanie]# date
Čt kvě 12 14:09:32 CEST 2016
[root@localhost testovanie]# perl detektor.pl
DBI connect('database=sprava-site;host=localhost','root',...) failed: Access den
ied for user 'root'@'localhost' (using password: YES) at detektor.pl line 60.
neuspesne pripojenie ku DB
kontaktujem administratora
```

Obr. 33 Chyba pripojenia ku databáze



Obr. 34 Emailová komunikácia – chyba databázy

Čítanie a úprava požiadavkou + subpožiadavkou v databáze

Obrázok Obr. 35 vyobrazuje tabuľku *change*, v ktorej sa nachádzajú dva požiadavky (*id*=6 a 7) v stave *waiting* (*state_id*=1) a jedná sa v obidvoch prípadoch o spomínanú úpravu DHCP + DNS servera (*type_id*=4). Nasledujúci obrázok Obr. 36 znázorňuje tabuľku *change* po uskutočnenej úprave a je možné vidieť zmenu. Konkrétne v stĺpcoch *dns* a *dhcp* boli zapísané hodnoty 1, signalizujúce, že úprava dobehla v poriadku a *state_id* sa zmenilo na hodnotu 3, vyjadrujúce stav OK.

id	device_id	person_id	type_id	state_id	sw	dns	dhcp	updated_at
1	3	2	4	3	0	1	1	2016-04-22 16:55:10
2	2	3	3	3	0	0	0	2016-04-24 16:07:03
3	6	3	3	4	3	0	0	2016-04-24 16:07:03
4	4	2	4	4	0	3	3	2016-04-22 16:55:10
5	4	2	4	3	0	1	1	2016-04-22 16:57:10
6	7	2	4	1	0	0	0	2016-05-05 16:33:36
7	8	2	4	1	0	0	0	2016-05-05 16:33:55

Obr. 35 Tabuľka *change* – požiadavok v stave *waiting*

id	device_id	person_id	type_id	state_id	sw	dns	dhcp	updated_at
1	3	2	4	3	0	1	1	2016-04-22 16:55:10
2	2	3	3	3	0	0	0	2016-04-24 16:07:03
3	6	3	3	4	3	0	0	2016-04-24 16:07:03
4	4	2	4	4	0	3	3	2016-04-22 16:55:10
5	4	2	4	3	0	1	1	2016-04-22 16:57:10
6	7	2	4	3	0	1	1	2016-05-05 16:35:21
7	8	2	4	3	0	1	1	2016-05-05 16:35:22

Obr. 36 Tabuľka *change* – ukončený požiadavok v stave OK

Ukončenie s chybou - RETRY

Obrázok Obr. 37 znázorňuje úpravu v databáze, ak niektorý skript, v tomto prípade konkrétne *sw.pl* skončil s chybou, respektíve nepodarilo sa nahrať novú konfiguráciu, prípadne nebolo možné sa pripojiť ku zariadeniu. V stĺpci *state_id* je hodnota 5, signalizujúca stav *retry*.

id	device_id	person_id	type_id	state_id	sw	dns	dhcp	updated_at
5	4	2	4	3	0	1	1	2016-04-22 16:57:10
6	7	2	4	3	0	1	1	2016-05-05 16:35:21
7	8	2	4	3	0	1	1	2016-05-05 16:35:22
8	8	2	3	5	2	0	0	2016-05-12 18:31:07

Obr. 37 Tabuľka *change* – požiadavok v stave *retry*

Ukončenie s chybou – ERR

Ak vráti niektorý subpožiadavok chybu ERR, znamenajúcu napríklad, že nefunguje TFTP server, v databáze sa pod daný subpožiadavok zapíše hodnota 3 a celkový požiadavok je vyhodnotený ako KO *state_id*=4.

id	device_id	person_id	type_id	state_id	sw	dns	dhcp	updated_at
5	4	2	4	3	0	1	1	2016-04-22 16:57:10
6	7	2	4	3	0	1	1	2016-05-05 16:35:21
7	8	2	4	3	0	1	1	2016-05-05 16:35:22
8	8	2	3	4	3	0	0	2016-05-12 18:58:07

Obr. 38 Tabuľka *change* – požiadavok v stave KO

Nenájdený žiadny nový požiadavok v stave *waiting* alebo *retry*

Ak sa nenachádza v databáze žiadny nový požiadavok, prípadne požiadavok na opätovné spustenie, skript *detektor.pl* je ukončený.

```
[root@localhost testovanie]# perl detektor.pl
0
Nenajdeny ziadny poziadavok v stave RETRY
Nenajdeny ziadny poziadavok v stave WAITING
[root@localhost testovanie]#
```

Obr. 39 Nevyskytujúci sa požiadavok v databáze v stave *retry*/*waiting*

9.2.2 Testovanie konfigurácie serveru DHCPv4/v6

Chyba služieb *dhcpd/dhcpd6*

Ak nastane chyba jednej zo služieb *dhcpd4/6*, ako znázorňuje napríklad obrázok Obr. 40 alebo obrázok Obr. 41, kde je vidieť, že nebeží žiadny proces

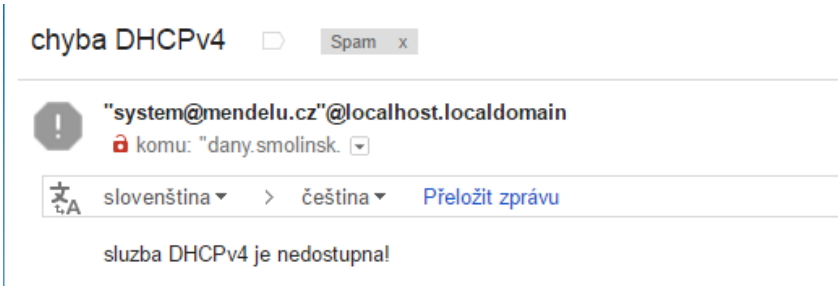
dhcpd/dhcpd6. Skript po zistení vracia príslušnú návratovú hodnotu detektoru a ten ďalej potom informuje administrátora pomocou emailu.

```
[root@localhost testovanie]# ps -ef | grep -v grep| grep dhcpd
[root@localhost testovanie]# perl detektor.pl
server DHCPv4 is down
nh= 4
3Ukončen (SIGTERM)
[root@localhost testovanie]#
```

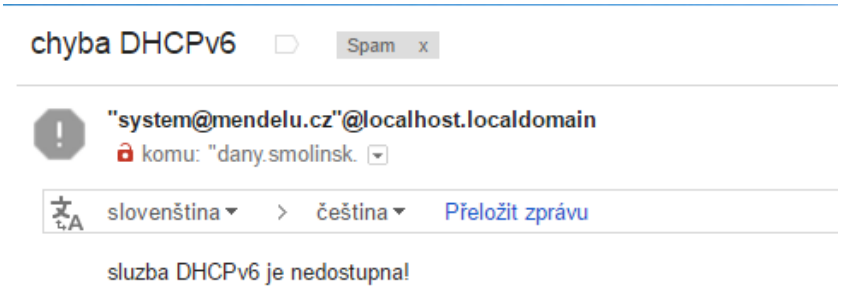
Obr. 40 Chyba služby *dhcpd*

