Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Diplomová práce

Vizualizace Spanning Tree Protocol

Plzeň 2015 Vladimír Mandák

Prohlášení

Prohlašuji, že diplomovou práci jsem vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 26. června 2015 Vladimír Mandák .....

Poděkování

Na tomto místě bych rád poděkoval vedoucímu práce Ing. Michalu Petrovičovi za cenné rady, připomínky a pomoc při vedení práce.

Zároveň bych chtěl poděkovat rodině za silné zázemí a podporu po celou dobu studia. Dále pak všem přátelům a kamarádům, kteří mě během studia podporovali.

Abstract

In this thesis I am dealing with gathering data about state of Spanning Tree protocol from Cisco network devices and subsequent visualization of obtained information.

At first of all I have to become familiar with network devices produced by Cisco Systems. Theese devices are providing basic text user interface of operating systems Catalyst IOS and NX OS. It is possible to connect to device localy via console serial interface or remotely using unsecured Telnet protocol eventually secure SSH protocol.

I have created a software providing an intuitive HTML5 user interface, where user creates a map of physical computer network. Interconnection between devices allows jsPlumb library. It is necessary to set IP address and connected ports to each device from wich user want to retrieve data. Maps created by user, can be stored in a database.

A script, which is gathering data from network devices is written in Perl language. Obtaining data from Cisco devices proceeds of periodical running a script. Script loads IP address of all deiveces and ports from database. Script remotely connects to device with secured SSH protocol and sends a command, which checks status of the Spanning Tree protocol. Obtained data are processed and saved in database. User can view obtained data in the web application.

Functionality was verified in an university network WEBnet.

Keywords: STP, SNMP, switch, router, Catalyst IOS, NX OS, Perl, MySQL, visualization, HTML5, PHP, JavaScript, jsPlumb, jQuery, map, draggable, UIkit

Abstrakt

V této práci se zabývám získáváním dat o stavu Spanning Tree protokolu ze síťových zařízení firmy Cisco a následnou vizualizací získaných informací.

Nejdříve jsem se musel seznámit se síťovými zařízeními firmy Cisco. Ty poskytují základní textové uživatelské rozhraní operačního systému Catalyst IOS a NX OS. K zařízení se lze připojit lokálně pomocí konzole sériovou linkou nebo vzdáleně s použitím nezabezpečeného protokolu Telnet, případně zabezpečeným protokolem SSH.

Dále jsem vytvořil programové vybavení poskytující intuitivní HTML5 uživatelské rozhraní, kde uživatel vytvoří mapu fyzické počítačové sítě. Propojování jednotlivých zařízení umožňuje knihovna jsPlumb. Každému zařízení, ze kterého chce uživatel získávat data, je potřeba nastavit IP adresu a jména zapojených portů. Vytvořené mapy může uživatel ukládat do databáze.

Skript, který získává data ze síťových zařízení je napsaný v jazyce Perl. Periodickým spouštěním skriptu získávám aktuální informace o stavu Spanning Tree protokolu. Skript načte z databáze IP adresy všech zařízení a připojené porty. Zabezpečeným SSH protokolem se skript vzdáleně připojí k zařízení a odešle příkaz, kterým zjistí stav STP. Získaná data zpracuje a uloží do databáze. Získaná data může uživatel sledovat ve webové aplikaci.

Funkčnost byla ověřena v prostředí univerzitní sítě WEBnet.

Klíčová slova: STP, SNMP, přepínač, směrovač, Catalyst IOS, NX OS, Perl, MySQL, vizualizace, HTML5, PHP, JavaScript, jQuery, jsPlumb, map, draggable, UIkit

Obsah

1 Úvod

Počítačové sítě zažívají v poslední době velmi rychlý vývoj. Postupně pronikají do různých oblastí lidského života. S nárůstem požadavků na rychlost a škálovatelnost současně roste i důraz na dostupnost a stabilitu sítě. Pro správce sítě z toho plyne povinnost zajistit rozumnou míru redundance na fyzické a linkové vrstvě ISO/OSI modelu. Přidávání dalších spojení mezi přepínači tvoří komunikační smyčky. Následkem je zahlcení sítě. Problémy se zahlcením sítě řeší protokol STP. Jeho úkolem je hledat v síti smyčky a odstranit je. Vizualizace stavu sítě pomáhá síťovému administrátorovi včas odhalit možné problémy a následně je řešit.

V první části se zabývám teorií. Uvádím základními pojmy z oblasti počítačových sítí. Jelikož se tato práce zabývá výhradně technologiemi firmy Cisco, je uveden i stručný přehled charakteristických vlastností jednotlivých produktových řad síťových zařízení. Značná část teorie je věnována seznámení se s protokolem Spanning Tree. Jelikož se jedná o stěžejní protokole této práce, je jeho popis oproti jiným protokolům rozsáhlejší kvůli správné implementaci konečné vizualizace ve webové aplikaci. Popsány jsou také úzce související pojmy jako VLAN, trunk, technologie EtherChannel a VTP protokol.

Druhá část práce se zabývá analýzou. V této části jsem věnoval pozornost některým konkurenčním produktům, které však nenabízejí požadovanou funkcionalitu. Dále je popsáno prostředí univerzitní sítě WEBnet, ve které bude výsledná aplikace nasazena. Náležitě jsem také analyzoval způsoby získávání informací ze síťových zařízení společně s úvahou nad souvisejícími výhodami a nevýhodami. Pro vytvoření návrhu databáze bylo zapotřebí si ujasnit vztah mezi získávanými daty a jejich strukturu. Vytvoření programového vybavení předchází poznání technologií a nástrojů použitelných pro jádro aplikace a webové rozhraní. Výsledkem je navržení sytému sledování stavů STP protokolu a jejich vizualizace.

Ve třetí části je popsána realizace programového vybavení. Zde jsem popsal použité programové konstrukce skriptu, pro získávání všech potřebných informací ze síťových zařízení. Dále je popsána struktura webové aplikace. Důležitou částí je popis zabezpečení serveru a přístupu k síťovým zařízením. Součástí je uživatelská příručka a inspirace pro další rozšíření.

Část I

Teorie

2 Síťová zařízení Cisco

2.1 Požadavky na síť

S rozvojem informačních technologií dochází k rozšiřování počítačových sítí a jsou na ně kladeny stále vyšší nároky. Těmi nejpodstatnějšími jsou:

Kapacita

Požadavek na přenosy velkých objemů dat - streamování videa, zvuku, výsledky a data pro distribuované výpočty.

Spolehlivost

Zajištění dostupnosti služeb v případě výpadku některé linky nebo poruchy aktivního prvku na cestě mezi klientem a serverem.

Bezpečnost

Důraz na autentizaci přístupu ke službám, síťovým prvkům a koncovým zařízením. Využívání silných šifrovacích algoritmů.

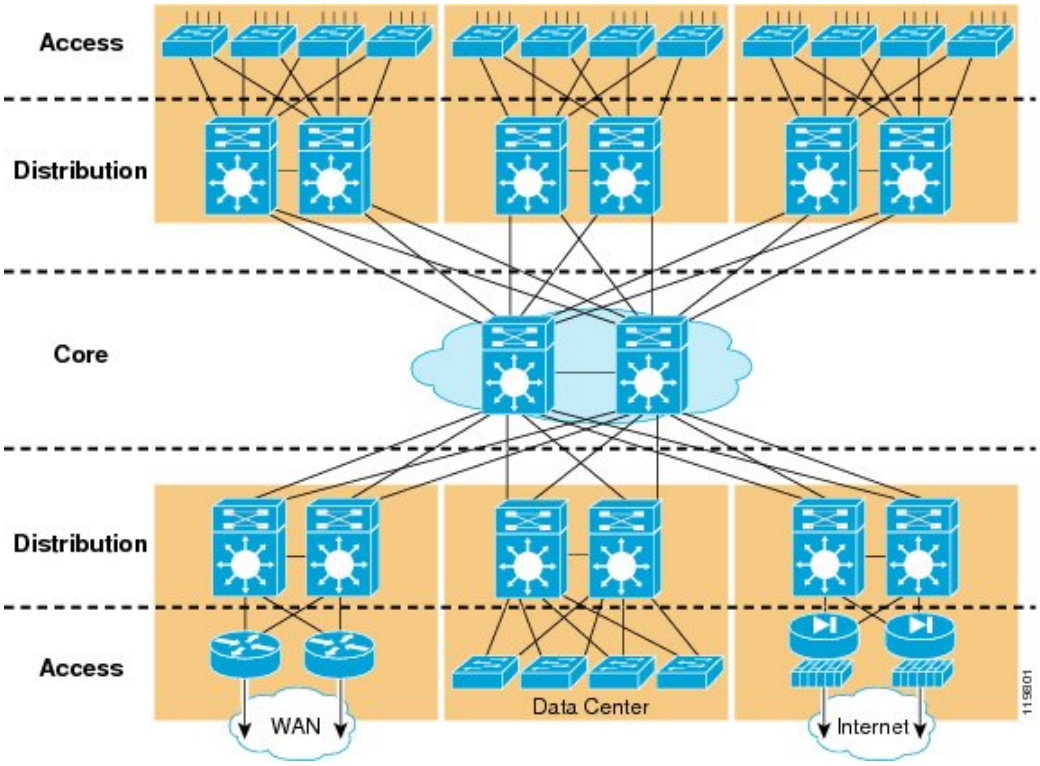
Rozšířitelnost

Každá vyvíjející se instituce nutně potřebuje mít možnost rozšiřovat svou IT infrastrukturu.

Přehlednost

Důležitá pro správce sítě v případě výskytu poruchy. Přehledně navržená síť zjednodušuje detekci a následnou nápravu poruch.

K usnadnění dosažení předchozích cílů se používá model hierarchického strukturování sítě do tří vrstev znázorněných na obr. 2.1.



Obrázek 2.1: Hierarchická struktura sítě [1].

Následuje popis významu jednotlivých vrstev:

Přístupová (Access)

Poskytuje koncovým zařízením přístup do sítě. Řeší oprávnění a bezpečnost připojení koncových zařízení. Každý přístupový přepínač je připojen redundantně ke dvěma různým přepínačům z distribuční vrstvy.

Distribuční (Distribution)

Propojuje uzly z přístupové vrstvy. Zamezuje šíření chyb sítě z přístupové vrstvy do páteře sítě. S přístupovou vrstvou komunikuje službami linkové vrstvy ISO/OSI[1] modelu. Tvoří hranici pro VLAN[2] (bližší popis v kapitole 3.1) z přístupové vrstvy a komunikaci dále směruje . Vyrovnává zatížení sítě. S vrstvou páteře je propojena vždy minimálně dvěma záložními linkami.

[1] Open Systems Interconnection model - Standardizační model počítačových sítí vyvinutý organizací International Organization for Standardization.

[2] Virtual Local Area Network - Logicky nezávislá síť.

Páteř (Core)

Vysokorychlostní páteř používá záložní spoje o kapacitě řádově desítek gigabitů za sekundu. Propojuje linky z přepínačů v distribuční vrstvě. Musí rychle reagovat na změny topologie sítě, vyrovnávat zátěž linek a hledat alternativní cesty do dalších uzlů.

Spoje mezi jednotlivými vrstvami se nazývají uplink. Zpravidla mívají vyšší přenosovou kapacitu než ostatní porty pro koncová zařízení. Jejich účelem je propojení menší části sítě s větším celkem.

2.2 Síťové prvky

Firma Cisco nabízí pro každou vrstvu sítě přepínače, které vyhovují požadavkům konkrétní vrstvy. Stručná charakteristika jednotlivých řad:

Catalyst 6500/6800

- Používané v páteřní části sítě, případně v datových centrech.
- Redundantní zdroje el. energie, chladící ventilátory a řídící jednotky.
- Modulární přepínač poskytující až 13 pozic pro výměnné karty.
- Datová propustnost jedné karty je u řady Catalyst 6500 80 Gbit/s. Propustnost u řady Catalyst 6800 je až 160 Gbit/s.
- Až 16 portů s kapacitou 10 Gbit/s na jedné kartě.
- Podporují služby z linkové a síťové vrstvy ISO/OSI modelu.

Catalyst 4500

- Používané v distribuční vrstvě a v prostředích se smíšenou páteří a distribuční vrstvou do jedné vrstvy.
- Škálovatelné přepínače poskytující až 10 pozic pro výměnné karty.
- Nižší počet portů s kapacitou 10 Gbit/s na kartě.
- Podpora L2[3] i L3[4] funkcionalit.
- Volitelně podporují i záložní zdroje energie a řídící jednotky.

[3] Linková vrstva ISO/OSI síťového modelu.
[4] Síťová vrstva ISO/OSI síťového modelu.

Catalyst 3560 a 3750

- Podpora služeb ze síťové vrstvy ISO/OSI modelu. Zejména směrovací protokoly.
- Díky podpoře L3 služeb může být použit i v distribuční vrstvě. Mohou však být použity i jako přístupové přepínače pro koncová zařízení.
- Podpora PoE[5] pro připojení telefonů a bezdrátových přístupových bodů.
- Pevně daný počet portů. Až 48 portů o kapacitě 1 Gbit/s. Až 4 uplink porty o kapacitě 10 Gbit/s pro konektivitu do distribuční vrstvy, případně do páteře sítě.
- Podpora záložních zdrojů el. energie.

Caralyst 2960

- Používané zpravidla v přístupové vrstvě sítě.
- Pevný počet portů. Až 48 portů o kapacitě 1 Gbit/s.
- Až 4 uplink porty pro připojení k zařízením v distribuční vrstvě sítě.
- Bez redundantních periferií.
- Podpora PoE pro připojení telefonů.

Nexus 5000

- Řada Nexus používá nově navržený operační systém NX-OS.
- Modulární přepínače navržené pro datová centra.
- V základu poskytují až 96 portů o kapacitě 1 Gbit/s.

Nexus 2000

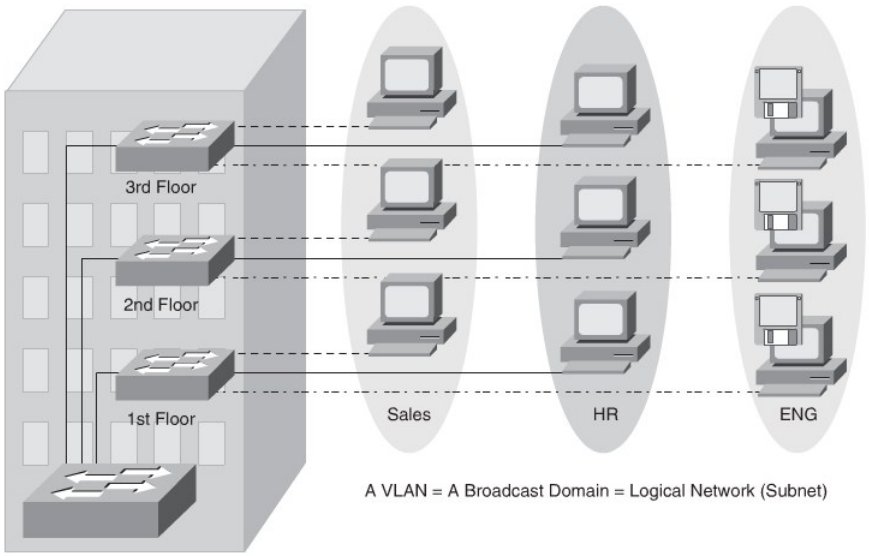
- Nejedná se o samostatný přepínač ale o zařízení zvyšující počet portů[6].
- Ke své činnosti musí být připojen k přepínači Nexus 5000 nebo 7000.
- Poskytuje až 48 portů o kapacitě 1 Gbit/s a 4 uplink porty o kapacitě 10 Gbit/s.

[5]Power over Ethernet - Technologie umožňující přenos napájení po datovém síťovém kabelu.
[6]V odborné terminologii se toto zařízení nazývá *fabric extender*.

3 Virtuální lokální síť

3.1 Popis

V běžné Ethernet sítí jsou všechna zařízení, která přijímají všesměrové zprávy (broadcast) od ostatních zařízení, členy jedné všesměrové domény. Virtuální lokální síť (dále VLAN) umožňuje logicky rozdělit přepínač do několika všesměrových domén. VLAN může být také chápána jako skupina síťových zařízení se společnými požadavky nebo vlastnostmi, sdružených do samostatných celků nezávisle na fyzickém zapojení. V porovnání s podsítěmi protokolu IP[7], dochází u VLAN k rozdělení sítě na L2 úrovni ISO/OSI modelu. V praxi bývá dobrým zvykem mapovat jednu IP podsíť do jedné VLAN. Komunikace mezi uzly z různých VLAN je směrována zařízením pracujícím na L3 vrstvě. Jedná se o mocný mechanismus zvýšení přehlednosti, bezpečnosti, stability a výkonnosti sítě.
Příklad rozdělení různých oddělení instituce do tří VLAN ukazuje obr. 3.1. Každé oddělení tvoří jednu všesměrovou doménu.



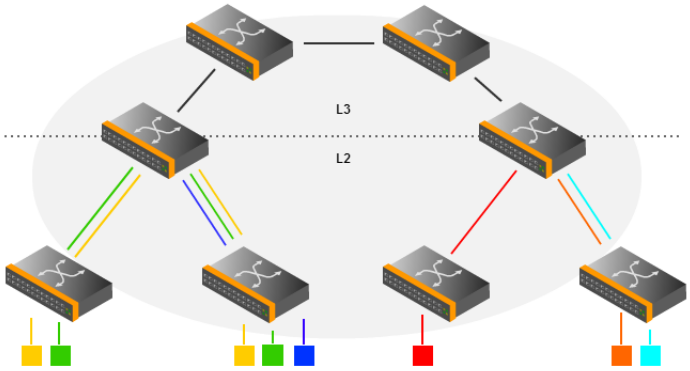
Obrázek 3.1: Příklad VLAN [6].

[7]Internet Protocol - Základní protokol pro přenos zpráv pracující na síťové vrstvě ISO/OSI modelu.

V praxi se používají dva modely VLAN. Lokální VLAN a End-To-End VLAN. Oba modely mají své výhody i nevýhody.

Local VLAN

Lokální VLAN je taková, která není šířena mezi přepínači v distribuční vrstvě. Používají se v malých lokalitách (např. místnost, patro budovy), případně pro testovací účely. Komunikaci mezi VLAN je třeba směrovat L3 zařízením. Příklad lokální VLAN ukazuje obr. 3.2. Např. uživatelé z VLAN označené zelenou barvou se mohou připojit pouze do dvou přepínačů. Do přepínačů z pravé části sítě mohou mít zakázaný přístup.



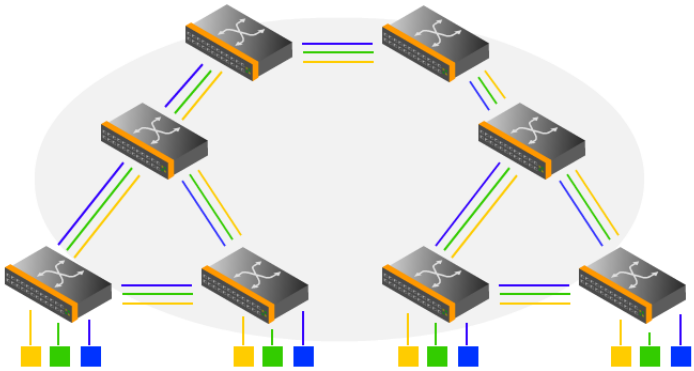
Obrázek 3.2: Local VLAN [7].