```
[root@localhost testovanie]# ps -ef | grep -v grep| grep dhcpd
dhcpd     5645     1  0 14:37 ?        00:00:00 /usr/sbin/dhcpd -f -cf /etc/dhcp
/dhcpd.conf -user dhcpd -group dhcpd --no-pid
[root@localhost testovanie]#
[root@localhost testovanie]# perl detektor.pl
server DHCPv6 is down
nh= 5
3Ukončen (SIGTERM)
[root@localhost testovanie]#
```

Obr. 41 Chyba služby *dhcpd6*



Obr. 42 Emailová komunikácia – chyba služby *dhcpd*



Obr. 43 Emailová komunikácia – chyba služby *dhcpd6*

Zmena konfiguračného súboru *dhcpd.conf*

Po vyčítaní požiadavkou detektor zistil, že je potrebné spustiť skript *dhcp.pl* pre úpravu DHCPv4/v6 servera. Obrázok Obr. 44 znázorňuje konfiguračný súbor *dhcpd.conf* pred úpravou. Po spustení subpožiadavku *dhcp.pl* pre úpravu

DHCPv4 servera, je možné vidieť na obrázku Obr. 45 jeho výsledok. Ak vychádzame z obrázku testovacieho prostredia, jedná sa o ľavú stranu siete s IP adresou 192.168.150.0/24. Kedy skript vytiahol z databáze požadované údaje a zapísal do konfiguračného súboru *dhcpd.conf*.

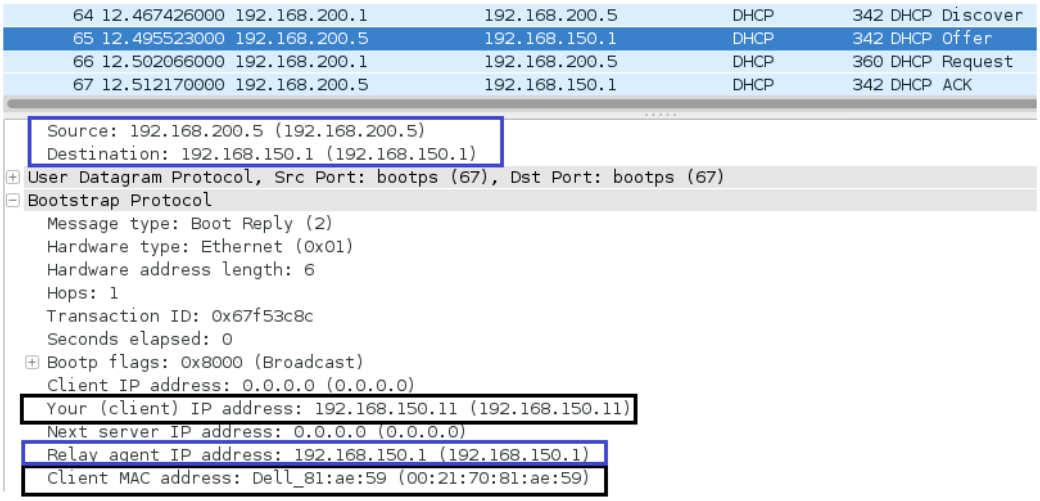
```
subnet 192.168.150.0 netmask 255.255.255.0 {
  range dynamic-bootp 192.168.150.5 192.168.150.100;
  option broadcast-address 192.168.150.255;
  option routers 192.168.150.1;

group {
}


}
```

Obr. 44 Časť konfiguračného súboru *dhcpd.conf* pred úpravou

```
subnet 192.168.150.0 netmask 255.255.255.0 {
  range dynamic-bootp 192.168.150.5 192.168.150.100;
  option broadcast-address 192.168.150.255;
  option routers 192.168.150.1;

group {
host test01 {
hardware ethernet 00:21:70:81:AE:59;
fixed-address 192.168.150.11;
}
}


}
```

Obr. 45 Časť konfiguračného súboru *dhcpd.conf* po úprave

Následný obrázok Obr. 46 znázorňuje odchyt paketov, jedná sa o situáciu, kedy klient test01 požiadal DHCP server o adresu. DHCP relay sprostredkoval jeho požiadavok a klientovi boli odoslané príslušné údaje, zapísané skriptom *dhcp.pl* do *dhcpd.conf*.



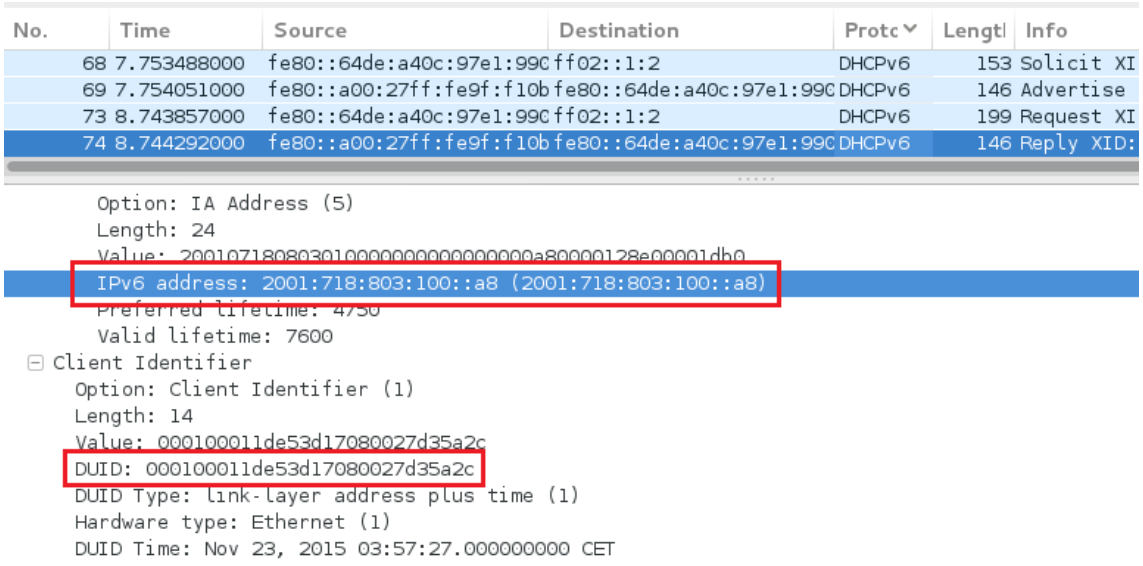
Obr. 46 Odchyt paketov – požiadavka pre IPv4 adresu klienta test01

Zmena konfiguračného súboru *dhcpd6.conf*

Pri úprave DHCPv6 servera, sa jedná o pravú stranu testovacieho prostredia, IP adresa siete 192.168.100.0/24, respektíve 2001:718:803:100:0/64. V tomto prípade na základe zapísaných údajov do konfiguračného súboru *dhcpd6.conf*, bola požadovaná adresa podľa DUID klienta odoslaná na cieľové zariadenie s názvom test02. Tomuto kroku predchádzalo ešte nastavenie na konkrétne rozhranie zariadenia CISCO príslušná MAC adresa klienta, aby mohol začať komunikovať s DHCP serverom (podkapitola 9.2.4).

```
host test02 {
host-identifier option dhcp6.client-id
00:01:00:01:1D:E5:3D:17:08:00:27:D3:5A:2C;
fixed-address6 2001:718:803:100::a8;
}
```

Obr. 47 Časť konfiguračného súboru *dhcpd6.conf* po úprave



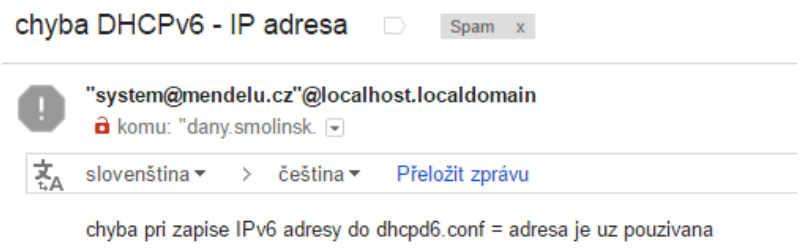
Obr. 48 Odchyt paketov – požiadavka pre IPv6 adresu klienta test02

Chyba – IP adresa je už pridelená

Ak nastala chyba, že skript zistil, že chce zapísať existujúcu IP adresu do konfiguračného súboru, ktorá je už raz pridelená inému zariadeniu, odovzdáva návratovú hodnotu 7 detektoru. V tomto prípade je informovaný taktiež administrátor.

```
[root@localhost testovanie]# perl detektor.pl
IPv4 assigned already
IPv6 is assigned already
nh= 7
3[root@localhost testovanie]#
```

Obr. 49 Chyba – IP adresa je už pridelená



Obr. 50 Emailová komunikácia – chyba pri zápise IPv4/6 adresy

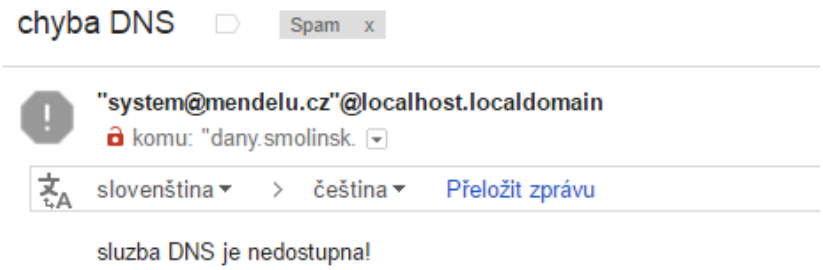
9.2.3 Testovanie konfigurácie serveru DNS

Chyba služby *dns*

Tak ako bolo v prípade chyby služieb *dhcpd/dhcpd6*, tak i v prípade chyby DNS servera nastáva ukončenie detektora a upozornenie administrátora o vzniknutej chybe.

```
[root@localhost testovanie]# perl detektor.pl
DNS server is down
nn= 12
3Ukončen (SIGTERM)
[root@localhost testovanie]#
```

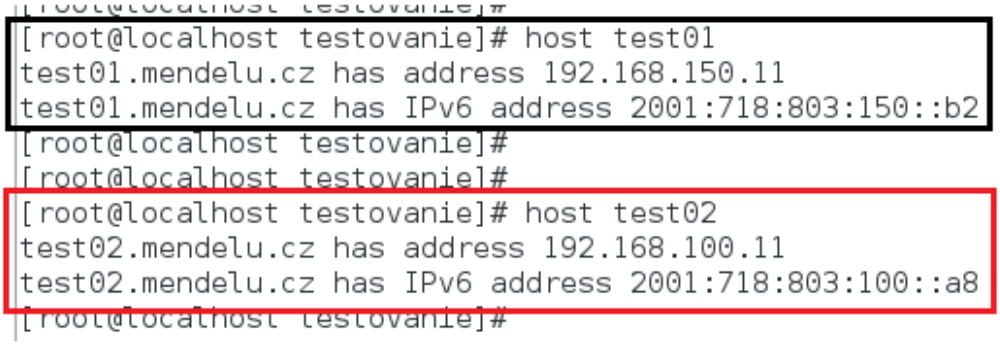
Obr. 51 Chyba DNS servera



Obr. 52 Emailová komunikácia – chyba služby *dns*

Zmena zónových súborov danej domény

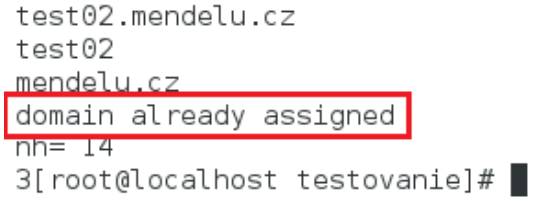
Pri testovaní služby *dns*, bolo cieľom pridanie doménových mien pre obidve testovacie stanice, t. j. test01.mendelu.cz, test02.mendelu.cz. Po úprave, obrázok Obr. 53 znázorňuje, či konfigurácia bola úspešná a preklad funguje správne.



Obr. 53 Test prekladu doménových mien klientov test01, test02

Chyba – požadované doménové meno je už pridelené

Ďalší prípad, ktorý nesmie nastať a skript *dns.pl* musí zistiť je, ak by sa jednalo o zápis doménového mena, ktoré je už používané. Skript vráti chybu a posiela návratovú hodnotu detektoru.



Obr. 54 Chyba – doménové meno je už pridelené

9.2.4 Testovanie konfigurácie SW

Konfigurácia konkrétneho sieťového rozhrania zariadenia

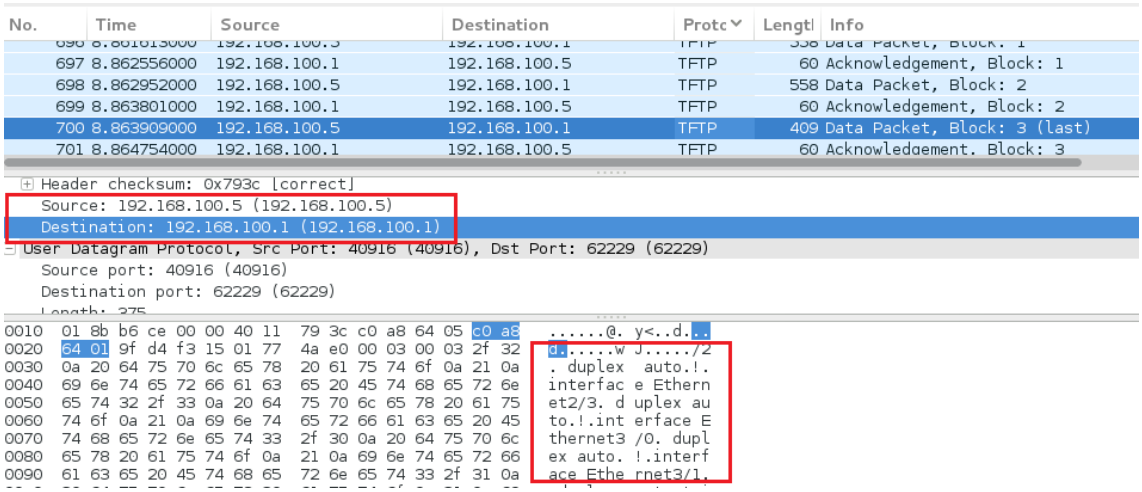
Obrázok Obr. 55 znázorňuje výpis konzoly testovacieho zariadenia CISCO. Jedná sa o zmenu konkrétneho rozhrania *interface Fa1/0,* na ktoré je potrebné nahrať konfiguráciu, aby mohol klient test02 komunikovať v sieti. Prvý červený rámik poukazuje, že sa nič nenachádza na rozhraní. Modrý rámik zvýrazňuje výpis zariadenia, že prišlo ku konfigurácii. Následný výpis v druhom červenom rámiku znázorňuje, že na rozhranie *Fa1/0* bola nahratá požadovaná konfigurácia.

```
Switch#sh running-config interface ethernet 1/0
Building configuration...

Current configuration : 42 bytes
!
interface Ethernet1/0
 duplex auto
end

Switch#
*May 12 17:06:23.325: Rollback:Acquired Configuration lock.
Switch#sh running-config interface ethernet 1/0
Building configuration...

Current configuration : 147 bytes
!
interface Ethernet1/0
 switchport access vlan 100
 switchport mode access
 switchport port-security mac-address 0800.277b.f4d6
 duplex auto
end

Switch#
```

Obr. 55 Výpis konzoly SW pri konfigurácii zariadenia

TFTP prenos

Obrázok Obr. 56 vyobrazuje odchyt paketov pri posielaní konfigurácie do cieľového zariadenia. Je možné vidieť cieľovú adresu 192.168.100.1, ktorá predstavuje manažment IP na CISCO zariadení a zdrojovú, v našom prípade stanicu Centos7. Nakoľko sa nejedná o šifrovaný prenos, je možné taktiež vidieť údaje, ktoré sú prenášané v paketoch.



Obr. 56 Odchyt paketov – TFTP komunikácia

Problém s pripojením ku SW

V prípade, ak nastane problém s pripojením ku zariadeniu SW (viď. Obr. 57), je vrátená hodnota detektoru 15 (chyba pripojenia SW, viď. Tab. 10) a celkový požiadavok v databáze vyhodnotený do stavu *retry*. Táto situácia, zmena požiadavku, je vyobrazená a opisovaná vyššie.