Charakteristiky:

- VLAN je omezena do oblasti mezi přístupovou vrstvou a distribuční vrstvou sítě.
- Komunikace mimo VLAN je směrována na distribuční vrstvě sítě do ostatních sítí.
- Jednodušší údržba VLAN při změnách topologie sítě.
- Protokol VTP je nutné nastavit do transparentního režimu, aby nedocházelo k šíření informací o VLAN.

End-To-End VLAN

Konkrétní VLAN může být přenášena mezi lokalitami celou sítí. Uživatelé se mohou vždy připojit do stejné VLAN bez ohledu na fyzické umístění. Obr. 3.3 ukazuje přenos VLAN 1, 2 a 3 celou sítí.



Obrázek 3.3: End-To-End VLAN [7].

Propojení End-To-End VLAN vyžaduje použití trunku.

Charakteristiky:

- Každá VLAN je vedena celou sítí mezi přepínači.
- Umožňují maximalizovat mobilitu uživatelů.
- Každá VLAN má svou IP podsíť.
- VTP protokol je na přepínačích nastaven do server/klient módu.
- Náročné na údržbu v sítích, kde dochází k častým změnám topologie sítě.

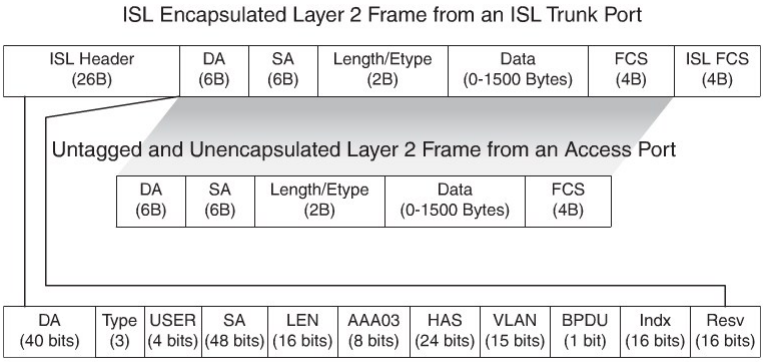
Každý port přepínače může být nastaven do jednoho ze dvou režimů. Prvním z režimů je access neboli přístupový port. Do access portů se zpravidla připojují koncová zařízení, např. servery, pracovní stanice, síťové tiskárny či telefony. Access port může být přiřazen nejvýše do jedné datové VLAN a jedné hlasové VLAN. Zařízení připojená do access portů ve stejné VLAN mohou mezi sebou komunikovat pomocí přepínače. Pokud by měla být komunikace šířena do dalších přepínačů, nabízí se dvě možnosti. První variantou je propojení dvou access portů z obou přepínačů a to pro všechny potřebné VLAN. Tato možnost je velice nevhodná, jelikož zbytečně plýtvá počtem volných portů, které mohou být použity pro připojení dalších koncových zařízení. Řešením je použití druhého režimu portu, který se nazývá trunk.

3.2 Trunk

Pokud je port nastaven v režimu trunk, může na rozdíl od access portu přenášet data z vícero VLAN. Spoje označené jako trunk se používají k hierarchickému propojení většího počtu přepínačů. Každý rámec musí být označen podle příslušnosti ke konkrétní VLAN. Pro značkování rámců mohou být použity následující protokoly.

3.2.1 Inter-Switch Link

Proprietární protokol vyvinutý firmou Cisco. Používá externí značkování Ethernet rámce. Přidává 26 bajtů informací včetně 15bitového identifikátoru VLAN a nově přepočítaný kontrolní součet celého rámce o velikosti 4 bajtů. Výsledný formát rámce je na obr. 3.4.



Obrázek 3.4: Enkapsulace ISL rámců [7].

V současnosti se protokol ISL už prakticky nepoužívá.

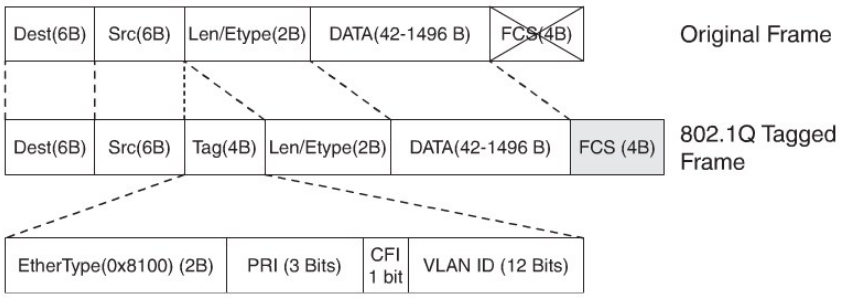
3.2.2 IEEE 802.1q

Otevřený standard vyvinutý asociací IEEE[8] používaný v moderních sítích. Na rozdíl od ISL používá interní značkování rámce. Vkládá nově přepočítaný kontrolní součet rámce včetně tagu. Princip značkování rámce je na obr. 3.5.

[8]Institute of Electrical and Electronics Engineers - Mezinározní organizace pro rozvoj elektrotechniky a souvisejících technologií.

Výhody IEEE 802.1q:

- Široce podporovaný protokol.
- Podpora priority rámců různých typů přenosu.
- Menší hlavička rámce. Přidává tag dlouhý 4 bajty. ISL celkem 30 bajtů.



Obrázek 3.5: Značkování 802.1q [7].

Význam jednotlivých polí tagu vloženého do Ethernet rámce:

EtherType (TPID)

Specifikuje 802.1q rámec. Nastaven na hodnotu 0x8100.

TCI

Tag Control Information - Řídící informace tagu o délce 16 bitů. Význam jednotlivých polí je následující:

PRI

Priority value - Nastavuje prioritu různým skupinám přenosu (data, zvuk, video). Obdoba QoS.

CFI

Canonical Format Identifier - Pro Ethernet nastaven na 0. Token Ring sítě mají nastaveno 1.

VID

VLAN ID - 12bitový identifikátor VLAN. Hodnota 0 indikuje prioritní rámce. Hodnota 4095 je vyhrazena.

Trunk standardu 802.1q definuje nativní VLAN pro rámce, které nejsou značkovány. Při příjmu neznačkovaných rámců, posílají přepínače neznačkované rámce pouze do nativní VLAN. Ve výchozím stavu je VLAN s číslem 1 používána jako nativní.

3.3 Dynamic Trunking Protocol

Cisco proprietární protokol slouží k navázání trunk linky vysíláním a přijímáním inicializačních zpráv.

Módy DTP protokolu nastavitelné na portu:

Access

Přepne port do permanentního access módu a vyjedná přepnutí linky do netrunk módu.

Trunk

Přepne port do permanentního trunk módu a vyjedná přepnutí linky do turnk módu.

Nonegotiate

Přepne port do permanentního trunk módu ale nevysílá žádné DTP zprávy. Pro vytvoření trunk linky je třeba nastavit trunk mód na obou stranách ručně. Používá se v případech, kdy jedno ze zařízení nepodporuje DTP protokol.

Dynamic desirable

Aktivně se snaží o vytvoření trunk linky. Port se přepne do trunk módu v případě, že druhá strana je nastavena jako Trunk, Desirable nebo Auto.

Dynamic auto

Přepne port do trunk módu v případě, že druhá strana je nastavena do Trunk nebo Desirable módu. Pokud je druhá strana nastavena na Dynamic Auto, trunk se nikdy nevytvoří.

Výsledný stav trunku popisuje tab. 3.1.

	Dynamic Auto	Dynamic Desirable	Trunk	Access
Acces	Access	Access	Omezená konektivita	Access
Trunk	Trunk	Trunk	Trunk	Omezená konektivita
Dynamic Desirable	Trunk	Trunk	Trunk	Access
Dynamic Auto	Access	Trunk	Trunk	Access

Tabulka 3.1: DTP konfigurace módu [17].

3.4 VLAN Trunking Protocol

Cisco proprietární protokol synchronizuje informace o VLAN mezi jednotlivými přepínači. Udržuje konzistentní informace o číselném označení VLAN a jejich pojmenování. VTP se nestará o příslušnost portů ke konkrétní VLAN. Skupinové zprávy (multicasty) přenáší pouze trunky po management VLAN. Konfigurace VTP protokolu vyžaduje nastavení jména domény, hesla a režimu protokolu.

VTP doména je skupina přepínačů, která sdílí stejné jméno VTP domény. Přepínače přijímají VTP zprávy jen ze stejné domény. Zprávy z ostatních domén zahazují. Protokol VTPv1 umí pracovat s číselným rozsahem VLAN od 1 do 1000. K udržení konzistentnosti informací o stavu všech VLAN je využíván mechanismus revize konfigurace. Vytvořením nové VTP domény je číslo revize nastaveno na 0 a jakákoliv další změna zvýší hodnotu čísla revize o 1. Další vývoj VTP protokolu přinesl následující vylepšení:

VTPv2

- Podpora Token Ring přepínačů a VLAN.
- VTPv1 transparentní přepínače přeposílají VTP zprávy pouze v případě, že mají stejné jméno domény a verzi. Nezávislý transparentní režim ve VTPv2 tyto hodnoty nekontroluje a ihned přeposílá.
- VTPv2 kontroluje konzistenci dat pouze při zadávání nových dat skrze konzolu nebo SNMP[9] protokol. Neprovádí kontrolu konzistence dat v případě, že přijme VTP zprávu od jiného VTP serveru a MD5 otisk souhlasí.

VTPv3

- Vylepšená autentikace.
- Rozšířený rozsah VLAN 1025 až 4096.
- Šíření informací o privátních VLAN.
- Ochrana proti nechtěnému přepsání VLAN databáze z nově připojeného VTP serveru s vyšším číslem revize.

[9]Simple Network Management Protocol - Internetový protokol pro správu síťových zařízení.

- Podporovaný operačním systémem Cisco IOS[10] Release 12.2(33)SXI a vyššími.

VTP protokol komunikuje mezi přepínači na L2 úrovni. Vyměňují si mezi sebou tři druhy zpráv, posílané na všesměrovou MAC[11] adresou `01:00:0C:CC:CC:CC`. Zprávy se dělí na:

Summary advertisement

Po přijetí zprávy zkontroluje přepínač jméno VTP domény, ze které zpráva přišla. Pokud jméno domény souhlasí, porovná číslo revize zprávy a porovná jej s vlastním číslem revize. Pokud je číslo revize zprávy nižší, zpráva se zahodí. V opačném případě odešle přepínač zprávu advertisement request.

Advertisement request

Zpráva je odesílaná jako odpověď na summary advertisement, která měla vyšší číslo revize. Stejný typ zprávy je odesílán i v případě, že došlo ke změně názvu VTP domény.

Subset advertisement

Vyskytnou-li se na straně serveru změny v konfiguraci VLAN, odešle přepínač subset advertisement klientovi. Zahrnuje název domény, číslo a název VLAN a číslo revize.

Přepínač s VTP ve verzi 2 lze nastavit do jednoho z těchto režimů:

Server

Výchozí režim VTP protokolu. V tomto režimu lze vytvářet, modifikovat a odstraňovat VLAN. VTP servery v doméně posílají konfiguraci VLAN ostatním přepínačům ve stejné VTP doméně. VTP verze 3 navíc umožňuje nastavit jeden primární server a několik sekundárních serverů.

Client

Klient se chová stejně jako server s tím rozdílem, že nelze vytvářet, měnit a odstraňovat VLAN.

[10] Internetwork Operating System - Operační systém určený pro produkty firmy Cisco.

[11] Medium Access Control - Podvrstva linkové vrstvy ISO/OSI modelu, která řídí přístup na fyzické médium. Adresuje komunikaci se sousedními uzly v síti.

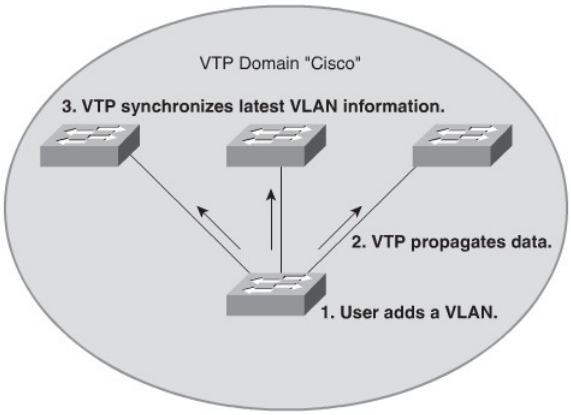
Transparent

V transparentním režimu přepínač nešíří svou vlastní konfiguraci VLAN, pouze přeposílá přijaté VTP zprávy dál. Na přepínači v transparetním režimu lze vytvářet, upravovat a mazat VLAN aniž by byla ovlivněna VTP doména.

Off

Přepínač se chová stejně jako v transparentním módu s tím rozdílem, že zprávy přijaté od ostatních zařízení rovnou zahazuje.

Postup propagace zpráv z VTP serveru je na obr. 3.6.



Obrázek 3.6: Šíření zpráv VTP protokolu [7].

Následuje vysvětlení jednotlivých kroků propagace zpráv.

1. Administrátor přidá, upraví nebo smaže na VTP serveru VLAN.
2. VTP odešle trunkem informaci o změně VLAN všem sousedním přepínačům ve stejné VTP doméně.
3. Každý přepínač ze získaných informací aktualizuje svou databázi VLAN.

3.5 Privátní VLAN

V některých situacích je třeba zamezit komunikaci mezi zařízeními ve stejné VLAN aniž by bylo nutné je umisťovat do různých podsítí. Privátní VLAN umožňují na L2 úrovni izolovat zařízení ve stejné podsítí. Privátní VLAN typicky využívají poskytovatelé internetového připojení. Každý z klientů je připojen do izolovaného

portu. Zákazník může přistupovat do veřejné sítě Internet ale nemůže komunikovat s dalšími stanicemi ve stejné VLAN. Poskytovatelé mohou tímto způsobem efektivně řešit bezpečnostní politiku uvnitř své sítě.

V privátní VLAN mohou být VLAN nastaveny ve dvou módech:

Primární privátní VLAN

Primární privátní VLAN může sestávat z vícero sekundárních privátních VLAN patřícími do stejné podsítě jako primární privátní VLAN. Přenáší komunikaci z promiskuitních portů do izolovaných, komunitních a dalších promiskuitních portů ve stejné primární privátní VLAN.

Sekundární privátní VLAN

Každá sekundární privátní VLAN je mapována na jednu primární privátní VLAN. Koncová zařízení jsou připojena do sekundárních privátních VLAN. Sekundární privátní VLAN se dále dělí na:

Komunitní privátní VLAN

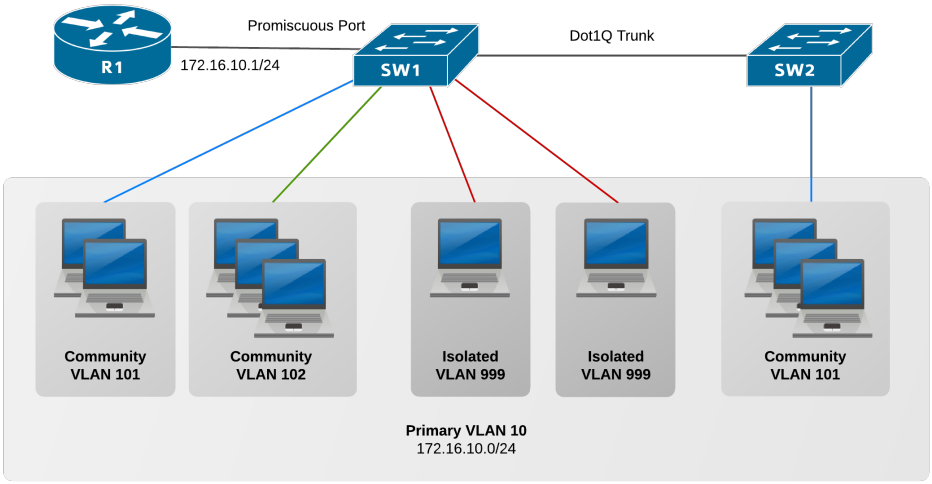
Porty v komunitní privátní VLAN mohou komunikovat s ostatními porty ve stejné komunitě a s promiskuitními porty ze stejné privátní VLAN.

Izolovaná privátní

Porty z izolované privátní VLAN mohou komunikovat pouze s promiskuitními porty. Porty v izolované VLAN nemohou komunikovat mezi sebou.

S použitím privátních VLAN je potřeba mít trunk nakonfigurován tak, aby přenášel informace o privátní VLAN. Pokud přepínač obdrží rámec na izolovaném nebo komunitním portu, přepošle rámec trunkem označeným identifikátorem sekundární VLAN. Přijímající přepínač předá přijatý rámec do patřičné sekundární VLAN. Pokud přepínač obdrží rámec na kterémkoliv z promiskuitních portů, přepošle rámec trunkem označeným s číslem primární VLAN. Přijímající přepínač přepošle rámec do konkrétní primární VLAN. Zajištění správného přeposílání rámců mezi přepínači vyžaduje konzistentní konfiguraci primárních a sekundárních VLAN. Pro šíření informací o konfiguraci privátních VLAN do ostatních přepínačů musí být použit VTPv3.

V privátní VLAN může být port nastaven do jednoho ze tří režimů:



Obrázek 3.7: Typy portů v privátní VLAN [7].

Izolovaný

Port je zcela izolovaný od ostatních portů ve stejné privátní VLAN. Komunikace je možná pouze mezi daným izolovaným portem a porty v promiskuitním režimu z primární VLAN. Obr. 3.7 ukazuje dvě stanice v izolované VLAN 999 připojené do stejné privátní VLAN na izolovaných portech. Komunikovat mohou pouze s promiskuitním portem, mezi sebou nikoliv.

Promiskuitní

Port může komunikovat se všemi porty ve stejné privátní VLAN. Promiskuitní port může patřit nejvýše do jedné primární VLAN. Promiskuitní port může být mapován na vícero sekundárních privátních VLAN. Obr. 3.7 ukazuje port připojený do směrovače, nastavený do promiskuitního režimu kvůli vzájemné komunikaci s ostatními klienty v jiných sítích.

Komunitní

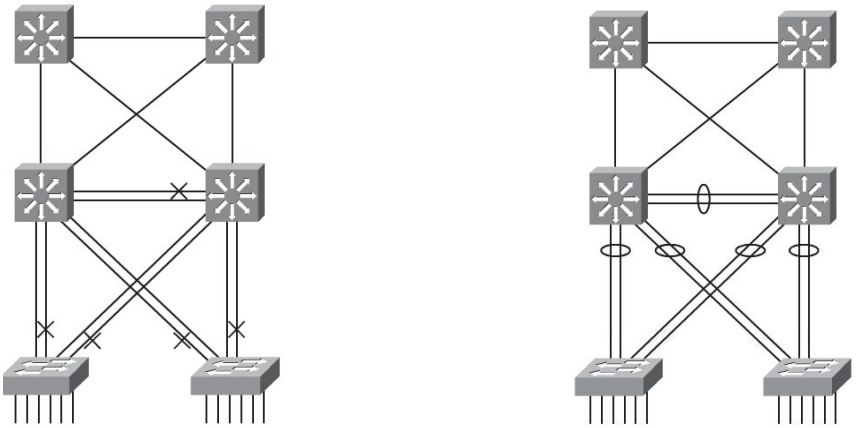
Komunikuje s ostatními komunitními porty ve stejné komunitní VLAN a s promiskuitními porty z primární VLAN. S izolovanými porty nekomunikují.