```
[root@localhost testovanie]# perl detektor.pl
Nemozno sa pripojit ku zariadeniu SW
nh = 15
[root@localhost testovanie]#
```

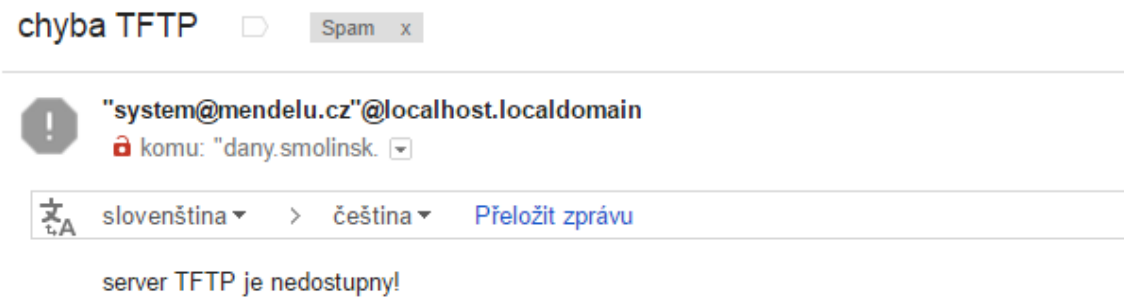
Obr. 57 Chyba – pripojenie ku zariadeniu SW

Problém s TFTP serverom

Podobný prípad nastáva pri probléme napríklad s TFTP serverom, kedy je vrátená návratová hodnota detektoru 17, a nakoľko sa jedná o ERR chybu, detektor je ukončený a administrátor informovaný o tejto chybe.

```
[root@localhost testovanie]# perl detektor.pl
prihlasenie ku SW uspesne

TFTP server je zastaveny
nh = 17
3Ukončen (SIGTERM)
[root@localhost testovanie]#
```

Obr. 58 Chyba – spadnutý TFTP server



Obr. 59 Emailová komunikácia – chyba TFTP

Nahratie záložnej konfigurácie pre zariadenie SW

Ak nastal prípad, kedy sa nezhodoval kontrolný súčet md5 konfigurácií na zariadení a na TFTP serveri, je potrebné nahrať záložnú konfiguráciu. Tento prípad je znázornený na obrázku Obr. 60, kedy je možné vidieť v modrom rámiku, že je odoslaný dvakrát príkaz do zariadenia pre nahratie *running-config*-u. V červených rámikoch je vidieť, že konfigurácia na rozhraní sa nezmenila.

```
Switch#sh running-config interface ethernet 1/0
Building configuration...

Current configuration : 42 bytes
!
interface Ethernet1/0
 duplex auto
end

Switch#
*May 12 18:30:57.265: Rollback:Acquired Configuration lock.
Switch#
*May 12 18:31:01.293: Rollback:Acquired Configuration lock.
Switch#
Switch#sh running-config interface ethernet 1/0
Building configuration...

Current configuration : 42 bytes
!
interface Ethernet1/0
 duplex auto
end

Switch#
```

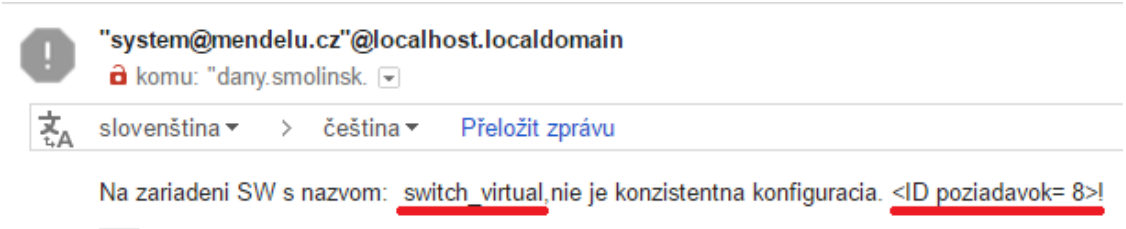
Obr. 60 Výpis konzoly SW pri nahratí záložnej konfigurácie

Nekonzistentná konfigurácia na SW

Ak nastane situácia, kedy ani záložná konfigurácia nebola správne nahratá na cieľové zariadenie, je odoslaná príslušná návratová hodnota detektoru o chybe. Na obrázku Obr. 61, je možné vidieť kontrolné súčty ako v prvom prípade, kedy sa skript *sw.pl* snaží nahrať novú, upravenú konfiguráciu, ale nebola úspešne nahratá, tak i v druhom prípade, kedy ani záložná konfigurácia nemá zhodný kontrolný súčet. V tomto prípade je nutné upozorniť na vzniknutú chybu taktiež administrátora prostredníctvom emailu, v ktorom sa dozvie presne názov zariadenia a *id* požiadavku, na ktorom vznikla daná chyba (viď. Obr. 62).

```
prihlasenie ku SW uspesne

24e7ce3008cba999e2a31ac0cc362912
f03663cfd52fc99ab190846da86c8af0

KO
prihlasenie ku SW uspesne

24e7ce3008cba999e2a31ac0cc362912
5a3ff8d8ab0e14a28be1101fe93a7079

KO

nh= 16
[root@localhost testovanie]#
```

Obr. 61 Chyba – nekonzistentná konfigurácia/nezhodné md5 súčty



Obr. 62 Emailová komunikácia – nekonzistentná konfigurácia na SW

10 Diskusia

Klady a zápory riešenia

Vzniknutý backend systému prináša hlavný prínos v automatickom generovaní konfigurácií pre DHCPv4/v6 server, DNS server a prístupové prepínače, čím uľahčuje prácu administrátorom siete MENDELU. Umožňuje spravovať komplexnú sieť založenú na adresách nielen typu IPv4 ale aj IPv6. Publikačná rešerš, zahrnutá v diplomovej práci (kapitola 6.) pojednáva o existujúcich systémoch a prácach, no žiadna z nich, ako je v závere rešerše opisované, nezahrňuje všetky spomínané súčasti vytváraného backend-u nového systému. Súčasne nové riešenie v sebe zahŕňa a rieši prakticky všetky alternatívy nepriaznivých situácií, ktoré môžu nastať pri konfigurácii jednotlivých spomínaných serverov. V tomto zmysle, o každej nepriaznivej situácii je administrátor informovaný a vie čo sa v sieti deje. Konfigurácia CISCO prepínačov je riešená nielen pre zmeny súčasných zariadení, ale aj pre budúce, nové prepínače, ktoré budú pridávané do siete MENDELU.

Konfigurácia týchto zariadení je avšak obmedzená len na prepínače spoločnosti CISCO a práve komunikácia s týmito zariadeniami je doposiaľ riešená nešifrovaným protokolom. Súčasné skripty sú otestované na rôznych verziách IOS-u v sieťovom laboratóriu MENDELU, no nie je vylúčené, že s príchodom nových verzií systémov, budú skripty kompatibilné. Podobný problém nastával už pri spomínanom výbere komunikácie so zariadeniami, kedy SNMP protokol má problémy práve v podobe rôznych verzií IOS-u.

Teoreticky, vytvorený backend je možné nasadiť v rôznych počítačových sieťach, nakoľko konfiguruje základné časti siete pre komunikáciu klientov.

Ďalší vývoj a zlepšenie

Vytvorený backend systému je len prvou verziou, ktorú čakajú niektoré zlepšenia pred nasadením v ostrej sieti MENDELU. V prvom rade musí byť dokončený vývoj frontend-u, aby bolo pohodlné ovládať pozadie systému.

Práve pre ďalší vývoj bol kladený najväčší dôraz na návrh a zrozumiteľnosť jednotlivých vývojových diagramov, scenárov či alternatívnych scenárov.