Obr. 3.7 ukazuje stanice v komunitní VLAN 101 na SW1 a VLAN 101 na SW2, které mohou mezi sebou komunikovat. Se stanicemi v komunitní VLAN 102 komunikovat nemůže, jelikož se jedná o jinou komunitní VLAN.

3.6 Agregace linek

Agregace linek se používá v případech, kdy je třeba zvýšit propustnost linky a rozložit zatížení linky na více spojů. V rozlehlejších sítích může docházet k přetížení linek a následně k výpadkům komunikace. Jednou z možností je použití portu s vyšší propustností. Toto řešení je pouze dočasné dokud opět nedojde k nárůstu vytížení linky. Druhou možností je násobné fyzické propojení přepínačů. Nevýhodou je, že porty mohou být zablokovány STP protokolem (obr. 3.8a), navíc musí být zajištěna konzistence v konfiguraci propojených portů.
Technologie EtherChannel umožňuje sdružit více fyzických portů do jednoho logického portu (obr. 3.8b). Výhody použití EtherChannel:

- Využívá existující porty. Není třeba měnit spojení za dražší technologii.
- Redundantní spojení mezi přepínači. Porucha jednoho fyzického spoje, nevyvolá změnu topologie sítě. Dokud existuje alespoň jedno fyzické spojení, je EtherChannel stále aktivní.
- Vyrovnává zatížení jednotlivých fyzických linek.
- Změna v konfiguraci celého EtherChannelu udržuje konzistenci konfigurace fyzických portů.



(a) (b)

Obrázek 3.8: Použití EtherChannel [7].

Důležitý je vztah mezi technologií EtherChannel a protokolem STP. EtherChannel vytváří logické spojení mezi dvěma přepínači. STP se snaží odstranit cyklické spoje.

Logický spoj je z hlediska STP protokolu nadřazen fyzickým rozhraním. Existují-li mezi dvěma přepínači více spojů typu EtherChannel, může dojít k zablokování jednoho ze spojů a tím pádem všech portů patřících do EtherChannelu.

3.6.1 Protokoly

EtherChannel používá k vytvoření PortChannelu jeden ze dvou protokolů. Oba protokoly zajišťují konzistenci nastavení fyzických portů. Porty musí mít nastavenu stejnou rychlost, duplex a informace o VLAN včetně nativní VLAN. K vytváření EtherChannel spojení jsou používány následující protokoly:

Port Agregation Protocol (PAgP)

Proprietární Cisco protokol odesílá PAgP zprávy mezi přepínači a vyjedná navázání kanálu. Spanning Tree protokol s vytvořeným kanálem zachází jako s fyzickým portem. Jednotlivé PAgP módy:

Auto

Port odpovídá na přijaté PAgP zprávy. Nezahajuje PAgP negociaci.

Desirable

Aktivně se snaží o negociaci spojení vysíláním PAgP zpráv.

On

Vynutí vytvoření kanálu bez odesílání a přijímání PAgP zpráv.

Výsledek vyjednávání navázání kanálu je na obr. 3.9a.

Link Aggregation Control Protocol (LACP)

Standard IEEE 802.3ad. Používán v sítích se zařízeními od různých výrobců. Funguje obdobně jako PAgP. Odesílá zprávy sousednímu přepínači a snaží se navázat kanál. Jednotlivé LACP módy:

Active

Aktivně vyjednává navázání kanálu odesíláním LACP zpráv.

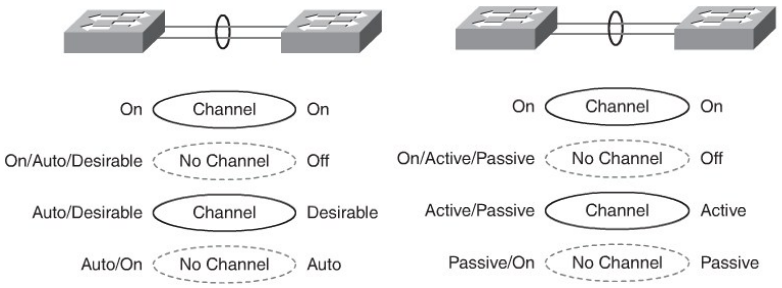
Passive

Odpovídá na přijaté LACP zprávy.

On

Vynutí nastavení kanálu na portu bez LACP zpráv.

Výsledek vyjednávání navázání kanálu je na obr. 3.9b.



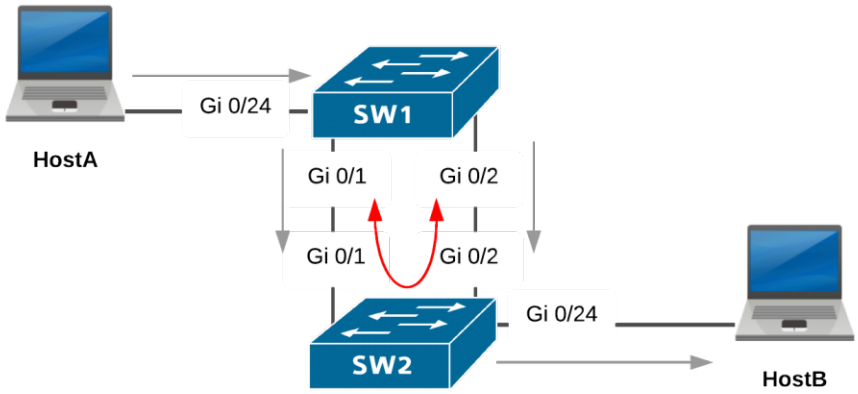
(a) PAgP (b) LACP

Obrázek 3.9: Módy protokolů EtherChannel [7].

4 Spanning Tree protokol

4.1 Smyčky v síti

Hlavním důvodem, proč vznikl STP protokol, byl výskyt smyček v síti. Ty mohou vznikat v síti buď nesprávnou konfigurací zařízení nebo zapojením vícero linek mezi přepínači kvůli redundanci a vyrovnávání zátěže sítě. Vzniklé smyčky mohou zavinit zahlcení sítě. Příkladem je obr. 4.1.



Obrázek 4.1: Zahlcení sítě [8].

Zahlcení sítě může nastat v případě, že nastane sled těchto událostí:

1. Stanice i přepínače po spuštění neznají MAC adresy ostatních zařízení.
2. Stanice HostA chce komunikovat se stanicí HostB. Odešle broadcast zprávu.
3. Přepínač SW1 odešle zprávu na všechny porty vyjma portu, ze kterého broadcast přišel. Do CAM[12] tabulky přidá číslo portu, na kterém se nachází HostA s jeho MAC adresou.

[12]Content-addressable memory - Speciální typ paměti vyvinutý pro rychlé vyhledávání uložených dat.

4. Přepínač SW2 přijme broadcast na dvou portech. Protože SW2 nezná cílovou MAC adresu, rozešle přijaté broadcast zprávy na všechny ostatní porty vyjma portů, ze kterých broadcast přišel. HostB přijme broadcast zprávu. SW2 o tom ale neví.
5. HostB odešle dvě odpovědi. SW2 přidá záznam do CAM tabulky. Rozešle zprávu na všech ostatních portech.
6. Současně SW1 dostane broadcast zprávu od SW2 a přepíše záznam o adrese portu HostA na porty, ze kterých přišel broadcast zpět. SW1 tedy považuje HostA za přepojenou na jiný port.
7. Obdobná situace s cyklickým doručováním nastává u zpráv od HostB.
8. Zahlcení sítě narůstá.

4.2 Vývoj STP protokolu

- První verzi Spanning Tree protokolu (dále jen STP) vyvinula v roce 1985 Radia Joy Perelman ze společnosti Digital Equipment Corporation. Původní verze bývá také označována jako DEC STP.
- V roce 1990 byla společností IEEE vydána standardizovaná verze STP protokolu pod označením 802.1d.
- Common Spanning Tree vychází ze standardu 802.1d. Vytváří jedinou instanci pro celou síť bez ohledu na počet VLAN. Nevýhodou jediné instance protokolu je, že vytváří jeden strom sítě a tím pádem může docházet k ovlivnění propustnosti během konvergence protokolu po změnách topologie sítě.
- Per VLAN Spanning Tree Plus (PVST+) je Cisco proprietární vylepšení standardu 802.1d, které pro každou VLAN vytváří jednu instanci STP protokolu. Oproti 802.1d navíc podporuje funkce jako BPDU[13] guard, BPDU filter, root guard, loop guard a PortFast. Bližší popis těchto funkcí je k nalezení v literatuře [17]. PVST+ má vyšší nároky na výkon procesoru a paměť přepínače.

[13]Bridge Protocol Data Unit - Datový ráměc přenášející informace o STP protokolu.

- V roce 2001 byl publikován standard Rapid STP (RSTP) označovaný jako IEEE 802.1w. Později v roce 2004 nahradil původní specifikaci STP 802.1d a je dále označován jako IEEE 802.1d. RSTP dosahuje oproti předchozímu standardu rychlejší konvergence. Stále však vytváří jediný STP strom v síti a může tedy docházet k neoptimálnímu toku dat. Ve srovnání se STP je RSTP náročnější na CPU a paměť ale méně náročný než PVST+.
- Multiple Spanning Tree (MST) standardizovaný jako IEEE 802.1s se inspiroval u dřívějšího Cisco proprietárního Multi-Instance Spanning Tree Protocol. Redukuje počet vytvářených STP instancí mapováním několika VLAN do jedné instance. Cisco zařízení podporují nejvýše 16 instancí RSTP. Do jedné RSTP instance slučuje množinu VLAN se stejnou fyzickou a logickou topologií.
- Per VLAN Rapid Spanning Tree Plus (PVRST+ někdy i RPVST+) je Cisco proprietární vylepšení RSTP, které je podobné PVST+. PVRST+ rychleji konverguje a má vylepšené chování během změn topologie sítě.

4.3 Algoritmus STP

Základní princip STP vychází z matematické teorie grafů. Na počítačovou síť pohlíží jako na ohodnocený neorientovaný graf, kde ohodnocením hran je propustnost linky. Algoritmus hledá kostru sítě, s minimálním ohodnocením. STP je postaven na konceptu kořenového přepínače a systému přiřazování rolí jednotlivým portům. Výběr kořenového přepínače má vliv na určení rolí všech portů, které STP protokol vidí. STP protokol komunikuje mezi přepínači zasíláním tzv. BPDU rámců na všesměrovou MAC adresu `01:80:C2:00:00:00`. Formát BPDU zprávy je na obr. 4.2. Dále následuje popis významu jednotlivých polí BPDU standardu 802.1d:

Protocol ID

Identifikátor STP protokolu. Výchozí hodnota je `0`.

Version

Identifikátor verze protokolu. Nastaveno na `0`.

Message Type

Určuje typ zprávy. Nastaveno na `0`.

The BPDU Fields

Field #	Bytes	Field
1-4	2	Protocol ID
	1	Version
	1	Message type
	1	Flags
5-8	8	Root ID
	4	Cost of path
	8	Bridge ID
	2	Port ID
9-12	2	Message age
	2	Max age
	2	Hello time
	2	Forward delay

Obrázek 4.2: Zpráva BPDU [9].

Flags

Používá dva bity. Topology change indikující změnu topologie sítě. Druhý bit je Topology change acknowledgment, indikující potvrzení příjmu změny topologie.

Root ID

Pole je složeno ze dvou části. Priorita o velikosti 2 bajtů a 6 bajtů MAC adresa kořenového přepínače. Po spuštění je Root ID přepínače stejné jako Bridge ID. Během procesu výběru kořenového přepínače se Root ID postupně nahrazuje Bridge ID s nejnižší hodnotou identifikující kořenový přepínač.

Cost of path

Cena cesty od vlastního přepínače ke kořenovému přepínači. Cena cesty je aktualizována každým přepínačem podél cesty ke kořenovému přepínači.

Bridge ID

Identifikátor priority a MAC adresy přepínače odesílající BPDU. Pokud ostatní přepínače dostanou více zpráv obsahující stejné Bridge ID a různé celkové ohodnocení cest, znamená to, že k přepínači existuje více cest a vybere tu s menším ohodnocením.

Port ID

Identifikuje port, ze kterého byla BPDU zpráva odeslána.

Message Age

Stáří BPDU zprávy od odeslání kořenovým přepínačem. Po vypršení platnosti zprávy je zahájena nová volba kořenového přepínače.

Max Age

Maximální doba platnosti zprávy. Inkrementována po každém přeposlání přepínačem. Výchozí hodnota je 20 sekund.

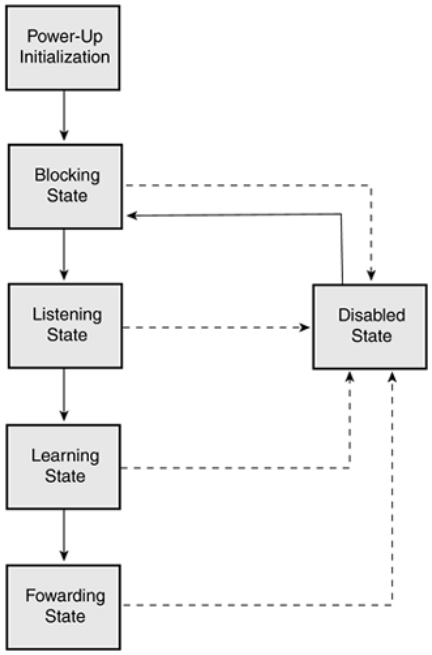
Hello time

Interval mezi odesláním BPDU kořenovým přepínačem. Výchozí hodnota jsou 2 sekundy.

Forward delay

Interval mezi změnou topologie a změnou stavu. Výchozí hodnota je 15 sekund

Během konvergence protokolu mění porty svůj stav v závislosti na přijatých BPDU zprávách v čase. Stavový diagram přechodů je na obr. 4.3.



Obrázek 4.3: Stavový diagram protokolu STP [10].

Následuje popis jednotlivých stavů portu během konvergence STP protokolu:

Disabled

Port je vypnutý. Neodesílá žádné rámce.

Blocking

Port nepřeposílá rámce datové komunikace. Přijímá pouze BPDU rámce informující o cestě ke kořenovému přepínači a výsledné roli všech portů.

Listening

Přijímá BPDU rámce od sousedních přepínačů. Navíc odpovídá svými vlastními BPDU rámci sousedním přepínačům.

Learning

Port je připraven odesílat datové rámce. Do tabulky MAC adres si přidává MAC adresy připojených zařízení.

Forwarding

Odesílá a přijímá datové i BPDU rámce. Port je součástí aktivní topologie STP protokolu.

V ustáleném stavu se porty mohou nacházet v jedné ze čtyř rolí:

Root port

Nenacházejí se na kořenovém přepínači. Na ostatních přepínačích se nachází právě jeden root port a vede z něj cesta ke kořenovému přepínači s nejnižší cenou.

Designated port

Kořenový přepínač má všechny porty designated. Na ostatních přepínačích jsou to porty, ze kterých se přeposílá komunikace na root port. Na jednom segmentu sítě, se může vyskytovat pouze jeden designated port. Pokud jich existuje více, STP algoritmus povolí pouze jeden z portů. Volba je založena na prioritě portů.

Non-designated port

Definovaný jako port, který není root ani designated port. Nepřeposílá žádná data. Port je blokovaný.

Disabled port

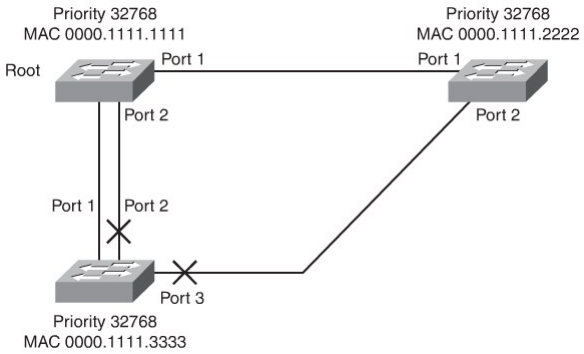
Port, který je správcem přepnutý do vypnutého stavu. Neúčastní se STP procesu.

Při výběru root portu se přepínač řídí cenou linky. Cena je odvozena od propustnosti linky. Cena linky je počítána kumulativně podél cesty mezi přepínačem a kořenovým přepínačem. V tabulce 4.2 jsou uvedeny ceny linek původního standardu STP 802.1d a pro porovnání RSTP 802.1w.

Rychlost [Mbit]	**Cena 802.1d**	**Cena 802.1w**
10	100	100
100	10	19
1000	1	4
10 000	1	2

Tabulka 4.2: Výchozí ceny portů.

Obr. 4.4 popisuje volbu root portů. Kořenovým přepínačem je přepínač s nejmenším Bridge ID s prioritou 32768 a MAC adresou `0000.1111.1111`. Přepínač s MAC `0000.1111.3333` má tři 100 Mbit cesty k root přepínači. Oba porty 1 a 2 mají cenu 19. Port 3 má celkovou cenu cesty ke kořenovému přepínači 38 (19+19). Pokud na jednom přepínači existují dva porty se stejnou cenou, rozhoduje o určení root portu priorita portu a číslo portu. Výchozí hodnota priority je 128. První port má celkovou prioritu `128.1`, druhý port `128.2` atd. Jako root port je vybrán port s nejnižší číselnou hodnotou priority.



Obrázek 4.4: Cena STP portů [11].

4.4 Rapid Spanning Tree

Protokol RSTP vznikl jako revize původního STP prtokolu. Ve srovnání s původním standardem 802.1d, RSTP rychleji konverguje po změně topologie sítě. Navíc je RSTP s původním standardem zpětně kompatibilní. Přidává další role portu - alternate a backup.

Discarding

Nepřeposílá datové rámce. Nevytváří tak smyčky v síti. Stav se objevuje v případě ustálené topologie sítě i během změny topologie.

Learning

Přijímá a odesílá datové rámce. Zaplňuje MAC tabulku. Stav se objevuje v případě ustálené topologie sítě i během změny topologie.

Forwarding

Vyskytuje se pouze v ustáleném stavu RSTP. Přeposílá všechna data pouze po dokončení procesu schválení změny topologie ostatními přepínači.

RSTP definuje dvě nové role portů, které vycházejí z původní non-designated role:

Alternate

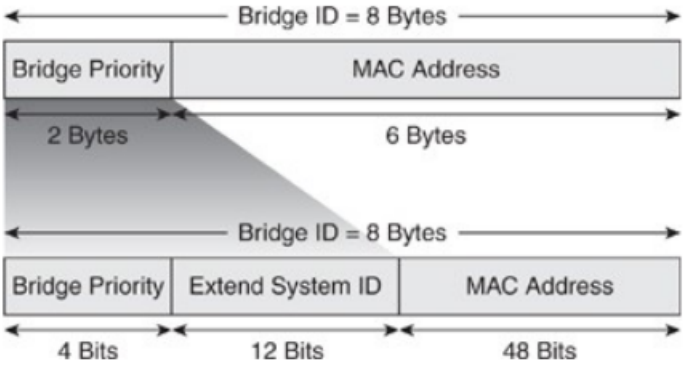
Poskytuje alternativní cestu ke kořenovému přepínači. Jestliže dojde k výpadku root portu, alternate port se ppřepne do designated role a odešle BPDU sousednímu zařízení. V ustáleném stavu se dřívější alternate port stane root portem. V ustáleném stavu přijímá pouze BPDU od jiného přepínače. Ostatní rámce zahazuje.