Návrhom pre zlepšenie je už spomínaná komunikácia so sieťovými zariadeniami CISCO, ktorá by mohla prejsť do šifrovanej podoby pomocou modulov SSH. V prípade ďalšieho rozširovania by mohol byť navrhnutý monitorovací systém zariadení, pre prípadnú kontrolu vyťaženia prepínačov z hľadiska či hardwarového alebo sieťového. S príchodom nových verzií IOS-ov rozšíriť a optimalizovať komunikáciu s týmito zariadeniami, či so zariadeniami iných spoločností.

11 Záver

Úlohou diplomovej práce bolo vytvorenie backend-u nového systému pre správu siete MENDELU. Po analýze súčasnej siete a stanovení špecifických požiadavkou administrátorom univerzitnej siete Mendelovej univerzity je navrhnutý, implementovaný a otestovaný nový návrh.

Kapitola špecifikácia požiadavkou, ktorá definuje celkovo 5 hlavných funkcií backend-u systému, spolu s kapitolou analýza súčasného systému vytvárali podklad pre vývoj nového systému. Stali sa východiskom nie len pre celkový návrh ale aj pre následnú implementáciu. Analýza poukázala na nedostatky súčasného systému a hlavne taktiež to, že súčasný systém nepodporuje IP adresy verzie 6, čo bolo hlavným impulzom zadania tohto projektu.

Kapitola návrh bola veľmi dôležitou časťou, nakoľko táto diplomová práca má byť podkladom pre celkový projekt, ktorý bude v sebe zahŕňať komplexný, nový systém správy siete MENDELU. Boli vypracované jednotlivé scenáre, či alternatívne scenáre možných situácií, ktoré sú spracované a opisované taktiež v podobe vývojových diagramov.

Implementácia začínala naštudovaním a zoznámením sa s programovým vybavením perl počas celého roka. Bolo potrebné sa naučiť, pracovať s týmto jazykom a boli vytvorené celkovo štyri skripty. Tieto skripty spĺňajú všetky požiadavky stanovené administrátorom univerzitnej siete MENDELU.

Pre overenie funkčnosti boli nasimulované a otestované naprogramované skripty, kde ich výsledky je možné vidieť na príkladoch v časti testovanie. Testy zahŕňali ako ideálny priebeh a úpravy tak i negatívne závery a možné chyby, ktoré môžu nastať.

V súčasnej dobe je vytvorený backend systému pripravený pre ďalší vývoj a nájde uplatnenie nielen v univerzitnej sieti MENDELU, ale pri prípadnom rozšírení je možné, že bude môcť byť nasadený i do iných sietí.

12 Literatúra

[1] KUROSE, J. F. a ROSS, K. W. Počítačové sítě. Brno: Computer Press, 2014. 622 s. ISBN 978-80-251-3825-0.

[2] LAMMLE, T. CCNA: výukový průvodce přípravou na zkoušku 640-802. Brno: Computer Press, 2010. 928 s. ISBN 978-80-251-2359-1.

[3] Cisco.com. Unified Access Network Design and Considerations. [online]. [cit. 2016-02-09]. Dostupné z: <http://www.cisco.com/c/en/us/td/docs/solutions/Enterprise/Borderless_Networks/Unified_Access/Unified_Access_Book/UA_Design.html>