Backup

V ustáleném stavu přijímá BPDU rámce vysílané ze stejného přepínače. Pokud selže designated port na stejném switchi, backup port se stane designated portem. Ostatní rámce zahazuje stejně jako v případě alternate portu.

4.5 Rapid Per VLAN Spanning Tree+

Cisco proprietární vylepšení standardu 802.1w. PVRST+ vytváří pro každou VLAN samostatnou instanci, proto má ze všech STP protokolů nejvyšší nároky na CPU a paměť přepínače. PVRST+ nabízí další pokročilé funkce, např. BPDU guard, BPDU filtry, Root guard, Loop guard, UDLD nebo PortFast. Podrobnější popis těchto funkcí lze nalézt v literatuře [17]. Oproti předchozím protokolům, má PVRST+ odlišnou strukturu BridgeID v BPDU rámci (obr. 4.5). Priorita přepínače je rozdělena na dvě pole. Původní hodnota Bridge priority je zkrácena na 4 bity a k ní je připojeno Extended system ID o délce 12 bitů. Pole pro MAC adresu je nezměněno.



Obrázek 4.5: BridgeID protokolu RPVST+ [12].

Hodnota Bridge priority roste o násobky hodnoty 4096. Extended system ID je číslo VLAN, pro kterou je dané BPDU přenášeno. Výsledná priorita je součtem Bridge priority a Extended system ID.

5 SNMP

5.1 Popis

Protokol SNMP je široce používaný aplikační protokol k získávání informací o stavu síťových zařízení. SNMP vychází z protokolu SGMP[14], který byl vyvinut v roce 1987. První verze protokolu SNMP je popsána v RFC 1157 z roku 1988. Postupem času se protokol vyvíjel. Následuje stručný popis vlastností jednotlivých verzí SNMP:

SNMPv1

Pro přenos zpráv využívá datagramových služeb protokolu UDP[15] na portu 161. Port 162 používá k naslouchání příchozím trap zprávám. Definuje pětici zpráv *GetRequest*, *GetNextRequest*, *SetRequest*, *Response* a *Trap*. Protokol SNMP verze 1 není nijak zabezpečen. Autentizace probíhá pouze zadáním jména komunity, které je přenášeno nešifrovaně v textové podobě. V dnešní době se tato verze protokolu již prakticky nepoužívá.

SNMPv2

Přidává nový požadavek *GetBulkRequest*. Optimalizuje komunikaci mezi manažery a vylepšuje výkonnostní nedostatky z první verze.

SNMPv3

Vylepšuje bezpečnost komunikace. Přidává podporu šifrování, zajišťuje integritu zpráv, ověřování a řízení přístupů. Zabezpečení je řešeno uživatelským jménem (obdoba komunitního řetězce), autorizačním heslem a klíčem. Komunikace je šifrována použitím algoritmu DES[16] nebo pokročilejším AES[17]. Autorizace používá otisk MD5[18] nebo SHA[19].

[14] Simple Gateway Monitoring Protocol - Zastaralý aplikační protokol pro správu sítě.
[15] User Datagram Protocol - Nespojovaná služba transportní vrstvy ISO/OSI modelu.
[16] Data Encryption Standard - Symetrická šifra, která je v dnešní době považována za nespolehlivou.
[17] Advanced Encryption Standard - Symetrická šifra, která je považována za bezpečnou.
[18] Message Digest algorithm 5 - Hašovací funkce pro vytváření digitálních otisků dat. V dnešní době se doporučuje MD5 nepoužívat z bezpečnostních důvodů.
[19] Secure Hash Algorithm - Bezpečný hašovací algoritmus.

5.2 Komunikace

SNMP definuje komunikační proces mezi síťovými zařízeními, které mají z hlediska protokolu specifické označení:

Manažer

Monitorovací manažer kontrolující spravovaná zařízení. V síti se může nacházet více manažerů. Odesílá zprávy typu get a set.

Spravované zařízení

Síťové zařízení, které má spuštěný SNMP protokol a vyměňuje si informace se SNMP manažerem. Manažer může do zařízení informace zapisovat a může také informace číst.

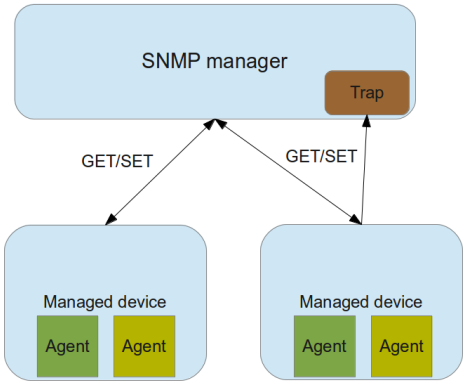
Agent

Aplikační modul SNMP protokolu, který udržuje informace o spravovaném zařízení. Agent odpovídá na požadavky a posílá je manažerovi.

Trap

SNMP podproces, který přijímá asynchronně odesílané zprávy oznamující důležitou událost na konkrétním spravovaném zařízení.

Vzájemná součinnost je znázorněna na obr. 5.1.



Obrázek 5.1: Entity SNMP protokolu. [13]

5.3 SNMP zprávy

SNMP používá ke vzájemné komunikaci mezi manažerem a spravovaným zařízením pět typů zpráv. Každá z nich vykonává jednu z následujících operací:

GetRequest

Požadavek na získání jedné nebo více hodnot od agenta.

GetNextRequest

Požadavek na získání hodnoty z další položky v OID[20] stromu.

GetBulkRequest

Požadavek na získání většího objemu dat, např. tabulky. *GetBulkRequest* je vylepšenou verzí požadavku *GetNextRequest*. Provede několik iterací požadavku *GetNextRequest*. Parametrem je počet maximálního počtu vrácených položek

SetRequest

Požadavek na nastavení jedné nebo více hodnot.

Response

Zpráva s odpovědí od agenta posílaná SNMP manažerovi.

Trap

Nevyžádaná zpráva odesílaná agentem manažerovi asynchronně při důležité události v systému.

Každý požadavek a odpověď má svůj specifický formát uvedený na obr. 5.2. Zpráva protokolu SNMP obsahuje číslo verze použitého protokolu, název SNMP komunity a datovou část. Význam polí datové části je následující:

PDU type

Typ požadavku - *GetRequest*, *GetNextRequest*, atd.

Request ID

Číselný identifikátor požadavku. Operace *Response* vrací zprávu se stejnou hodnotou Request ID.

[20] Object Identifier - Identifikátor používaný k popisu objektů.

Error

Error status je číselná hodnota chybového stavu. Error index vrací ukazatel na objekt, který chybovou zprávu vyvolal.

Enterprise

Specifický identifikátor agenta, který odeslal trap.

Agent addr

IP adresa agenta.

Generic trap

Protokolem stanovené hodnoty trapu.

Specific trap

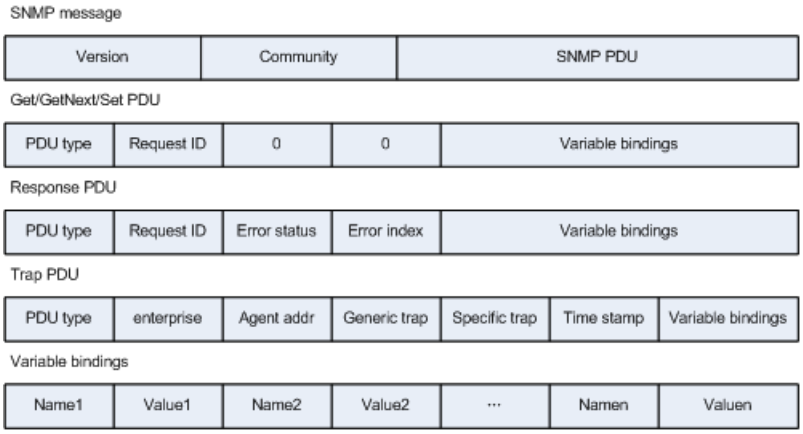
Uživatelem definovaná hodnota trapu.

Time stamp

Časové razítko důležité události.

Variable bindings

Seznam identifikátorů a souvisejících hodnot objektů.



Obrázek 5.2: Formát SNMP zpráv [2].

5.4 SMI a MIB

SNMP využívá k popisu přenášených dat jazyk SMI[21], který vychází z konvencí zavedených ve standardu ASN.1[22]. Definuje jedenáct základních datových typů, kterými lze data jednoznačně popsat syntakticky i sémanticky. Popis datových typů SMI jazyka se stručným popisem je v tabulce 5.3.

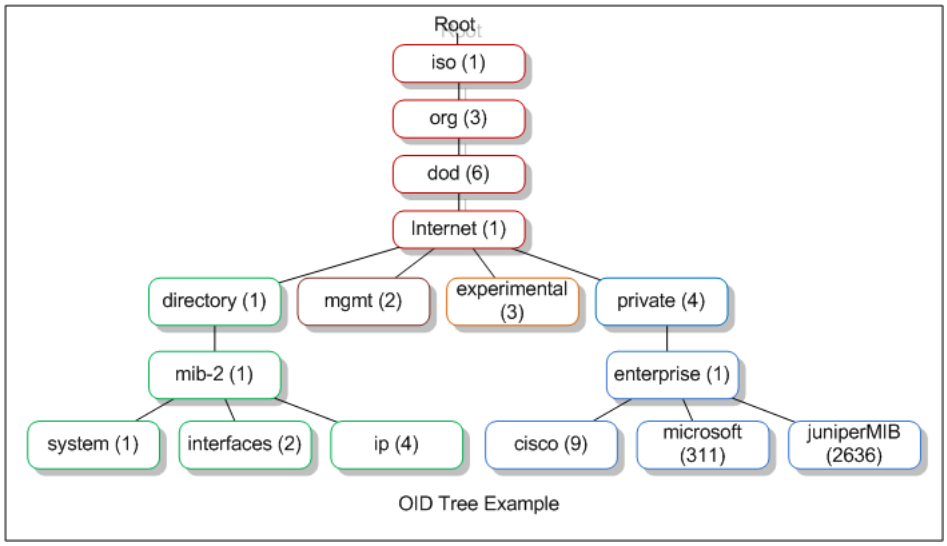
Typ	**Popis**
INTEGER	32bitová celočíselná hodnota v rozsahu -2^{31} až $2^{31}-1$. Definice přejata z ASN.1.
OCTET STRING	Řetězec bajtů o maximální délce 65536 znaků.
OBJECT IDENTIFIER	Identifikátor objektu používaný v MIB tabulkách.
Integer32	Číselná hodnota v rozsahu od -2^{31} až $2^{31}-1$.
Unsigned32	Číselná hodnota v rozsahu od 0 až $2^{32}-1$.
TimeTicks	Počet stovek milisekund od poslední události (spuštění zařízení).
Gauge32	32bitová hodnota, která při inkrementaci nepřeteče nad hodnotu $2^{32}-1$ a při dekrementaci nepodteče pod hodnotu 0.
Counter32	32bitový čítač. Při přetečení hodnoty $2^{32}-1$ nastaví hodnotu 0.
Counter64	64bitový čítač. Chování je stejné jako u 32bitového. Po přetečení hodnoty $2^{64}-1$ nastaví hodnotu 0.
IPAddress	32bitová IP adresa uložená po jednotlivých bajtech.
Opaque	Uchovávaný z důvodů zpětné kompatibility.

Tabulka 5.3: Datové typy jazyka SMI [18].

Popis dat ve formátu SMI používají tzv. MIB (Management Information Base) tabulky. MIB tabulka je databáze, která udržuje informace o objektech stromu OID. Organizace ISO zavedla hierarchické označení pro pojmenované objekty. Model hierarchické struktury je na obr. 5.3.

[21] Structure of Management Information - Podmnožina jazyka ASN.1 sloužící k popisu dat ze sledovaných zařízení protokolem SNMP.

[22] Abstract Syntax Notation One - Standard pro popis struktury dat.



Obrázek 5.3: Hierarchický strom OID [13].

Identifikátor objektu OID jednoznačně určuje umístění objektu v hierarchickém stromu. MIB tabulka popisuje datový typ instance objektu ve sledovaném zařízení. Syntaxe identifikátoru objektu má tvar číselných hodnot oddělených tečkami. Např. identifikaci privátní větve firmy Cisco odpovídá řetězec `1.3.6.1.4.1.9.0`, kde hodnota 0 za poslední tečkou označuje instanci identifikátoru objektu.

Část II

Analýza

6 Konkurenční software

V době přípravy praktické části bylo potřeba vyhledat dostupný software s obdobnou funkcionalitou. U nalezeného softwaru je třeba provést analýzu konkrétních vlastností společně s výhodami a nevýhodami. Porovnání s dostupnými nástroji by mělo přinést základní koncepci návrhu vlastní aplikace tak, aby přinášela zcela odlišný přístup a vyhovovala požadavkům zadání.

6.1 UNINETT NAV

Jedno z podobných řešení nabízí norská společnost UNINETT AS. Tato společnost se orientuje na výzkum a vývoj síťových aplikací. Jednou z aplikací je nástroj NAV, který poskytuje komplexní systém pro monitorování sítě. Systém NAV je postaven na skriptovacím jazyce Python, aplikačním frameworku Django, vykreslovací knihovně Graphite a databázovém systému PostgreSQL. Jednou z funkcí je vizualizace topologie sítě. Ve vizualizaci je možní sledovat informace o zařízeních, kudy tečou jednotlivé VLAN, zatížení linek, atd. K detekci informací o stavu sítě využívá protokoly CDP[23], SNMP a LLDP[24]. Nedetekuje však stavy STP protokolu na portech zařízení. Systém NAV má vysoké systémové nároky na vstupně-výstupní operace a čas procesoru. Nedostatkem automatického vykreslování topologie sítě je nemožnost ručního opravení mapy sítě v případě, že se vyskytnou chyby během detekce topologie sítě.

[23]Ciso Discovery Protocol - Cisco proprietární protokol pracující na linkové vrstvě ISO/OSI modelu. Slouží ke žjišťování informací o přímo pripojených sousedních prvích.

[24]Link Layer Discovery Protocol - Standardizovaný protokol pracující na linkové vrstvě ISO/OSI modelu, používaný pro šíření informací o síťových prvcích.

6.2 Cisco IOU Web Interface

Volně dostupný nástroj vytvořený italským programátorem Andreou Dainesem. Aplikaci je možné stáhnout jako hotový obraz virtuálního stroje ve formátu OVF[25], případně jako instalační balíček pro systémy Linux. Toto řešení nabízí vytvoření virtuální laboratoře, ve které může uživatel vytvořit model zapojení sítě a následně přistupovat k virtuálnímu operačnímu systému IOS, který je kompilován pro platformu Linux. Poskytuje systém správy jednotlivých laboratoří, verzí IOS a spojů v dané laboratoři. IOU web je však vytvořen spíše k výukovým účelům. Nenabízí žádné pokročilé funkce co se týče vizualizace vlastností skutečné sítě. Nevýhodou této aplikace je navíc nemožnost dynamického přidávání dalších zařízení do vytvořené laboratoře. To je řešeno úpravou načítané šablony laboratoře. [14]

7 Univerzitní síť

Počítačová síť Západočeské univerzity v Plzni je typickým zástupcem MAN[26] sítě. Více než dvě stovky přepínačů a směrovačů ve čtyřiceti budovách jsou propojeny optickými páteřními linkami. Dohromady propojují zhruba 15000 pracovních stanic a stovky serverů. Přidělený adresní rozsah univerzitní sítě je 147.228.0.0/16. Z tohoto rozsahu jsou vyčleněny podsítě pro síťová zařízení.

Na páteřních spojích jsou instalovány přepínače řady Catalyst ze sérií 6500, 4900, 4500 a 3560. Pro připojení koncových zařízení jsou přítomny přepínače řady Catalyst ze sérií 2950, 2960, 3560, 3750 a řada Nexus série 5548 s rozšiřujícími zařízeními z řady Nexus 2000.

Pro potřeby této práce je důležité, že na všech zařízeních je nakonfigurován protokol RPVST+ a možnost vzdáleného přístupu po SSH a SNMP.

[25] Open Virtualization Format - Standardizovaný formát pro distribuci softwaru, který má být spuštěn ve virtualizačním software.

[26] Metropolitan Area Network - Metropolitní síť.

8 Server

Programové vybavení je třeba nasadit na samostatný server. Jelikož je celá praktická část práce postavena na síťové komunikaci a navíc obsahuje webovou aplikaci, je vhodné použít operační systém založený na Linuxovém jádru. Operační systémy od firmy Microsoft jsou pro účely této práce nevhodné z hlediska náročnosti na výkon procesoru, spotřebu paměti a velikost diskového prostoru. Na virtuální server byl tedy nainstalován operačním systémem Debian GNU/Linux verze 7.8 (kódové označení Wheezy) s jádrem verze 3.2.68. Pro účely práce je navíc potřeba mít nainstalované balíčky následujících aplikací:

Apache 2

Velmi populární HTTP[27] server s bohatou dokumentací a snadnou konfigurací. Podporuje autentizační systém WebAuth používaný v univerzitním prostředí Orion, který ověřuje přihlašované uživatele.

PHP

Skriptovací programovací jazyk určený pro vytváření dynamických webových aplikací. *PHP*[28] skripty jsou zpracovány na straně serveru. Výstup skriptu je vkládán do struktury HTML[29] kódu.

MySQL

Databázový systém, do kterého se ukládají informace získané ze síťových zařízení a uživatelská data zadaná do webového rozhraní. Svou výkonností je databáze MySQL[30] pro účely této práce zcela dostačující.

Perl

Perl je skriptovací jazyk se specifickými vlastnostmi. Má jednoduchou syntaxi, efektivní kód (vyžaduje méně řádků kódu), širokou škálu rozšiřujících modulů a rozsáhlou dokumentaci. Silnou stránkou jazyka Perl je, že dokáže velice rychle pracovat se znakovými řetězci. Nevýhoda Perlu je jeho značná benevolentnost. Syntaxi jazyka není třeba striktně dodržovat. Absence typové

[27] HyperText Tranfer Protocol - Internetový protokol určený pro výměnu HTML dokumentů
[28] PHP: Hypertext Preprocessor - Skriptovací jazyk pro tvorbu dynamických webových stránek.
[29] HyperText Markup Language - Značkovací jazyk pro popis struktury webové stránky.
[30] MySQL - Multiplatformní relační databázový systém vyvinutý švédskou firmou MySQL AB.

kontroly ztěžuje ladění programu a může ovlivnit spotřebu paměti. Přídavné moduly mohou být instalovány z archivu CPAN[31] nebo je lze nainstalovat ručně. Příklad instalace modulu z repozitáře CPAN je ukázán ve výpisu 8.1.

```
1         root@hostname:~# cpan
2         cpan[1]> install DBD::mysql
```

Výpis 8.1: Instalace z CPAN repozitáře.