[4] Ciscopress.com. Hierarchical Network Design. [online]. [cit. 2016-02-09]. Dostupné z: <http://www.ciscopress.com/articles/article.asp?p=2202410&seqNum=4>

[5] GILMORE, W. J. Beginning PHP and MySQL: from novice to professional. 4th ed. New York: Apress, 2010. 824 s. ISBN 978-143-0231-141.

[6] MATOUŠEK, P. Síťové aplikace a jejich architektura. Brno: VUTIUM, 2014. 396 s. ISBN 978-80-214-3766-1.

[7] Cisco.com. DHCPv6 Based IPv6 Access Services. [online]. 2011 [cit. 2016-02-10]. Dostupné z: <http://www.cisco.com/c/en/us/products/collateral/ios-nx-os-software/enterprise-ipv6-solution/whitepaper_c11-689821.html>

[8] BAKER, D. Control4.com. DHCP VS. STATIC IP—WHICH IS BETTER?. [online]. 2012 [cit. 2016-02-15]. Dostupné z: <http://www.control4.com/blog/2012/11/dhcp-vs-static-ip-which-is-better>

[9] Tomshardware.co.uk. What Is the Difference between DHCP Reservation and DHCP Exclusion?. [online]. 2013 [cit. 2016-02-15]. Dostupné z: <http://www.tomshardware.co.uk/faq/id-1932491/difference-dhcp-reservation-dhcp-exclusion.html>

[10] SATRAPA, P. IPv6: internetový protokol verze 6. 3., aktualiz. a dopl. vyd. Praha: CZ.NIC, 2011. 407 s. ISBN 978-80-904248-4-5.

[11] Standards.ieee.org. Guidelines for 64-bit Global Identifier (EUI-64). [online]. [cit. 2016-03-03]. Dostupné z:

<https://standards.ieee.org/develop/regauth/tut/eui64.pdf>

[12] IETF. RFC 3315. 2003. Dostupné z:
< https://www.ietf.org/rfc/rfc3315.txt>

[13] MORR, D. Wikispaces.psu.edu. IPv6 DNS. [online]. 2009 [cit. 2016-03-03]. Dostupné z:
<https://wikispaces.psu.edu/display/ipv6/IPv6+DNS>

[14] PUŽMANOVÁ, R. TCP/IP v kostce. 2., upr. a rozš. vyd. České Budějovice: Kopp, 2009. 619 s. ISBN 978-80-7232-388-3.

[15] BOUŠKA, P. samuraj-cz.com. IPv6 DNS. [online]. 2011 [cit. 2016-03-12]. Dostupné z:
<http://www.samuraj-cz.com/clanek/vpn-1-ipsec-vpn-a-cisco/>

[16] Cisco.com. Configuring EtherChannels. [online]. [cit. 2016-03-12]. Dostupné z:
<http://www.cisco.com/c/en/us/td/docs/switches/lan/catalyst3850/software/release/3se/layer2/configuration_guide/b_lay2_3se_3850_cg/b_lay2_3se_3850_cg_chapter_0101.html>

[17] Advanxer.com. EtherChannel vs LACP vs PAgP. [online]. 2013 [cit. 2016-03-15]. Dostupné z:
<https://advanxer.com/blog/2013/08/etherchannel-vs-lacp-vs-pagp/>

[18] Ciscopress.com. Basic Switching Concepts and Configuration. [online]. 2014 [cit. 2016-03-22]. Dostupné z:
<http://www.ciscopress.com/articles/article.asp?p=2181836&seqNum=4>

[19] Cisco.com. Configuring SSH and Telnet. [online]. [cit. 2016-03-22]. Dostupné z:
<http://www.cisco.com/c/en/us/td/docs/switches/datacenter/sw/4_1/nx-os/security/configuration/guide/sec_nx-os-cfg/sec_ssh.html>

[20] Cisco.com. Configuring Access and Trunk Interfaces. [online]. [cit. 2016-03-22]. Dostupné z:
<http://www.cisco.com/c/en/us/td/docs/switches/datacenter/nexus5000/sw/configuration/guide/cli/CLIConfigurationGuide/AccessTrunk.html>

[21] SOSINSKY, B. A. Mistrovství - počítačové sítě: [vše, co potřebujete vědět o správě sítí]. Brno: Computer Press, 2010. 840 s. ISBN 978-80-251-3363-7.

[22] Cisco.com. Port Security. [online]. [cit. 2016-03-22]. Dostupné z: <http://www.cisco.com/c/en/us/td/docs/switches/lan/catalyst6500/ios/12-2SX/configuration/guide/book/port_sec.html>

[23] DAŘENA, F. Myslíme v jazyku PERL. Praha: Grada, 2005. 700 s. ISBN 80-247-1147-8.

[24] SATRAPA, P. Perl pro zelenáče. 2. vydání. Praha: Neocortex, 2001. 224 s. ISBN 80-86330-02-8.

[25] ŠEDÁ, J. a TYLLICH, M. Management sítě. Dokumentace UIS MENDELU. Svazek 21. Mendelova univerzita v Brně. 2016.

[26] Freenetis - open source informační systém pro administraci neziskové sítě. [online] [cit. 2015-11-17]. Dostupné z: <http://www.freenetis.org/>

[27] Foreman. [online] [cit. 2015-11-28]. Dostupné z: <http://theforeman.org/>

[28] GORWITS, O. Netdisco - An open source web-based network management tool. [online] [cit. 2015-11-30]. Dostupné z: <http://netdisco.org>

[29] NET service solutions, s.r.o. ISPadmin - informační a administrační systém pro správu sítě. [online] [cit. 2015-11-17]. Dostupné z: <http://www.ispadmin.eu/cz/>

[30] GUTVEIS, M. CIBS - Clever ISP Business Service. [online] [cit. 2015-11-28]. Dostupné z: <http://www.cibs.cz/>

[31] Univerzita Tomáše Bati ve Zlíně. Knihovna. [online] [cit. 2015-11-7]. Dostupné z: <https://digilib.k.utb.cz/>

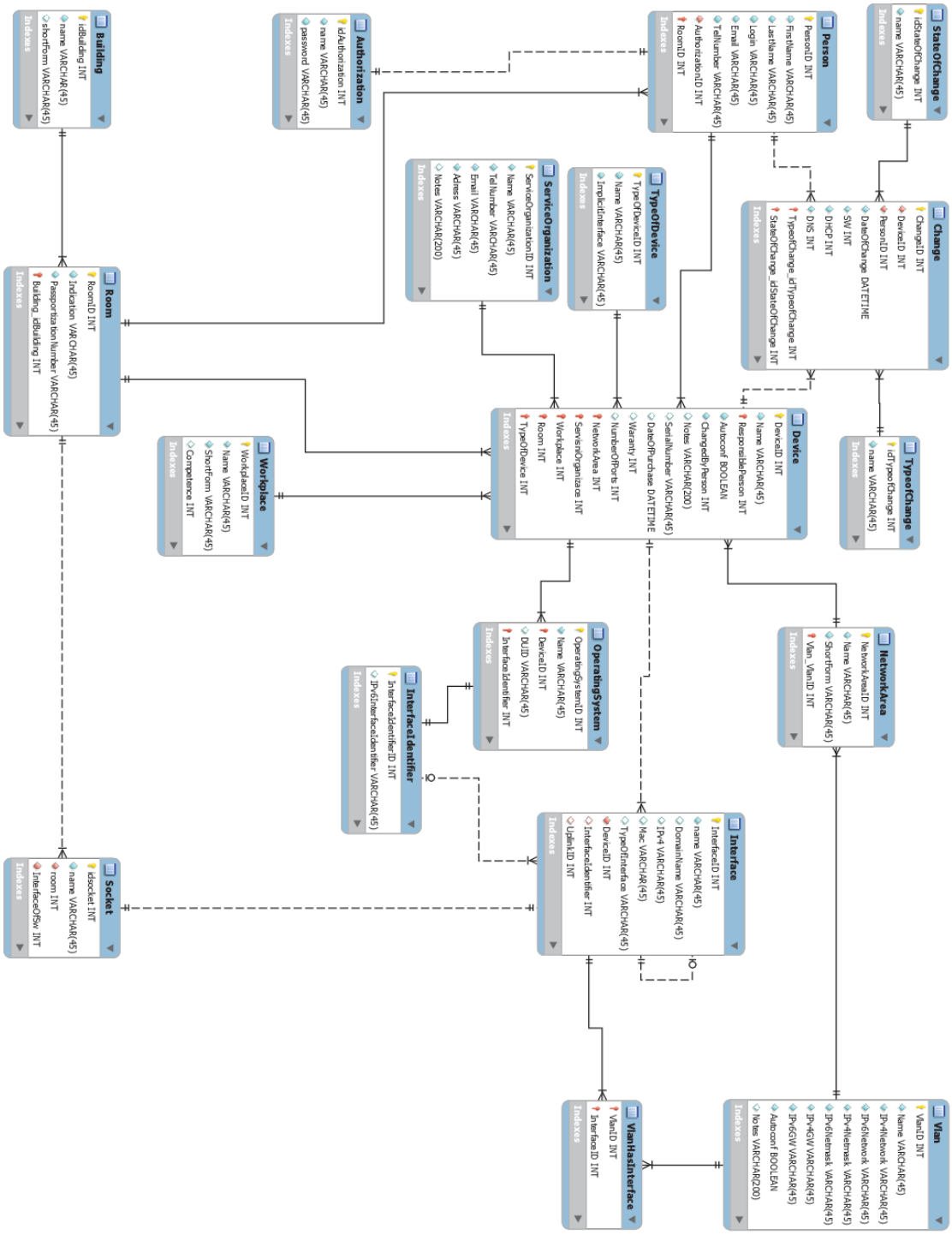
[32] Informační systém Masarykovy Univerzity. Absolventi a závěrečné práce. [online] [cit. 2015-11-7]. Dostupné z: <http://is.muni.cz/thesis/>

[33] Vysoké učení technické v Brně. [online] [cit. 2015-11-8]. Dostupné z:

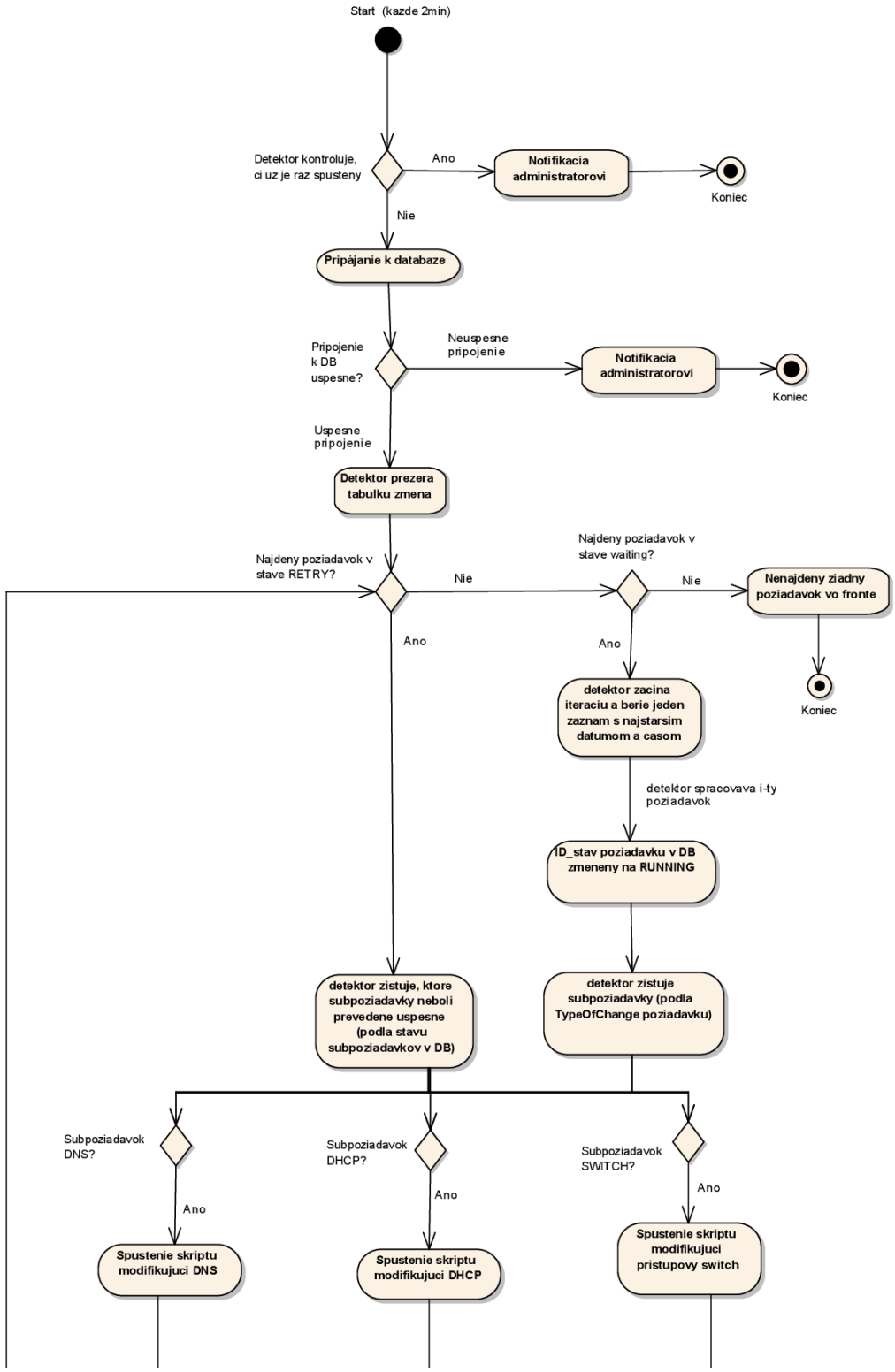
<https://www.vutbr.cz/>
[34] SÜSS, M. Případová studie – Získaní dat z Cisco zařízení. Mendelova univerzita v Brně. 2016.

Prílohy

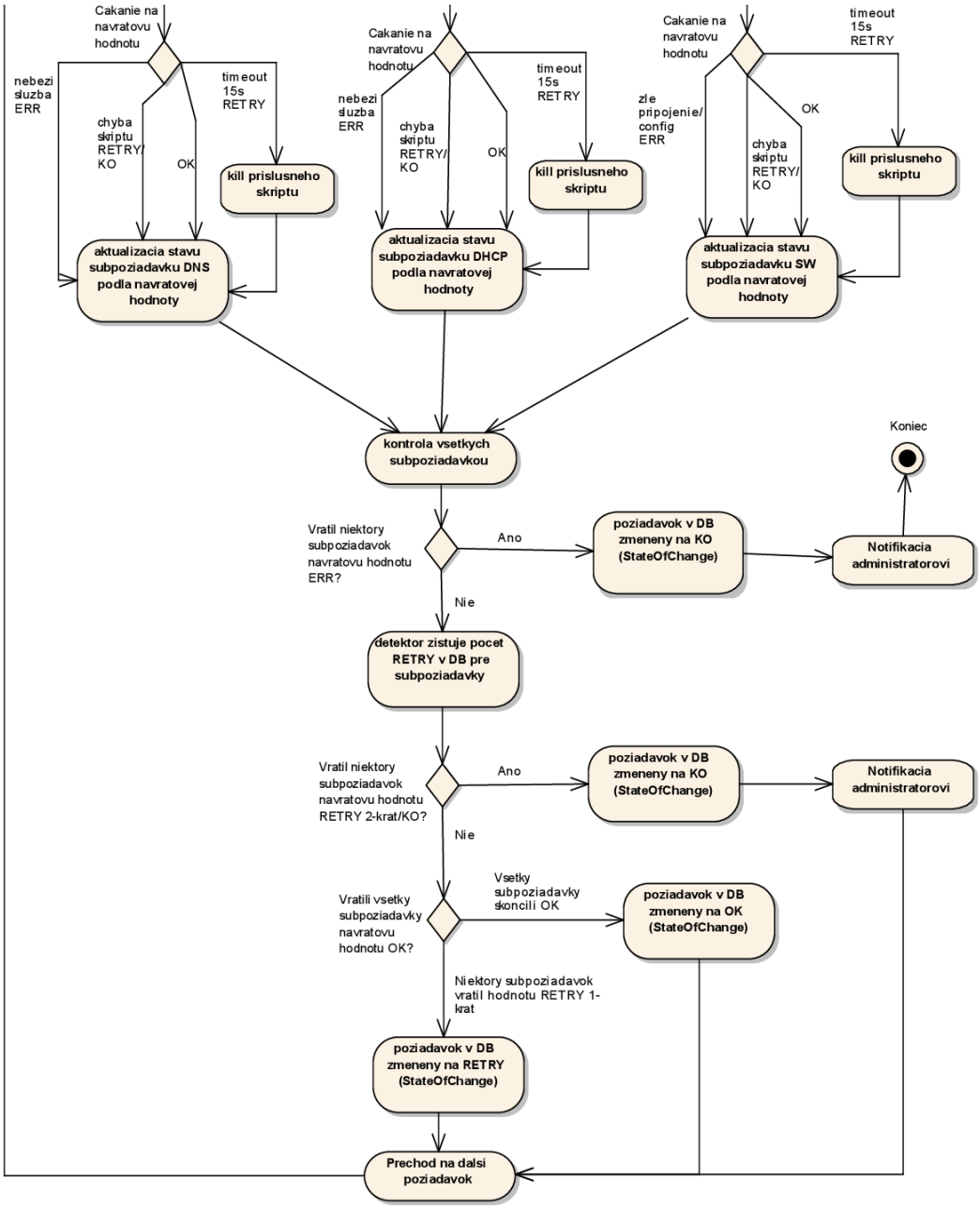
A - Entity-relationship diagram (ERD)



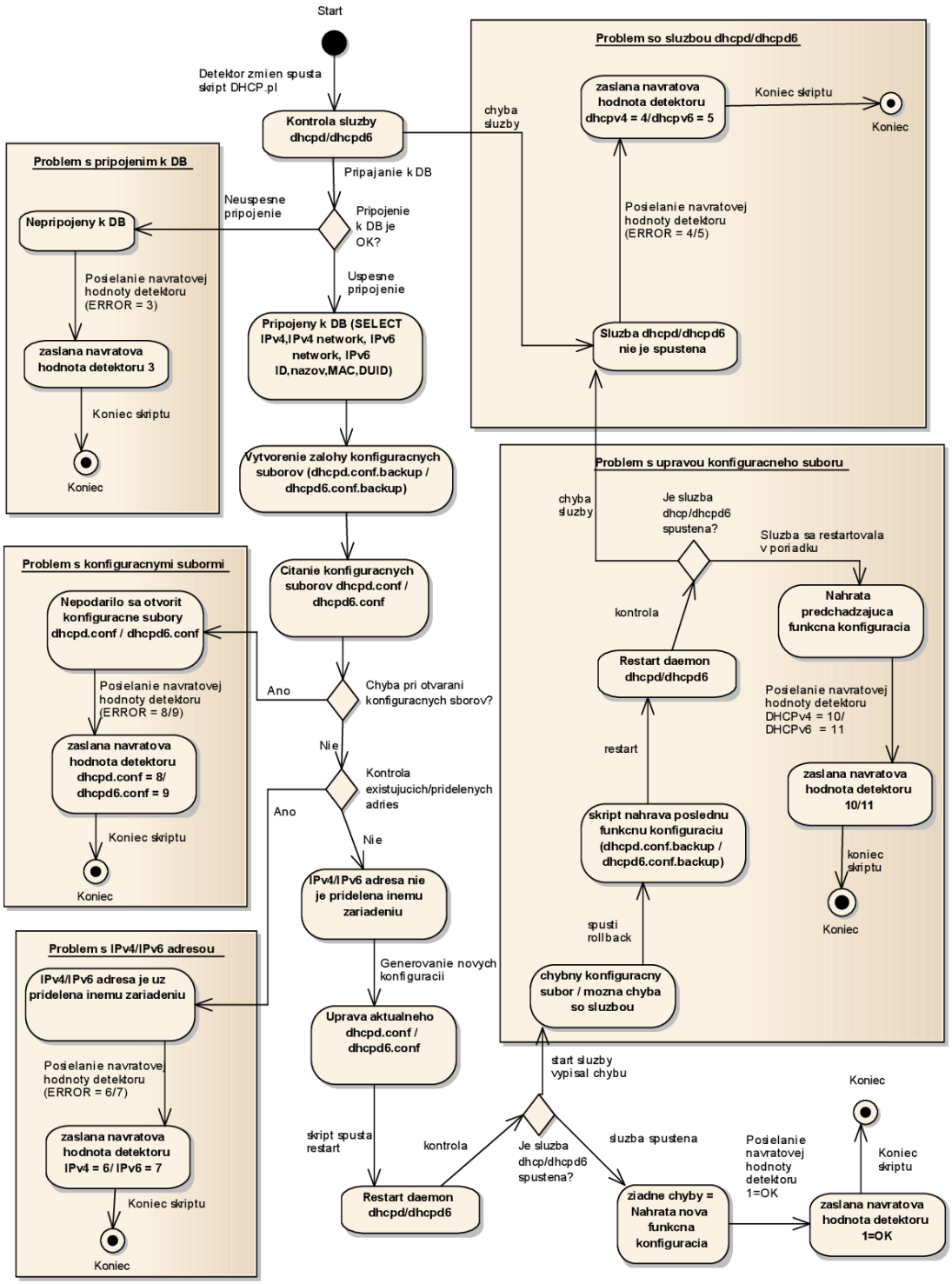
B – Vývojový diagram detektor.pl (časť 1)



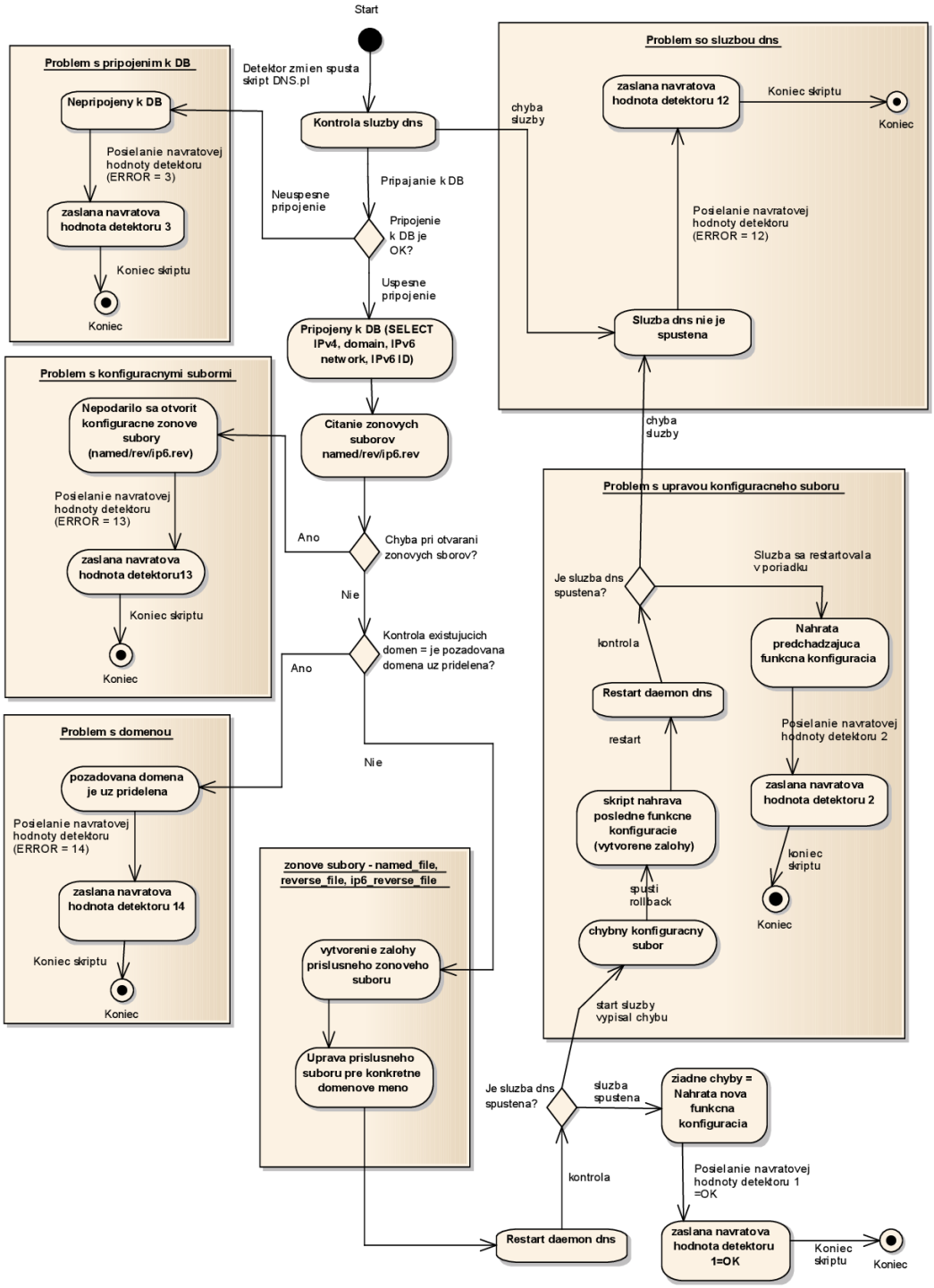
B – Vývojový diagram detektor.pl (časť 2)



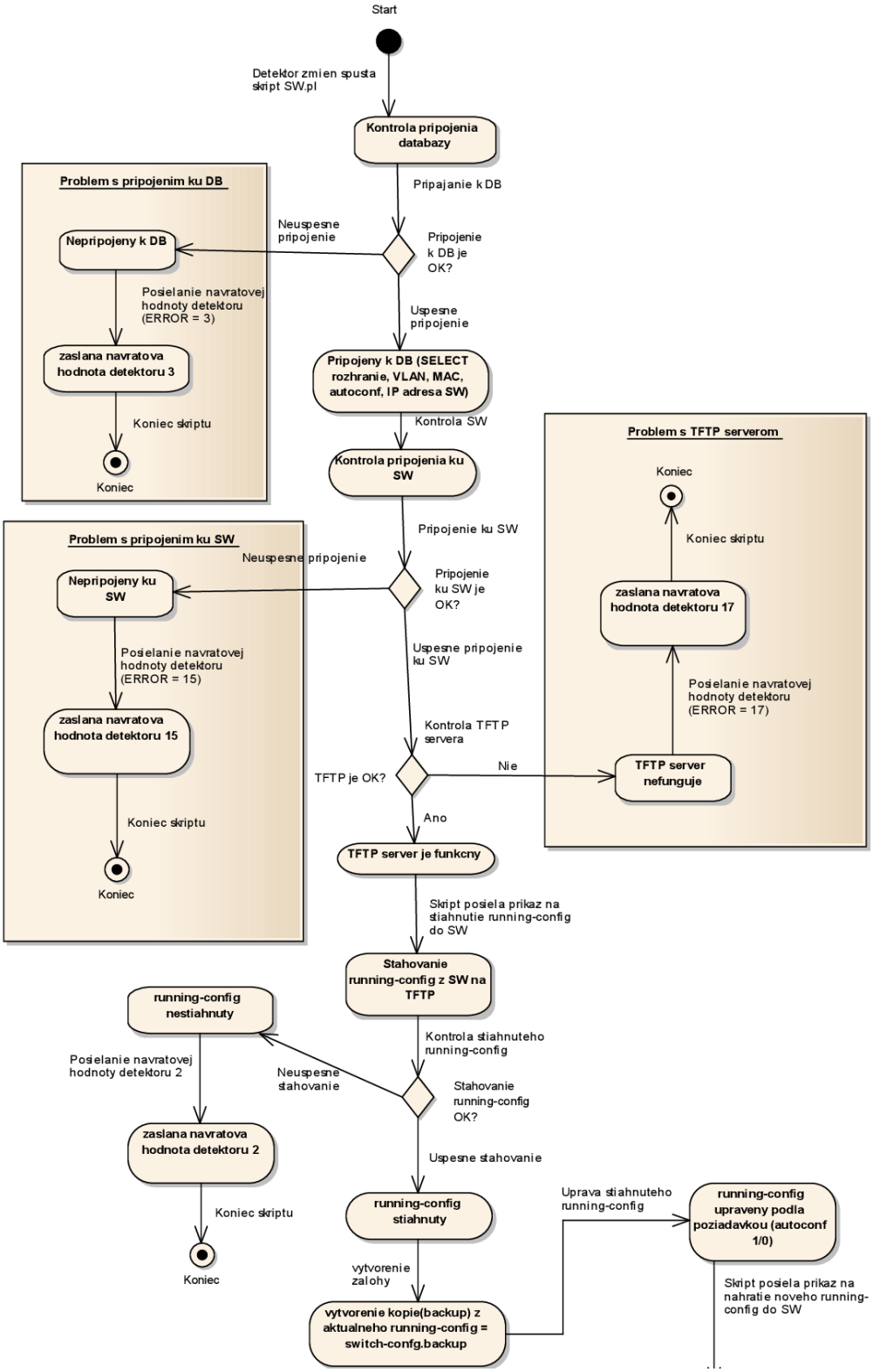
C – Vývojový diagram dhcp.pl



D – Vývojový diagram dns.pl



E – Vývojový diagram sw.pl (časť 1)



E – Vývojový diagram sw.pl (časť 2)