Postup ruční instalace Perl modulů je ukázán ve výpisu 8.2.

```
1 tar zxf [module name].tar.gz
2 cd [module name]
3 perl Makefile.pl
4 make
5 make test
6 make install
```

Výpis 8.2: Ruční instalace Perl modulu.

Potřebné moduly pro získávání dat ze síťových zařízení:

- SNMP - SNMP modul instalovaný s balíkem Net-SNMP.
- Net::SNMP - Původní PERL implementace objektově orientovaného rozhraní protokolu SNMP.
- DBI - Rozhraní pro komunikaci s ovladačem databázového systému. Definuje metody a proměnné nezávislé na používaném databázovém systému.
- DBD::MySQL - Ovladač databáze MySQL.
- Net::SSH::Expect - SSH modul, který volá instalovaný SSH balík v systému.

Potřebné moduly se instalují z repozitáře pomocí nástroje `cpan` z příkazového řádku způsobem uvedeným ve výpisu 8.1.

TFTP server

Síťové prvky Cisco umožňují přesměrování výstupu příkazů na TFTP[32] server, což je velice zjednodušená verze standardního protokolu FTP[33]. Stručná

[31] Comprehensive PERL Archive Network - Repozitář rozšiřujících modulů pro skriptovací jazyk Perl.

[32] Trivial File Transfer Protocol - Zjednodušená verze aplikačního protokolu FTP pro přenos dat.

[33] File Transfer Protocol - Aplikační protokol pro přenos dat.

charakteristika protokolu TFTP:

- TFTP používá nespojovaný protokol UDP na portu 69.
- Potvrzuje každý přijatý datagram.
- Nepodporuje přihlašování uživatele chráněného heslem.
- Data jsou odesílána ve formě prostého textu.

MIB tabulky

Instalace MIB tabulek umožňuje používat místo číselného identifikátoru OID názvy objektů definované v MIB tabulkách. Pro úplnost je dále uveden postup instalace MIB tabulek:

1. Nejdříve je třeba nainstalovat program pro stahování MIB tabulek. Poté zkopírovat šablonu repozitáře MIB tabulek firmy Cisco. Poté je potřeba v adresáři programu `snmp-mibs-downloader` rozbalit seznam balíků:

```
1 apt-get install snmp-mibs-downloader
2 cp /usr/share/doc/snmp-mibs-downloader/examples/cisco* /etc/snmp-
      mibs-downloader/
3 cd /etc/snmp-mibs-downloader && sudo gzip -d ciscolist.gz
```

Výpis 8.3: Instalace programu *SNMP MIBs Downloader*.

2. V souboru `/etc/snmp-mibs-downloader/snmp-mibs-downloader.conf` upravit řádky na:

```
1 BASEDIR=/var/lib/mibs
2 AUTOLOAD="rfc ianarfc iana cisco"
```

Výpis 8.4: Úprava souboru *snmp-mibs-donwloader.conf*.

Tím dojde k instalaci MIB tabulek firmy Cisco.

3. Upravit soubor `/etc/snmp-mibs-downloader/cisco.conf` na:

```
1 HOST=ftp://ftp.cisco.com
2 ARCHIVE=v2.tar.gz
3 ARCHTYPE=tgz
4 DIR=pub/mibs/v2/
5 ARCHDIR=auto/mibs/v2
6 CONF=ciscolist
7 DEST=cisco
```

Výpis 8.5: Obsah souboru *cisco.conf*.

4. Ze souboru `/etc/snmp-mibs-downloader/ciscolist` odstranit řádky obsahující řetězce:

```
1 CISCO-802-TAP-MIB
2 CISCO-IP-TAP-CAPABILITY
3 CISCO-IP-TAP-MIB
4 CISCO-SYS-INFO-LOG-MIB
5 CISCO-TAP2-CAPABILITY
6 CISCO-TAP2-MIB
7 CISCO-TAP-MIB
8 CISCO-USER-CONNECTION-TAP-MIB
```

Výpis 8.6: Odstranění řádků ze souboru *ciscolist*.

5. Stažení a instalace MIB tabulek příkazem:

```
1 sudo download-mibs
```

Výpis 8.7: Stažení MIB tabulek.

6. Instalace balíku Net-SNMP:

```
1 sudo apt-get install snmp
```

Výpis 8.8: Instalace balíku *Net-SNMP*.

7. Nakonec upravit soubor `/etc/snmp/snmp.conf` zakomentováním řádku obsahující řetězec `"mibs :"` a přidat adresář s Cisco MIB tabulkami:

```
1 #mibs :
2 mibdirs +/var/lib/mibs/cisco
```

Výpis 8.9: Úprava konfoguračního souboru *snmp.conf*.

Po spuštění programu `download-mibs` se vytváří symbolický odkaz z adresáře `/usr/share/mibs/cisco` na adresář `/var/lib/mibs/cisco`. Po zadání symbolického odkazu do souboru `snmp.conf`, jsou hlášeny chyby při spouštění programů ze sady Net-SNMP. Proto je v souboru `snmp.conf` nastavena cesta `/var/lib/mibs/cisco`.

Alternativou je stažení archivu z webového repozitáře firmy Cisco[34]. Archiv je potřeba rozbalit do některého z adresářů:

`~/.snmp/mibs`

V domovském adresáři uživatele.

`/usr/share/mibs/`

Adresář používaný balíkem Net-SNMP.

[34]Dostupné na adrese: `ftp://ftp.cisco.com/pub/mibs/v2/v2.tar.gz`

`/var/lib/mibs`

Adresář se staženými tabulkami. Na tyto adresáře je potřeba vytvořit symbolický link z adresáře `/usr/share/mibs`.

Poté stačí upravit soubor jako v bodě 7 instalace MIB tabulek.

Net-SNMP

Balík obsahující kompletní vybavení pro práci se SNMP protokolem. Obsahuje knihovny pro jazyk C/C++, moduly jazyků Perl a Python. Navíc do systému instaluje následující sadu nástrojů používaných v příkazové řádce:

snmpget

Implementuje SNMP požadavek *Get* na konkrétní instanci OID objektu.

snmpgetnext

Provede *GetNext* požadavek nad zadaným OID. Vrátí instanci následujícího objektu z OID stromu.

snmpbulkget

Musí být spuštěn s parametrem verze protokolu SNMPv2. Implementuje požadavek *GetBulk*.

snmpwalk

Implementuje požadavek *GetNext*. Prochází rekurzivně celý strom od zadaného OID do všech potomků.

snmpbulkwalk

Rozdíl oproti snmpwalk spočívá v odeslání odpovědi v jedné zprávě. Nevytěžuje síť jako snmpwalk. Nástroj snmpwalk posílá odpověď pro každou nalezenou hodnotu proměnné samostatně.

snmptable

Využívá snmpgetnext a výstup formátuje do tabulkové formy. Vhodné pro získávání objektů typu tabulka.

9 Vzdálený přístup

Zařízení firmy Cisco podporují všechny běžně používané protokoly pro vzdálený přístup. V prostředí univerzitní sítě WEBnet je možné použít nezabezpečený protokol SNMP s oprávněním pouze pro čtení a šifrovaný protokol SSH.

9.1 SNMP

Síťové prvky Cisco se chovají konzistentně v případě, kdy přijde jediný SNMP požadavek současně. Požadavek přijatý před dokončeným předchozího však nemusí být dokončen. Pokud je brán zřetel na přítomnost vícero SNMP manažerů v síti, pak je možné, že tato situace dříve či později nastane. Toto riziko bylo jedním z důvodů ke zvážení alternativního řešení vzdáleného přístupu.

Získání informací o stavu RPVST+ protokolu spočívá ve spojení dat z těchto tří objektů typu tabulka:

- `BRIDGE-MIB::dot1dBasePortTable`
- `IF-MIB::ifTable`
- `CISCO-STP-EXTENSIONS-MIB::stpxRSTPPortRoleTable`

Výchozí tabulkou je `dot1dBasePortTable`. V této tabulce jsou dvě důležité položky. První je `dot1dBasePort`, což je interní číselné označení portu, pod kterým je možné se odkazovat do tabulek, které mají spojitost s činností STP protokolu. Zejména s tabulkou `stpxRSTPPortRoleTable` popisující informace o stavu protokolu RPVST+. Druhou položkou je hodnota `dot1dBasePortIfIndex`, která je indexem čísla portu v tabulce `ifTable`. Pro lepší představivost je uvedena krátká ukázka výstupu ve výpisu 9.1.

```
snmptable -v 2c -Ci -c community 147.228.aaa.bbb BRIDGE-MIB::
    dot1dBasePortTable

SNMP table: BRIDGE-MIB::dot1dBasePortTable
 index dot1dBasePort dot1dBasePortIfIndex
  2055          2055                    7
  2179          2179                  538
  2309          2309                   13
  2311          2311                   15
  2312          2312                   16
  2313          2313                   17
  2340          2340                   44
  2433          2433                   57
```

Výpis 9.1: Výstup programu *snmptable* z tabulky *dot1dBasePortTable*.

V tabulce `ifTable` lze přečíst název portu `ifDescr` pod indexem `dot1dBasePortIfIndex`. Objekt tabulky `stpxRSTPPortRoleTable` popisuje vztah mezi VLAN, portem a rolí portu ve stavu protokolu RPVST+. Důležitá položka je hodnota `index`. Ta je složená ze dvou hodnot. Číslo před tečkou je číslo VLAN. Číslo za tečkou je index portu. Index portu odkazuje na hodnotu `dot1dBasePort` z předchozí tabulky. Příklad výstupu programu `snmptable` pro tabulku `stpxRSTPPortRoleTable` je ve výpisu 9.2.

```
snmptable -v 2c -Ci -c community 147.228.aaa.bbb CISCO-STP-EXTENSIONS-MIB::
    stpxRSTPPortRoleTable

SNMP table: CISCO-STP-EXTENSIONS-MIB::stpxRSTPPortRoleTable

    index stpxRSTPPortRoleValue
   1.2055            designated
   1.2179            designated
   1.2312            designated
   1.2313            designated
   1.2433            designated
   1.2434            designated
   1.2435            designated
   1.2437            designated
```

Výpis 9.2: Výstup programu *snmptable* z tabulky *stpxRSTPPortRoleTable*.

Opakující se hodnota čísla VLAN má svůj smysl. Do jedné VLAN, reprezentované prvním číslem, může být přiřazeno více portů a pro každou VLAN se nachází port v právě jedné roli protokolu RPVST+.

Implementace protokolu SNMP na zařízeních Cisco může způsobovat značné zpoždění při vyřizování požadavků. Zejména pak na páteřních zařízeních s vysokým počtem portů. Např. požadavek na tabulku `stpxRSTPPortRoleTable` trval v některých případech řádově desítky vteřin až jednotky minut.

Na síťových prvcích Cisco je nakonfigurován SNMP protokol verze 2. Komunikace tedy není šifrována a nedochází k ověřování požadavků vůči AAA[35] serveru. Vzhledem k potřebě získávání citlivých dat, je absence zabezpečení komunikace bezpečnostním rizikem.

9.2 SSH

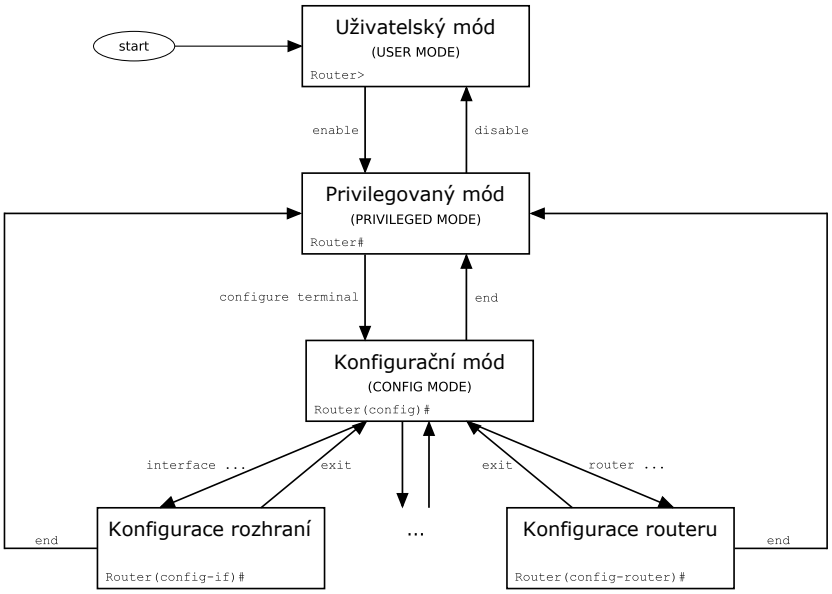
Riziko SNMP protokolu ohledně příchozího požadavku během zpracovávání předchozího požadavku s použitím SSH protokolu odpadá, jelikož SSH vytváří vyhrazený komunikační kanál. Stejně tak je eliminováno riziko odposlechnutí komunikace. Pro získání informací o stavu STP, je třeba odeslat patřičné příkazy a přijímat postupně odpověď od dotazovaného zařízení.

Výhodou protokolu je univerzálnější přístup ve smyslu získávání užitečných dat. Pokud by měla být data získávána ze síťových zařízení od jiných výrobců než od produktů firmy Cisco, muselo by dojít k úpravě rozsáhlé části kódu, která manipuluje s daty získávanými z různých objektů v OID stromu. Protokol SSH poskytuje textový výstup komunikace. Výrobci se zpravidla snaží dodržovat strukturovaný textový výstup příkazů, který lze snadno zpracovávat s použitím regulárních výrazů. Úprava kódu pak spočívá v přidání dalších regulárních výrazů, které budou získaná data zpracovávat.

Operační systémy IOS, pro řadu Catalyst, a NX OS, pro řadu Nexus, jsou ovládány pevně definovanou množinou slovních příkazů zadávaných do příkazové řádky. Konkrétní příkazy mohou být spouštěny v určitých módech operačního systému.

[35] Authentication, Authorization, Accounting - Autentizační, autorizační a účtovací protokol.

Příkazy lze zadávat i ve zkrácené formě, pokud zkrácená forma nemá jiný sémantický význam. Příkazy musí být jednoznačně identifikované. Např. příkaz `sh int` je zkrácenou verzí příkazu `show interfaces`. Před samotným vykonáním příkazu, je zkrácená forma expandována do plného tvaru a až poté vykonána. Přechody mezi módy s potřebnými příkazy jsou znázorněny na obr. 9.4.



Obrázek 9.4: Módy operačních systémů Cisco. [3]

Následuje stručná charakteristika jednotlivých módů:

Uživatelský mód

Poskytuje omezený počet příkazů. Příkazy jsou převážně informativního charakteru.

Privilegovaný mód

Poskytuje širší množinu příkazů. Poskytují detailnější informace o stavu zařízení.

Globální konfigurační mód

Příkazy provedené v globálním konfiguračním módu ovlivňují chování zařízení. Provedené změny se ukládají v aktuálně běžící konfiguraci. Změny je třeba uložit do non-volatilní inicializační paměti před vypnutím nebo restartem zařízení. Veškeré neuložené změny jsou během vypnutí smazány. Při startu je načítána konfigurace právě z inicializační paměti.

Detailní konfigurace

Možnost specifikovat detailní nastavení různých parametrů. Např. rychlost portů, parametry směrovacích protokolů, názvy VLAN, STP, atd.

Následující příklady ukazují potřebné příkazy, a jejich výstupy, ke shromáždění informací o stavu STP protokolu pro všechny VLAN na každém portu síťového zařízení. Příkaz `show vlan brief` vrací seznam všech VLAN v interní paměti přepínače a názvy portů, které jsou do konkrétní VLAN přiřazeny. Ukázka výstupu je ve výpisu 9.3.

```
1  Switch# show vlan brief
2
3  VLAN Name              Status    Ports
4  ---- ----------------- --------- -------------------------------
5  1    default           active    Te1/6, Te1/7, Te1/8, Te2/1,
6                                   Te2/2, Te2/3, Te2/4, Te2/5,
7  7    vlan-name-7       active
8  8    vlan-name-8       active
9  9    vlan-name-9       active
10 10   vlan-name-10      active    Gi3/1, Gi3/2, Gi3/3, Gi3/4,
11 13   vlan-name-13      active
12 14   vlan-name-14      active    Gi3/15
13 ..
```

Výpis 9.3: Výstup příkazu *show vlan brief*.

Příkaz `show interfaces status` z výpisu 9.4 vzpisuje stav všech portů.

```
1  Switch# show interfaces status
2
3  Port     Name   Status     Vlan   Duplex  Speed Type
4  Te1/1/1         connected  routed   full    10G 10Gbase-CX4
5  Te1/1/2         connected  routed   full    10G 10Gbase-CX4
6  Te1/1/3         connected  trunk    full    10G 10Gbase-SR
7  Te1/1/4         connected  trunk    full    10G 10Gbase-SR
8  ...
```

Výpis 9.4: Výstup příkazu *show interfaces status*.

Na operačním systému NX OS je nutno zadat příkaz `show interface status`.

Dále je ve výpisu 9.5 uveden příklad výpisu stavu STP protokolu na konkrétním rozhraní. Výstup poskytuje informace o stavu RPVST+ protokolu pro všechny VLAN.

```
Switch# show spanning-tree interface Port-channel 76

Vlan       Role Sts Cost Prio.Nbr Type
---------- ---- --- ---- -------- ----
VLAN0001   Desg FWD 1    128.5764 P2p
VLAN0003   Desg FWD 1    128.5764 P2p
VLAN0044   Root FWD 1    128.5764 P2p
VLAN0047   Root FWD 1    128.5764 P2p
VLAN0048   Root FWD 1    128.5764 P2p
VLAN0049   Root FWD 1    128.5764 P2p
```

Výpis 9.5: Výstup příkazu *show spanning-tree interface*.

Příkaz `show spanning-tree root` poskytuje informace o kořenovém přepínači protokolu PVRST+. Pokud je hodnota `Root Cost` rovna nule, jedná se o kořenový přepínač pro danou VLAN. Z výpisu 9.6 je též patrné, že na kořenovém přepínači se nenachází root port.

```
Switch# show spanning-tree root

                                  Root Hello Max Fwd
Vlan             Root ID          Cost  Time Age Dly  Root Port
---------- -------------------- ---- ----- --- ---  ---------
VLAN0001    8193 0008.e3ff.fc2c    0    2   20  15
VLAN0003   16387 0008.e3ff.fc2c    0    2   20  15
VLAN0027   12315 0008.e3ff.fc20    1    2   20  15  Po8
VLAN0040   16424 0008.e3ff.fc2c    0    2   20  15
VLAN0041   12329 0008.e3ff.fc20    1    2   20  15  Po8
VLAN0042   12330 0008.e3ff.fc20    1    2   20  15  Po8
```

Výpis 9.6: Výstup příkazu *show spanning-tree root*.

U přepínačů s vysokým počtem portů a VLAN mohou být výpisy velmi dlouhé. Proto je ve výchozím nastavení výpis přerušen po určitém řádcích a čeká na stisk mezerníku nebo enteru. Počet řádků je stanoven dynamicky po vytvoření spojení s emulátorem koncového terminálu. Z hlediska vzdáleného přístupu je toto chování nežádoucí. Musela by s ověřovat poslední vrácená řádka, která v případě přerušeného výpisu má tvar `-More-`. Toto chování lze změnit zadáním příkazu `terminal length 0`, kde hodnota 0 znamená vypnutí přerušování výpisu, jak je uvedeno ve výpisu 9.7.

```
Switch# terminal length 0
```

Výpis 9.7: Nastavení nepřerušovaného výpisu terminálu.

Obnovení přerušování výpisu do výchozího stavu se provádí podle výpisu 9.8.

```
Switch# terminal no length
```

Výpis 9.8: Obnovení výchozího přerušování výpisu terminálu.

Operační systémy IOS od verze 12.2(13) i NX OS umožňují přesměrování výstupu příkazu na ostatní připojené periferie (flash paměť, NvRAM[36]) případně odeslání dalšími protokoly (FTP, HTTP, HTTPS[37], SCP[38], TFTP). Příklad přesměrování výstupu na TFTP server v operačním systému IOS je ve výpisu 9.9.

```
1 Switch# show interfaces status | redirect tftp://10.0.0.1/output.txt
```

Výpis 9.9: Ukázka přesměrování výstupu příkazu na TFTP server.

[36]Non-volatile Random Access Memory - Paměť s náhodným přístupem, která po odpojení napájení neztrácí uloženou informaci.

[37]HyperText Tranfer Protocol Secure - Šifrovaná podoba protokolu HTTP.

[38]Secure Copy Protocol - Protokol pro šifrovaný přenos dat pomocí protokolu SSH.

10 Návrh databáze

Pro účely práce plně dostačuje relační databáze MySQL. Tento databázový systém nabízí podporu transakcí, zamykání tabulek a paralelního přístupu. Tyto vlastnosti je vhodné využívat v co možná nejvyšší míře tak, aby byla zachována konzistence uložených dat. Při návrhu databáze byla brána v potaz následující fakta:

1. Jediná databáze bude muset uchovávat informace ze síťových zařízení a společně s informacemi z webové aplikace, které zadá uživatel.
2. Potřeba vyřešit vztahy mezi entitami:

   Síťové zařízení - port

   Na každém boxu se může jméno portu vyskytovat právě jednou. Jedno jméno portu se může vyskytovat na více zařízeních.

   port - VLAN

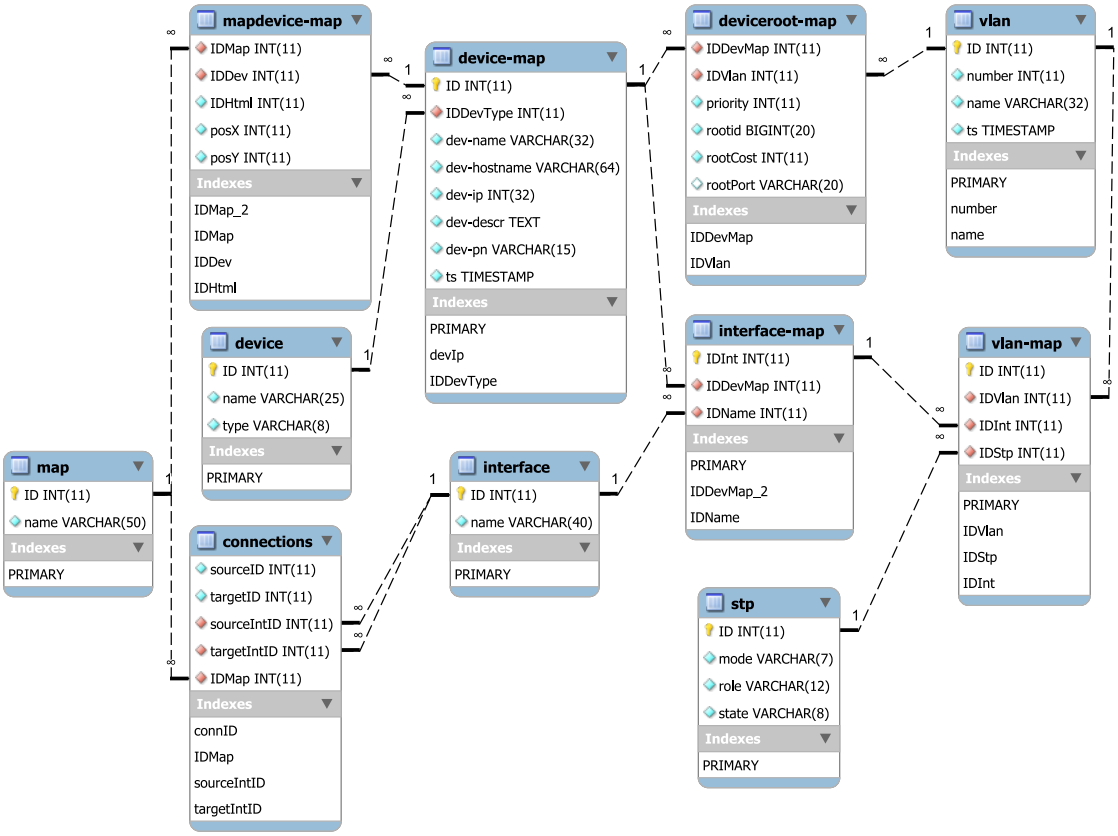
   Vztah mezi jedním konkrétním portem na daném zařízení a VLAN, které na daném portu prochází. V případě trunk spojů, je pro jeden port potřeba uchovat vztah s vícero VLAN. Access porty mohou být přiřazeny pouze do jedné datové VLAN a do jedné hlasové VLAN.

   port - RPVSTP+

   Na jediném portu je pro každou VLAN spuštěna instance protokolu RPVST+, který se může nacházet v právě jednom stavu.
3. Uživatel, který vytvoří schéma sítě ve webovém prohlížeči, si jej uloží do databáze pod názvem mapy. S předpokladem, že na jednom zařízení se může jméno portu nacházet právě jednou, pak může existovat pouze jedno spojení z portu na jednom zařízení do portu druhého zařízení. Nemůže se tedy stát, že budou existovat dva spoje mezi dvěma zařízeními a obě spojení budou mít na souhlasných koncích stejná jména portů.
4. Vytvořená spojení z webové aplikace uloží vztah mezi jmény portů a konkrétními zařízeními, která jsou daným spojením propojeny.
5. V každé mapě může být dané zařízení pouze jednou. V jedné mapě je každé zařízení jednoznačně určeno IPv4 adresou a hodnotou `id` v HTML kódu.

6. Do databáze budou síťová zařízení ukládána právě jednou a to ze dvou důvodů:
   - Jednodušší manipulace při získávání dat ze sítě. Skript, který se bude připojovat k jednotlivým zařízením se nebude muset připojovat k témuž zařízení vícekrát.
   - Jednodušší manipulace s daty v databázi.

ERA diagram navržené databáze je na obr. 10.5.



Obrázek 10.5: ERA model databáze.

Navržená databáze má dohromady 11 tabulek. Následuje stručný popis uchovávaných informací v tabulkách:

connections

Záznamy o uživatelských spojeních mezi zařízeními v konkrétní mapě.

device

Produktové řady jednotlivých zařízení.

device-map

Uživatelem nastavená zařízení ve webové aplikaci.

deviceroot-map

Informace o kořenových přepínačích protokolu STP pro jednotlivé VLAN.

interface

Seznam detekovaných jmen portů.

interface-map

Uchovává vztah mezi fyzickým zařízením, jménem portu. Primární klíč tabulky je cizím klíčem v tabulce vlan-map.

map

Jména map, které uložil uživatel.

mapdevice-map

Uchovává vztah mezi mapou a zařízeními, která jsou v této mapě.

stp

Kombinace stavů STP protokolu.

vlan

Detekovaný seznam VLAN, které se nacházejí na jednotlivých zařízeních.

vlan-map

Spojuje dohromady informace o portu zařízení (cizí klíč z tabulky interface-map), VLAN na portu a stav STP protokolu pro konkrétní VLAN.

Nezávisle na použitém způsobu paralelizace kódu, bude ovlivněno také čtení a zapisování do databáze. Proto je nutné tabulky zamykat a všechny databázové operace provozovat jako transakce. Transakce jsou často využívány v situacích, kdy je potřeba udržet integritu databáze, při přechodu mezi konzistentními stavy databáze. Transakce se navenek tváří jako jedna atomická operace. Navíc musí dodržovat čtyři zásadní vlastnosti:

Atomicita

Transakce je dále nedělitelná. Provádí se jako jeden ucelená operace anebo vůbec. V případě, že není operace dokončena, je vrácena chybová hláška.

Konzistence

Během transakce by se měla databáze chovat vždy konzistentně. Pokud má transakce měnit data na vícero místech databáze, neměl by mít jiný uživatel přístup k aktuálním datům v jedné části a ke starým datům v jiné části databáze.

Izolovanost

Transakce by měly být prováděny samostatně a neměly by se ovlivňovat. Proto by měly být řazeny do fronty a zpracovávány po sobě.

Trvalost

Potvrzení transakce má za následek zapsání změn do databáze.

11 Programové vybavení

11.1 Jádro

Požadavkem zadavatele bylo vytvoření jádra v jazyce Perl. Program bude potřebovat modul pro komunikaci s databází a modul pro vytváření SSH spojení. Vzhledem k počtu zařízení v prostředí univerzitní sítě, je třeba vytvářet více paralelních připojení k různým zařízením. V případě, že vlákna nebo podprocesy přistupují během vykonávání programu ke společné proměnné, musí být deklarována jako sdílená proměnná.

Paralelizaci kódu je možné vyřešit několika způsoby. Prvním je použití vláken implementovaných v jazyce Perl. Jedná se o modul `threads`. Obsluha vláken je obdobná jako v jazyce C/C++. Při vytváření vlákna je v argumentu předán odkaz na obslužný podprogram vlákna a vstupní argument. Řízení běhu a získávání informací o stavu vláken je možné ovládat běžnými metodami[39], které jsou známé z jiných programovacích jazyků. Pokud jsou používána vlákna jazyka Perl, pak je třeba použít též moduly, které jsou tzv. Thread-safe. Thread-safe moduly jsou navrženy

[39]Bližší popis Perl vláken na stránce: `http://perldoc.perl.org/threads.html`

tak, aby vykonávání kódu bylo možné i při paralelním běhu vláken. Druhou variantou je použití modulu `Parallel::ForkManager`. Ten vytváří podprocesy interpretu jazyka Perl na úrovni operačního systému. Množství vytvářených podprocesů je potřeba omezit, aby nedošlo k zaplnění paměti a následnému pádu programu.

Modul použitý pro komunikaci se síťovými zařízeními musí podporovat odesílání více příkazů po sobě, jelikož je potřeba zjistit o jednom zařízení informace plynoucí z různých příkazů. Jeden z nalezených modulů `Net::SSH2::Channel` má implementovanou metodu `exec()`, které je jako parametr předán odesílaný příkaz ve formě textového řetězce. Po vykonání této metody se uzavře komunikační kanál a pro odeslání dalšího příkazu je třeba kanál opět vytvořit. Dalším modulem je `Net::SSH::Expect`. Ten obaluje instalovaný systémový nástroj SSH (jedná se o tzv. *wrapper*). Tento modul dovoluje odesílat libovolný počet příkazů a navíc lze tento modul použít v paralelních podprocesech.

11.2 Webová aplikace

Webová aplikace by měla poskytovat uživateli jednoduché a přehledné rozhraní. Aplikace by měla dodržovat specifikace doporučené ve standardu HTML5. Požadavek zadavatele byl takový, že si uživatel vytvoří mapu sítě, kam přidá síťová zařízení a propojí porty obou zařízení podle potřeby. Po vytvoření mapy sítě si může uživatel mapu uložit. Dále má možnost vybrat jednu z VLAN a zvýraznit tak spojení, kterými daná VLAN prochází. K tomu budou poskytovány informace o stavu protokolu STP na jednotlivých portech.

Jelikož vytvořená aplikace by se měla na straně klienta chovat dynamicky, bude potřeba vybrat vhodné nástroje. V dnešní době je dostupná řada knihoven napsaná v jazyce JavaScript. Jazyk JavaScript je široce rozšířený multiplatformní, objektově orientovaný skriptovací jazyk používaný k ovládání interaktivních prvků webové stránky. Z velkého množství dostupných knihoven byly pro vytvoření webové aplikace vybrány následující knihovny.

11.2.1 jQuery

Velice populární knihovna vydávaná pod licencí MIT. Nabízí programátorovi sadu funkcí, která usnadňuje práci s jazykem JavaScript. Využitelné jsou následující funkce:

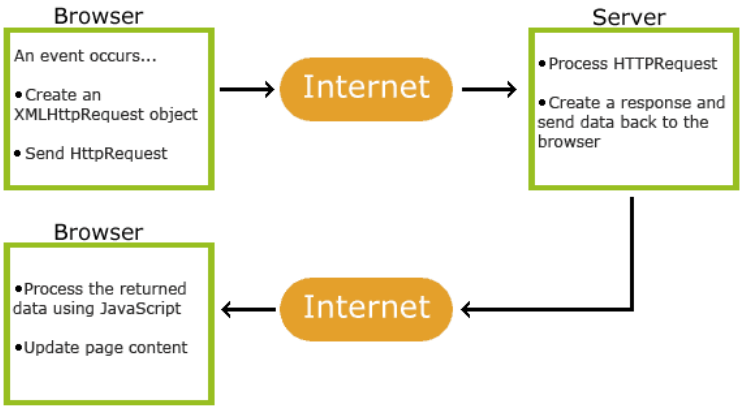
- Výběr DOM[40] elementů postavený na selektorech jazyka CSS[42]. *jQuery* navíc umožňuje manipulovat s DOM elementy ve stromu HTML kódu.
- Úprava vlastností hodnot elementů CSS.
- Editování atributů HTML elementů.
- Ukládání aplikačních dat o DOM objektu na pozadí.
- Úprava obsluhy událostí dle vlastních potřeb.
- Animace a různé efekty.
- Rozšířitelnost použitím pluginů.
- AJAX[43] pro dynamické nahrávání obsahu stránek. AJAX není jazyk ale obecná technika použití webových technologií HTML, JavaScript a XMLHttpRequest[44]. Odeslání požadavku vzniká zpravidla po nastání nějaké události v prohlížeči na straně klienta. Prohlížeč vytvoří požadavek typu XMLHttpRequest. Ten je odeslán na server. Na straně serveru je požadavek zpracován skriptem (nejčastěji *PHP* skript). Výstupní data skriptu jsou odeslány v odpovědi na příchozí požadavek XMLHttpRequest. Na straně klienta dojde ke zpracování odpovědi a aktualizaci obsahu webové stránky. Schéma AJAXu je ukázáno na obr. 11.6.

[40]Document Object Model - Objektově orientovaná reprezentace HTML nebo XML[41] dokumentů.

[42]Cascading Style Sheets - Jazyk popisující zobrazení elementů jazyků (X)HTML a XML v prohlížeči.

[43]Asynchronous JavaScript and XML - Technologie pro asynchronní nahrávání obsahu do webových stránek.

[44]Rozhraní, které umožňuje komunikaci klienta se serverem pomocí protokolu HTTP.



Obrázek 11.6: Princip technologie AJAX. [4]

11.2.2 jQuery UI

Knihovna *jQuery UI* je vyvíjena jako plugin pro knihovnu *jQuery*. Přidává další funkce zaměřené na rozšíření interaktivity uživatelského rozhraní (GUI[45] widgety). Poskytuje rozšíření efektů a animací.

11.2.3 jQuery contextmenu

Plugin do knihovny *jQuery* umožňující vytváření kontextových nabídek pro jednotlivé elementy webové stránky. Umožňuje vytváření různých kontextových nabídek v závislosti na typu elementu.

11.2.4 jsPlumb

Knihovna *jsPlumb* je navržena pro vytváření grafických spojů v podobě čar a křivek mezi elementy ve webové stránce. Spolupracuje s knihovnou *jQuery*. Základním konceptem knihovny *jsPlumb* je objekt `Connection`. Objekt `Connection` je vnitřně sestavený z těchto komponent:

[45] Graphical User Interface - Grafické uživatelské rozhraní.

Anchor

Pozice, na které se může vyskytovat komponenta `Endpoint` vzhledem k nadřazenému HTML elementu. `Anchor` není nikde viditelný. Jedná se pouze o logické umístění `Endpoint`u.

Endpoint

Graficky reprezentuje koncový bod komponenty `Connector`. Lze je vytvářet dynamicky na libovolných elementech HTML kódu a propojovat je komponentou `Connector`.

Connector

Graficky reprezentuje čáru, propojující dva elementy. `Connector` má dva koncové body `Endpoint`. Knihovna *jsPlumb* umožňuje vytvořit čtyři druhy spojů - přímou linii, Bézierovu křivku, spoj využitelný pro vývojové diagramy a spoj pro vytváření modelů stavových automatů.

Overlay

`Overlay` umožňuje specifikovat grafické vlastnosti komponenty `Connector`. Zajímavou funkcí je vytváření popisků jednotlivých spojení.

Knihovna *jsPlumb* dále poskytuje navázání událostí (event binding) na komponenty `Connection`, `Endpoint` a `Overlay`.

11.2.5 select2

Knihovna používaná pro vylepšení vlastností HTML elementu select. Umí používat technologii AJAX a dynamicky tak načítat obsah položek. Užitečnou funkcí je též filtrování vyhledávaného obsahu.

11.2.6 UIkit

Odlehčený framework pro vytváření jednoduchých uživatelských rozhraní. Poskytuje sadu komponent jazyka HTML, CSS a JavaScript, se kterými lze rychle a snadno vytvářet přehledné interaktivní webové aplikace. Použití frameworku je řešeno apli-

kací specifických tříd na konkrétní elementy v těle HTML dokumentu. Navíc eliminuje rozdíly ve vykreslování stránek v různých prohlížečích. Důvodem pro volbu frameworku *UIkit* je, že oproti konkurenčním řešením (Bootstrap od Twitteru nebo Foundation od ZURB) nabízí rozsáhlejší sadu komponent při úspornějším kódu.

12 Cron

Potřebu periodického spouštění úloh řeší systémový nástroj Cron, který umožňuje spouštění úloh automatizovat. Čas spuštění úlohy se na operačním systému Debian konfiguruje v souboru `/etc/crontab`. Syntaxe souboru je uvedena ve výpisu 12.1.

```
1         *   *   *   *   *   command to be executed
2         -   -   -   -   -
3         |   |   |   |   |
4         |   |   |   |   +----- day of week (0 - 6) (Sunday=0)
5         |   |   |   +--------- month (1 - 12)
6         |   |   +------------- day of month (1 - 31)
7         |   +----------------- hour (0 - 23)
8         +--------------------- min (0 - 59)
```

Výpis 12.1: Syntaxe souboru *crontab*.

Znak hvězdičky `*` pro daný sloupec znamená, že jsou přípustné všechny hodnoty uvedené v závorce. Ovlivnění spouštění každý *n-násobek* časové jednotky se zapisuje za lomítkem. Např. spuštění Perl skriptu každou dvacátou minutu lichých hodin v pondělí a pátek:

```
1 */20   1-23/2   *   *   1,5   perl /home/user/script.pl > /dev/null
```

Výpis 12.2: Ukázka naplánované úlohy nástroje *Cron*.

V případě určování lichých hodin, je třeba uvést interval od první do poslední požadované hodiny a potřebný krok intervalu o délce dvou hodin za lomítkem. Výčtové hodnoty se oddělují čárkou.

Část III

Realizace

13 Jádro

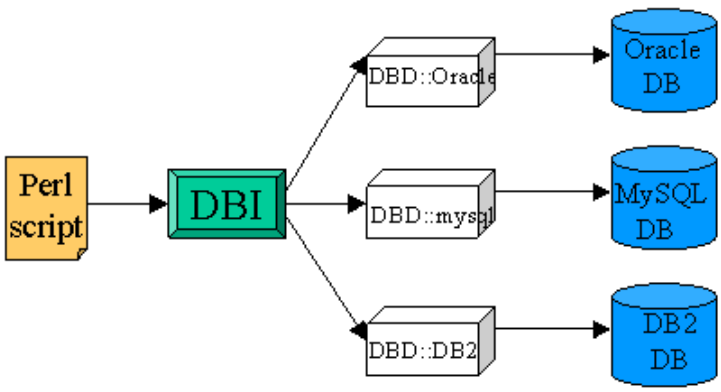
V této části uvádím použité programové konstrukce a ukázky použití modulů jazyka Perl. Přidávání rozšiřovacích modulů do zdrojového kódu skriptu se provádí direktivou `use` následovanou jménem modulu (v příkladu bez uvozovek) podle výpisu 13.1.

```
1 use "ModuleName1";
2 use "ModuleName2";
```

Výpis 13.1: Přidání Pelr modůlů do programu.

13.1 MySQL

Pro komunikaci s databází MySQL je třeba použít modul `DBI`. Základní schéma vysvětlující komunikaci modulu `DBI` s ovladači databází je na obr. 13.7.



Obrázek 13.7: Schéma použití modulu DBI. [5]

Pro vytvoření spojení s databází je nejdříve nutné nastavit parametry rozhraní `DBI`. Příklad je ve výpisu 13.2.

```
1     my $dbname    = "jmenoDatabaze";
2     my $hostname  = "localhost";
3     my $dbUser    = "dbUser";
4     my $dbPwd     = "dbVeryStrongPassword"
5     my $dbConf    = "DBI:mysql:database=$dbname;host=$hostname;port=3306"
6     my $dbh       = DBI->connect($dbConf, $dbUser,
7                          $dbPwd,
8                          {
9                              'RaiseError' => 1,
10                             'AutoCommit' => 0
11                         }
12                    ) or die $DBI::errstr;
```

Výpis 13.2: Vytvoření spojení s databází.

Metoda `connect()` navazuje spojení s databází. Prvním parametrem je řetězec specifikující použitý ovladač databáze, hostname systému a TCP[46] port databáze. Pokud není číslo portu explicitně uvedeno, použije se výchozí hodnota 3306. Druhým parametrem je uživatelské jméno, pod kterým se přihlašuje do databáze. Třetím parametrem je heslo uživatele. Posledním parametrem jsou nastavitelné volby chování databáze. V tomto případě je nastaveno ukončení skriptu chybovou zprávou v případě, že nelze navázat spojení s databází. Druhou volbou je vypnutí automatického provedení ukládání do databáze. Nastavením volby `AutoCommit` na hodnotu 0 je umožněno používání transakcí. Ukončení spojení s databází se provádí metodou `disconnect()` podle výpisu 13.3.

```
1 $dbh->disconnect();
```

Výpis 13.3: Odpojení od databáze.

Odesílání SQL[47] dotazů do databáze vyžaduje dva kroky. Nejdříve je potřeba předpřipravit databázový dotaz metodou `prepare()`. Tato metoda vrací manipulátor příkazu, tzv. statement handler. Manipulátor je nutné použít pro vykonávání metody `execute()` jako ve výpisu 13.4.

[46] Transmission Control Protocol - Spojovaná služba transportní vrstvy ISO/OSI modelu. Garantuje spolehlivé doručení dat.

[47] Structured Query Language - Strukturovaný dotazovací jazyk používaný v relačních databázích pro práci s daty.

```
1 my $sth = $dbh->prepare("SELECT * FROM table WHERE value1 = ? AND value2 =
      ?");
2 $sth->execute(x, y);
3 $sth->finish();
```

Výpis 13.4: Předpřipravení SQL dotazu.

Výhodou tohoto přístupu je možnost přehledné přípravy všech dotazů na začátku programu a pak dotazy spouštět metodou `execute()` na různých místech programu. Výše uvedený program ukazuje použití SQL dotazu s použitím parametru reprezentovaného otazníkem. Parametry jsou SQL dotazu předávány v metodě `execute()` ve stejném pořadí, v jakém jsou uvedeny otazníky v SQL dotazu. Nakonec je třeba ukončit statement a uvolnit paměť metodou `finish()`. Další možností spouštění dotazů je metoda `do()` z výpisu 13.5

```
1 $dbh->do("INSERT INTO table VALUES(17, "hostname")");
```

Výpis 13.5: Vykonání MySQL dotazu metodou *do()*.

Metoda `do()` je nepoužitelná pro dotazy typu SELECT. Tato metoda totiž nevrací statement handler, kterým je možno načíst vrácená data.

Použití transakcí vyžaduje další speciální programovou konstrukci využívající blok `eval{}`. Tento blok umí odchytávat různé chyby a výjimky při vykonání kódu. To znamená, že program při výskytu chyby v tomto bloku neskončí. Při výskytu jakékoliv výjimky program nastaví hodnotu speciální proměnné `$@`. Příklad použití bloku `eval{}` společně s mechanismem transakce je ukázán ve výpisu 13.6.

```
1  eval {
2      $sth = $dbh->prepare("INSERT INTO table VALUES(?, ?)");
3      $sth->execute(x, y);
4      $sth->finish();
5      $dbh->commit();
6  }
7  if ($@) {
8      $sth->finish();
9      $dbh->rollback();
10 }
```

Výpis 13.6: MySQL dotazu v transakci.

Nejdříve je vytvořen statement handler SQL dotazu, poté vykonán a řádně ukončen. Pokud se nevyskytne žádná chyba během zpracování SQL dotazu, provede se metoda `commit()` a změny v databázi budou uloženy. Pokud se vyskytne chyba, bude

nastavena hodnota proměnné `$@` a následně zavolána metoda `rollback()`, která zruší všechny změny provedené v databázi vykonané v bloku `eval{}`.

13.2 SNMP

Protože byl v konečné implementaci použit pouze protokol SSH, budou zde uvedeny základní části kódu, které získávají informace o stavech protokolu STP na jednotlivých portech.

Vytvoření nové SNMP relace vyžaduje hostname monitorovaného zařízení, jméno komunity, číslo UDP portu protokolu SNMP, použitou verzi protokolu SNMP. Užitečné je vypnutí překládání objektů typu OCTET STRING do formy čitelné pro člověka nastavením parametru `-octetstring => 0`. V posledním řádku výpisu je uvedeno ukončení SNMP relace:

```
1 ($session, $error) = Net::SNMP->session(
2          -hostname  => deviceHostname
3          -community => communityString
4          -port      => 161
5          -version   => 'snmp2c'
6          -translate => [-octetstring => 0]
7          );
8
9 $session->close();
```

Výpis 13.7: Vytvoření SNMP relace.

Programové zpracování dat ze SNMP protokolu spočívá ve správném zacházení s OID identifikátory. Při procházení OID stromu mezi objekty různých tabulek je potřeba zpracovávat hodnoty z požadovaných sloupců a hodnot indexů instancí jednotlivých položek.

Potřebné OID identifikátory stromu jsem ukládal v jednom asociativním poli (neboli hash). Klíč k OID hodnotě je jméno OID v MIB tabulkách:

```
my %oids = (
   'ifDescr'              => '1.3.6.1.2.1.2.2.1.2',
   'dot1dBasePort'        => '1.3.6.1.2.1.17.1.4.1.1',
   'dot1dBasePortIfIndex'=> '1.3.6.1.2.1.17.1.4.1.2',
   'vtpVlanName'          => '1.3.6.1.4.1.9.9.46.1.3.1.1.4',
   'stpxRSTPPortRole'     => '1.3.6.1.4.1.9.9.82.1.12.2.1.3'
);
```

Výpis 13.8: Asociativní pole uchovávající OID identifikátory.

Získání dat z objektu typu tabulka se provádí metodou `get_entries()`.

```
my $resultDot1D = $session->get_entries(-columns =>
    [
        $oids{dot1dBasePort},
        $oids{dot1dBasePortIfIndex}
    ]);
```

Výpis 13.9: Získání dat z objektu typu tabulka.

Metoda `get_entries()` implementuje SNMP požadavek `getBulkRequest`. Parametr `-columns` specifikuje jaký sloupec má být požadován ze vzdáleného objektu. Pro každou hodnotu OID je vyslána zpráva s požadavkem na získání hodnot konkrétního OID.

K průchodu hodnot v získaném sloupci je použit cyklus `foreach`. Před samotným průchodem se musí získané objekty lexikograficky seřadit podle OID identifikátorů. K tomu slouží rutina `oid_lex_sort(@list)`.

```
my $index;
my $BasePortIfIndex;
my $DescrIndex;
my $DescrIndexName;

foreach my $oid (grep /^$oids{dot1dBasePort}\./, oid_lex_sort(keys(%$resultDot1D)))) {
    ($index) = $oid =~ m|\.([0-9]+)$|;
    $BasePortIfIndex = $session->get_request(-varbindlist => [$oids{dot1dBasePortIfIndex} . "
        ." . $index]);
    $DescrIndex = $oids{ifDescr} . "." . $BasePortIfIndex->{$oids{dot1dBasePortIfIndex} . "."
         . $index};
    $DescrIndexName = $session->get_request(-varbindlist => [$DescrIndex]);
}
```

Výpis 13.10: Získání souvisejících dat z více tabulek.

V těle cyklu je třeba nejdříve získat hodnotu indexu položky v sloupci získané z objektu tabulky `dot1dBasePortTable`. K rozpoznání čísla indexu je využit regulární výraz, který zkopíruje čísla za poslední tečkou po konec řádku. Index je poté připo-

jen za řetězec OID pro získání hodnot ve sloupci `dot1dBasePortIfIndex`. Metodou `get_request()` je získána hodnota indexu do tabulky `ifDescr` a uložena do proměnné `$BasePortIfIndex`. Hodnota z proměnné `$BasePortIfIndex` je připojena k řetězci OID `ifDescr` a následně získán řetězec jména portu metodou `get_request`.

Obdobný postup je při získávání hodnot z tabulky `stpxRSTPPortRoleTable`. Hodnota indexu ze sloupce `dot1dBasePort` v tabulce `dot1dBasePortTable` je hodnota části indexu pro položky v tabulce `stpxRSTPPortRoleTable`. V této tabulce je nutné zpracovat hodnotu indexu regulárním výrazem a dostat tak příslušnou VLAN, do které port patří. Získání všech VLAN, které jsou na daném portu přenášeny je třeba použít funkci `grep`, která vybere všechny OID, kde na místě indexu VLAN bere všechny hodnoty, které jsou následované indexem `dot1dBasePort`. Potřebný regulární výraz je uveden ve výpisu 13.11.

```
1    grep /^$oids{stpxRSTPPortRole}\.([0-9]+)\.$index$/
```

Výpis 13.11: Filtrování výstupu tabulky *stpxRSTPPortRoleTable*.

13.3 SSH

K vytvoření SSH spojení je k dispozici několik modulů. Z praktického hlediska této práce je potřeba použít takový modul, který lze použít v paralelizovaném kódu a navíc umí odesílat více požadavků po sobě. Modulem, který splňuje oba předchozí požadavky je `Net::SSH::Expect`.

Ke komunikaci po SSH je potřeba nejdříve vytvořit novou instanci komunikačního kanálu. Nutné parametry jsou adresa vzdáleného síťového zařízení, uživatelské jméno, timeout a parametr `raw_pty`. Nastavením posledního parametru se lokální linka přepne do tzv. raw režimu. Odstraní se tak některé problematické funkcionality v různých implementacích SSH klientů. Ukázka vytvoření nového SSH spojení je ve výpisu 13.12.

```
$ssh = Net::SSH::Expect->new(
       host     => ipAddress,
       user     => userName,
       password => userPassword,
       timeout  => 3,
       raw_pty  => 1
       );
```

Výpis 13.12: Vytvoření nového SSH spojení.

Po vytvoření SSH kanálu, je potřeba se ke vzdálenému zařízení připojit. Přihlášení se provádí metodou `login()`. Příklad kódu je ve výpisu 13.13.

```
$login_output = $ssh->login();
    if ($login_output =~ /[\/a-zA-Z0-9._-]+ ?(?:\(config[^)]*\))? ?[#>] ?$
        /)) {
        # login uspesny - odesilej prikazy
    }
```

Výpis 13.13: Přihlášení ke vzdálenému SSH serveru s ověřením výstupu.

Úspěšné přihlášení vrátí prompt vzdálené konzole. K ověření úspěšného přihlášení lze využít regulární výraz, který zkontroluje hodnotu uloženou v proměnné `$login_output`. Regulární výraz popisuje možné tvary promptu síťových zařízení Cisco.

Vykonávání příkazů na vzdáleném zařízení umožňuje metoda `exec($command)`. Metoda vrací výstupní text po vykonání příkazu `$command` na vzdáleném hostovi jako jeden dlouhý řetězec. Odeslání příkazu na síťové zařízení je ukázáno ve výpisu 13.14.

```
$output = $ssh->exec("show vlan brief", 1);
@lines = split(/^/, $output);
```

Výpis 13.14: Vykonání příkazu a následné rozdělení textu do pole řádků.

Obdržený výstup z metody `exec()` je uložen do proměnné `$output`. Před dalším zpracováním je text uložený v proměnné `$output` rozdělen regulárním výrazem na řádky a uložen do pole řetězců.

13.4 Paralelizace

Vytváření podprocesů má na starosti modul Parallel::ForkManager. Vytvoření instance manažeru je ve výpisu 13.15.

```
1  my $MAX_PROCESSES = 10;
2  my $pm = new Parallel::ForkManager($MAX_PROCESSES);
3  foreach $arrayItem (@array) {
4      my $pid = $pm->start and next;
5
6      # kod programu, ktery je~vykonavan ve~vytvorenem podprocesu
7      # dalsi kod programu
8
9      $pm->finish;
10 }
11 $pm->wait_all_children
```

Výpis 13.15: Příklad práce s modulem *Parallel::ForkManager*.

Při vytváření instance `Parallel::ForkManager` je třeba omezit maximální počet vytvořených podprocesů. Ve výsledném kódu byl zvolen maximální počet podprocesů na 10. Tento počet vytvářených podprocesů je dostatečný z hlediska počtu zařízení, ze kterých jsou získávána data. Stejně tak nedochází k přetížení procesoru a zaplnění operační paměti serveru, na kterém celá aplikace běží.

V cyklu `foreach` jsou postupně vytvářeny nové podprocesy příkazem `$pm->start`. Příkaz vrací hodnotu 0 do procesu potomka a hodnotu PID pro proces rodiče. Kód `and next` přeskočí vykonávanou smyčku v rodičovském procesu a spustí další podproces, dokud nedojde ke spuštění povoleného počtu podprocesů.

Příkaz `$pm->finish` ukončí podproces po skončení vlastního kódu. Čekání na dokončení všech podprocesů zajišťuje příkaz `$pm->wait_all_children`. Jedná se o blokující čekání. Program tedy nepokračuje dál, dokud všechny podprocesy nedokončí svůj program. Pak může následovat vykonávání dalšího kódu v rodičovském procesu.

14 Webové rozhraní

Celá aplikace je řešená jako jedna statická stránka, do které jsou dynamicky nahrávány další elementy popisující stav reálného stavu STP protokolu v univerzitní síti. Jedním z požadavků zadavatele bylo, aby aplikace poskytovala dva módy. Prvním je editační mód, ve kterém si uživatel „nakliká“ mapu sítě a tu poté uloží. Druhým módem je tzv. „view“ mód, ve kterém bude zvýrazněna zvolená VLAN a stav STP protokolu na jednotlivých portech pro zvolenou VLAN.

Stránka je logicky členěna do následujících částí:

Navigační menu

Poskytuje uživateli přístup k načítání, ukládání a mazání vytvořených map. Umožňuje přidávat do vytvářené mapy nové elementy reprezentující síťová zařízení. Součástí navigačního panelu je i barevná legenda zobrazená během vykreslování stavu STP protokolu.

Mapa sítě

Obdoba kreslícího plátna, ve kterém si může uživatel vytvořit model sítě. V HTML elementu mapy sítě jsou dynamicky vytvářeny objekty reprezentující síťová zařízení.

Ovládací panel

Obsahuje strukturovaný výpis podrobných informací o stavu STP protokolu na jednotlivých portech zařízení, která jsou zanesena v mapě sítě.

Do zdrojového kódu HTML stránky jsou přidány CSS soubory vizuálního stylu a potřebné skripty napsané v jazyce JavaScript.

14.1 JavaScript knihovny

Chování celé aplikace je naprogramováno v jazyce JavaScript. Navíc byly použity další knihovny, které pomohly řešit jednotlivé části celkové funkcionality aplikace. Dále bude uveden popis použití jednotlivých knihoven.

14.1.1 jQuery

Přednostní využití knihovny *jQuery* sloužilo k vytváření nových elementů v HTML kódu a obsluha nastalých událostí (kliknutí na element, zpracování formulářů). Princip vytvoření nového elementu reprezentujícího síťové zařízení v mapě sítě je ukázáno ve výpisu 14.1.

```
1  $("<div></div>", {
2     'id': n,
3     'class': className,
4     'css': {
5        'top': "50px",
6        'left': "50px"
7     },
8     data: {
9        "hostname": devHostname,
10       "ipAddress": devIp,
11       "devType": devType,
12       "interfaces": []
13    }
14 }).appendTo($("#map1"));
```

Výpis 14.1: Vytváření nových elementů pomocí knihovny jQuery.

Z předchozího výpisu je viditelné, že pokud je do funkce `$()` předán řetězec HTML elementu, dojde k vytvoření nového DOM objektu v paměti a následně na něj mohou být aplikovány jakékoliv metody knihovny *jQuery*. Zde například připojení vytvořeného DOM objektu do elementu s HTML identifikátorem `map1`. S použitím programové konstrukce uvedené ve výpisu 14.1 jsou vytvářeny všechny objekty reprezentující síťové zařízení. Zajímavou funkcí je přiřazování aplikačních dat libovolného typu vytvořenému DOM objektu. Ve vytvořeném DOM objektu tak lze uchovávat potřebná data, která jsou nutná pro zajištění správné funkcionality aplikace ale není třeba je nikde zobrazovat. Reálně vypadající výsledný HTML kód je uveden ve výpisu 14.2.

```
<div id="node0" class="device l3switch" style="top: 50px; left: 50px;">
   <div class="dev">
      <img class="dev-img" src="images/l3switch.png" alt="l3switch0">
   </div>
   <div class="dragConn" id="dragConn0">
   </div>
   <div class="dev-label uk-text-center " id="hostname0">
      <div class="dev-hostname uk-text-center uk-text-small" id="input0"
          contenteditable="true">ic-sw</div>
   </div>
</div>
```

Výpis 14.2: HTLM kód elementu síťového zařízení vytvořený pomocí jQuery.

Element `<div class="dragConn"id="dragConn0»</div>` je důležitý pro další knihovnu *jsPlumb*. Pomocí tohoto elementu je možné vytvářet spojení mezi dalším elementem síťového prvku. Bližší popis vytváření spojení je uveden v části 14.1.6 popisující používání knihovny *jsPlumb*. Ukázka výsledného objektu vykresleného prohlížečem je na obr. 14.8.

Obrázek 14.8: Vytvořený DOM objekt síťového zařízení vykreslený prohlížečem.

14.1.2 jQuery UI

Knihovnu *jQuery UI* bylo třeba použít kvůli možnosti posouvání elementů síťových prvků po mapě (tzv. dragging). Nastavení vlastnosti `draggable` DOM objektu je ukázáno ve výpisu 14.3.

```
$("#node0").draggable();
```

Výpis 14.3: Nastavení vlastnosti *draggable*.

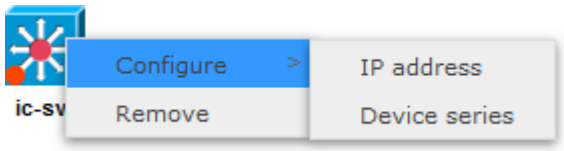
14.1.3 jQuery contextmenu

Nastavování potřebných parametrů síťového zařízení v mapě je vyřešeno implementací kontextového menu. K vytváření jednoduchých kontextových nabídek je určen plugin *contextmenu* pro knihovnu *jQuery*. Krátká ukázka kódu pro vytvoření kontextové nabídky o dvou položkách je ve výpisu 14.4.

```
$(function(){
   $.contextMenu({
      selector: '.elementClass',
      callback: function(key, options) {
         // vykonání kódu po vyvolání kontextové nabídky
      },
      items: {
         "edit": {name: "Edit", icon: "edit"},
         "cut": {name: "Cut", icon: "cut"},
      }
   });
});
```

Výpis 14.4: Vytvoření kontextové nabídky pomocí pluginu *jQuery contextmenu*.

Reálně vypadající kontextová nabídka pro element síťového zařízení je na obr. 14.9.



Obrázek 14.9: Kontextové menu elementu síťového zařízení.

Položky v podmenu *Configure* vyvolají dialogová okna pro zadání příslušných hodnot. Tyto hodnoty jsou pak uloženy jako datové objektu *jQuery*. Položka *Remove* odstraní celý element síťového zařízení z mapy a odstraní případné spojení, vytvořená knihovnou *jsPlumb*, připojená k odstraňovanému objektu. Kontextová nabídka je přiřazena též spojením vytvářeným pomocí knihovny *jsPlumb*. Nabídka však obsahuje pouze jednu položku pro odstranění daného spojení z mapy.

14.1.4 select2

Knihovna *select2* poskytuje rozšíření funkcionality standardního elementu `<select>`. Pro účely této práce je využito vyhledávání mezi položkami elementu `<select>`. Inicializace knihovny *select2* nad požadovaným elementem `<select>` se provádí podle výpisu 14.5.

```
$(".selectElementClass").select2({
width: 100,
  ajax: {
     url: "loadMapNames.php",
     dataType: 'json',
     data: function (term, page) {
        return {
           'name': term.term
        };
     },
     results: function (data, page) {
        return {
           'results': data
        };
     }
  }
}));
```

Výpis 14.5: Inicializace knihovny *select2*.

Ve výpisu 14.5 je ukázána technika načítání položek vysíláním AJAX požadavku, který je zpracovávaný *PHP* skriptem `loadMapNames.php`. V těle AJAX požadavku je uveden parametr s názvem `'name'` a hodnotou `term.term`. Hodnota `term.term` je řetězec zadaný do vyhledávacího pole elementu `select`. Data vrácená ze serveru jsou formátována do JSON[48] objektu. Pojmenování vráceného JSON objektu musí odpovídat řetězci `'results'`, jinak nedojde k rozparsování obdržených dat. Filtrování obsahu položek pracuje na základě postupného zadávání znaků do textového pole. Položky v elementu `select` jsou postupně filtrovány podle zadávaného retězce.

14.1.5 UIkit

Upravování uživatelského rozhraní s frameworkem *UIkit* spočívá v přidávání specifických HTML tříd elementům, na které má být vizuální styl aplikován. Framework *UIkit* jsou využity následující komponenty:

Grid

Rozděluje prostor webové stránky na jednotlivá pole v pevně daném poměru.

Panel

Ohraničuje logicky ucelené částí webové aplikace. K panelu je vždy přidán nadpis charakterizující účel použití panelu.

[48] JavaScript Object Notation - Způsob zápisu dat nezávislý na počítačové platformě. Syntaxe vychází ze skriptovacího jazyka JavaScript.

Utility

Implementuje drobné korekce zarovnání obsahu stránky, nastavení marginu a paddingu jednotlivých elementů.

Form

Stylizace formulářů, kterými uživatel nastavuje načítaný obsah stránky.

Accordion

Komponenta accordion vytváří rozbalovací elementy, které po kliknutí na hlavičku elementu odkryjí další obsah uvnitř elementu.

Table

Sjednocuje styl tabulek, do kterých jsou vypisovány informace o stavu STP protokolu na jednotlivých portech.

Icon

Během čekání na vyřízení AJAX požadavků jsou zobrazování animované ikony kvůli informovanosti uživatele o činnosti na pozadí webové aplikace.

Text

Usnadňuje formátování textu.

Notifier

Zobrazuje potvrzení, upozornění a chybové hlášky uprostřed horního okraje stránky. Uživatel je tímto způsobem informován, zda došlo např. ke korektnímu uložení mapy do databáze. Úroveň zpráv je barevně odlišena podle důležitosti.

Jelikož je použití celého frameworku *UIkit* ve výsledném kódu rozsáhlé, bude dále uveden pouze jeden příklad použití komponenty *Grid*. Výpis 14.6 ukazuje rozdělení do dvou sloupců v poměru 8:2. Další obsah jednotlivých sloupců může být libovolný.

```
1 <div class="uk-grid">
2    <div class="uk-width-8-10">...</div>
3    <div class="uk-width-2-10">...</div>
4 </div>
```

Výpis 14.6: Příklad použití komponenty *Grid* frameworku *UIkit*.

Notifikace o úspěšnosti dokončení AJAX požadavku je třeba vyvolávat programově. Krátká ukázka kódu je ve výpisu 14.7.

```
$.ajax({
   url: "loadVlan.php",
   type: "POST",
   dataType: "JSON"
}).done(function (data) {
   // zpracovani vracenych dat
   UIkit.notify({message: "Success.", status: 'success', timeout: 750});
}).fail(function () {
   UIkit.notify({message: "Error.", status: 'danger', timeout: 750});
}).always(function () {
   // vzdy provedeny kod bez ohledu
   // na~chybu v~prubehu AJAX pozadavku
});
```

Výpis 14.7: Programové vyvolání *Notify* komponenty frameworku *UIkit*.

Další demonstrační ukázky použití komponent je k nalezení na webových stránkách projektu *UIkit*[49].

14.1.6 jsPlumb

K vizuálnímu spojení dvou elementů síťových prvků je využívána knihovna *jsPlumb*. Před samotným propojováním elementů musí být provedena prvotní inicializace nastavení knihovny tak, aby její chování odpovídalo požadavkům. Po prvotní inicializaci základního nastavení lze vytvářet spojení mezi elementy dvěma následujícími způsoby:

Ručně

Ruční vytváření spojení je používáno v editačním módu, kdy je vytvářena nová mapa sítě, nebo upravována již načtená mapa. Ruční vytváření spojení vyžaduje buď nastavení počátečního a koncového objektu *Endpoint* popsaného v části 11.2.4 nebo použít metody *makeSource* a *makeTarget* na požadované elementy, na kterých mají být objekty *Endpoint* dynamicky vytvořeny při vytvoření spojení. V aplikaci je použitý druhý postup. Ten má výhodu v tom, že není nutné vytvářet žádné další elementy ručně.

Jako zdrojový element pro vytvoření spojení mezi zařízeními je použito červené kolečko v levém dolním rohu elementu reprezentující síťový prvek. Kliknutím

[49] Webová adresa dokumentace frameworku *UIkit* - `http://getuikit.com/docs/documentation_get-started.html`

na červené kolečko a tažením myši se stisknutým levým tlačítkem se natáhne spojení, které lze vytvořit puštěním tlačítka myší nad dalším elementem síťového zařízení. Po vytvoření spojení se objeví formulář (upravený *UIkit*em), kde je uživatel vyzván k zadání jmen propojených portů na obou zařízeních. Programově je ošetřena nemožnost propojení zařízení se sebou samým. V praxi není důvod tato spojení vytvářet.

V příloze je ukázán postup vytvoření spojení mezi dvěma prvky. Na obr. A.1a je samotné zařízení bez spojení. Obr. A.1b je zobrazuje přesouvání volného konce spojení ve směru šipky na druhé zařízení bez dalších spojení. Obr. A.1c ukazuje vytvořené spojení s textovými popiskami zadanými v dialogovém okně.

Programově

Programové vytváření spojení je podstatně jednodušší. K programovému vytvoření spojení slouží metoda `connect({source: "e1", target: "e2"})`. Parametry této metody jsou jednoznačné HTML identifikátory dvou objektů, které mají být propojeny. Tento postup je používán při načítání mapy z databáze. Každému elementu síťového zařízení je přiřazen unikátní HTML identifikátor a následně vytvořená spojení mezi zařízeními.

Každému spojení je navíc přiřazen textový popis obou konců spojení. V těchto popisech je znázorněno jméno portu, do kterého je dané spojení ve fyzické síti skutečně zapojeno. Programové vytvoření spojení a přiřazení popisu je ukázáno ve výpisu 14.8.

```
1 var connection = jsPlumb.connect({
2    source: "node2",
3    target: "node5",
4    overlays: [
5       ["Label", {id: "label1", label: "Te1/1", location: 0.15}],
6       ["Label", {id: "label2", label: "Te1/3", location: 0.85}]
7    ]
8 });
```

Výpis 14.8: Vytvoření spojení a použití komponentoy *Overlay* knihovny *jsPlumb*.

S použitím knihovny *jsPlumb* jsou implementovány další funkcionality jako např. zobrazení popisků spojení při najetí ukazatelem myši na spojení, změna barvy

a tloušťky čáry spojení, změna barvy a velikosti objektu *Endpoint* nebo načtení všech spojení před uložením do databáze.

14.2 Komunikace mezi klientem a serverem

Dynamické načítání obsahu stránky zajišťuje použití AJAX požadavků odesílaných z prohlížeče klienta na server. Vyřízení požadavků provádějí krátké *PHP* skripty. Všechny *PHP* skripty, které přistupují do databáze, načítají přihlašovací informace z konfiguračního souboru `config.php`, ve kterém je uloženo přihlašovací jméno a heslo, hostname počítače, na kterém běží MySQL server a jméno databáze, do které má být přistupováno. Veškeré SQL dotazy do databáze jsou zpracovávány jako transakce. Následuje stručný popis činnosti skriptů vyřizujících AJAX požadavky:

deleteMap.php

Smaže z databáze mapu sítě.

hostnameTranslate.php

Po zadání IP adresy zařízení ve webovém rozhraní provede překlad IP adresy na doménové jméno. Ve webovém rozhraní je poté nahrazen řetězec `hostname` doménovým jménem.

interfaces.php

Hledá v databázi jména portů, která jsou zadávána ve formuláři při navazování spojení. Vrací filtrovaný seznam jmen portů podle zadaného řetězce ve formuláři při vytváření spojení.

loadIntStp.php

Pro přijaté číslo VLAN načítá stav a roli portů protokolu STP na všech zařízeních v mapě. Dále hledá v databázi kořenový přepínač pro číslo VLAN přijaté v požadavku. Pokud není nalezen kořenový přepínač pro danou VLAN, je vrácen přepínač, který je nejbližší ke kořenovému přepínači (má nejnižší hodnotu *RootCost*). Vrácené informace vypíše do tabulek v ovládacím panelu aplikace. A v mapě označí oranžovým rámečkem kořenový přepínač, případně nejbližší přepínač ke kořenovému obarví modrou barvou.

loadMap.php

Načte všechna potřebná data o mapě. V odpovědi předává informace o zařízeních v mapě a informace o jednotlivých spojeních.

loadMapNames.php

Obdoba skriptu *interfaces.php*. Hledá v databázi jména uložených map, která jsou zadávána ve formuláři. Vrací seznam jmen map podle zadaného řetězce ve formuláři při načítání požadované mapy

loadStp.php

Pro zadanou VLAN vrací seznam spojení, na kterých se daná VLAN nachází. Na koncových bodech spojení, kterými daná VLAN prochází, je přidána informace o stavu a roli portu v instanci STP protokolu.

loadVlan.php

Načte seznam všech detekovaných VLAN z univerzitní sítě. Vrácený seznam je vypsán do elementu `select` pro výběr VLAN, pro kterou má být zobrazen stav protokolu STP.

saveDevices.php

Ukládá do databáze data o vytvořené mapě.

Data přenášená mezi serverem a klientem jsou vždy uložena ve formátu JSON. Formát JSON se dá velice rychle vytvořit i procházet programově. V jazyce *PHP* i v knihovně *jQuery* lze s JSON daty snadno zacházet. Ve výpisu 14.9 je ukázána struktura dat ve formátu JSON. Konkrétně se jedná o dvě zařízení propojená spojením.

```
{
   "webnet": {
      "dev": [
         {
            "ID": "17",
            "dev-hostname": "ic-sw",
            "dev-ip": "147.228.aaa.bbb",
            "IDHtml": "0",
            "type": "l3switch",
            "posX": "333",
            "posY": "385"
         },
         {
            "ID": "19",
            "dev-hostname": "ek-sw",
            "dev-ip": "147.228.aaa.ccc",
            "IDHtml": "2",
            "type": "l3switch",
            "posX": "320",
            "posY": "670"
         }
      ],
      "con": [
         {
             "sourceID": "0",
             "sourceIntName": "Te1/2/3",
             "targetID": "2",
             "targetIntName": "Te1/5"
         }
      ]
   }
}
```

Výpis 14.9: Data ve formátu JSON.

14.3 Vizualizace STP

Po přepnutí do „view“ módu je vyslán AJAX požadavek na získání seznamu všech dostupných VLAN. Zvolením konkrétní VLAN, jsou odeslány další dva AJAX požadavky na obdržení informací z databáze o stavu STP.

První požadavek se dotazuje na stav všech portů v mapě, které jsou k dané VLAN přiřazeny. Vrácen je seznam spojení v mapě, u kterých se alespoň jeden port přiřazen do požadované VLAN. Ukázka dvou vrácených spojení ve VLAN 302 společně se stavy každého portu je ve výpisu 14.10.

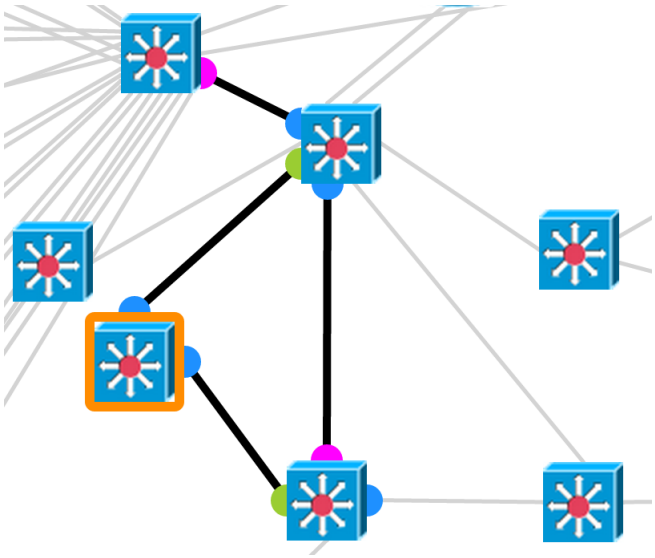
```
{
    "302": [
       {
          "con": {
             "srcID": "0",
             "srcIntName": {
                "label": "Te1/2/3",
                "state": "FWD",
                "role": "Desg"
             },
             "tgtID": "2",
             "tgtIntName": {
                "label": "Te1/5",
                "state": "FWD",
                "role": "Root"
             }
          }
       },
       {
          "con": {
             "srcID": "0",
             "srcIntName": {
                "label": "Po8",
                "state": "FWD",
                "role": "Desg"
             },
             "tgtID": "3",
             "tgtIntName": {
                "label": "Po8",
                "state": "FWD",
                "role": "Root"
             }
          }
       }
    ]
}
```

Výpis 14.10: Spojení v mapě se stavy portů protokolu STP.

Pokud se oba porty nacházejí v definovaném stavu pro danou VLAN, znamená to, že komunikace z dané VLAN tímto spojem prochází a spoj je následně zvýrazněn a objekty `Endpoint` jsou zvýrazněny příslušnou barvou symbolizující stav portu v instanci protokolu STP. Pokud nejsou porty na obou koncích spojení přiřazeny do stejné VLAN, zůstane spojení nezvýrazněné. Barva použitá na zvýraznění stavu portu je pro každý stav jiná. Zvoleny byly takové barvy, které dobře kontrastují s okolím a hlavně mezi sebou. Následuje popis obarvení jednotlivých stavů portů:

Root	YellowGreen
Designated	DodgerBlue
Alternating	Magenta
Backup	Pink
Inconsistent	Red

Ukázka zvýraznění výsledného stromu STP je na obr. 14.10.



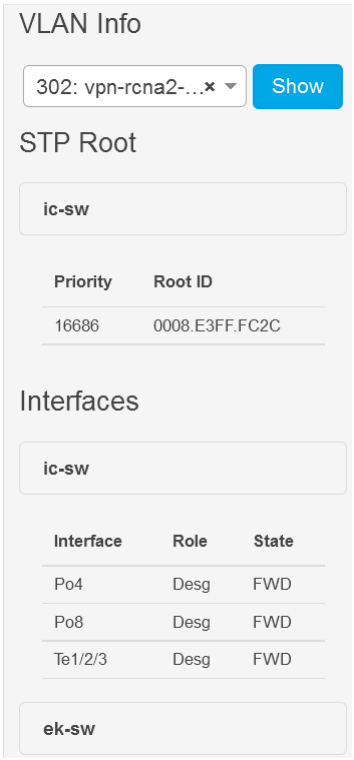
Obrázek 14.10: Zobrazení stavu STP.

Druhý požadavek se dotazuje na získání informací o stavu každého síťového zařízení s porty aktivními v dané VLAN, které jsou v mapě zapojeny. Další získávaná data jsou informace o kořenovém přepínači pro danou instanci STP. Pokud pro danou VLAN v mapě existuje kořenový přepínač, jsou vrácena data související s protokolem STP (priorita a RootID přepínače). Pokud v mapě sítě neexistuje kořenový přepínač pro danou VLAN, jsou vrácena data o přepínači, který se nachází nejblíže kořenovému přepínači (má ze všech přepínačů nejnižší hodnotu RootCost). Ukázka objektu JSON s informacemi o zařízeních, na kterých jsou porty v konkrétní VLAN a o kořenovém přepínači je ve výpisu 14.11.

```
{
   "302": {
      "dev": {
         "ic-sw": {
            "Po4": {
               "role": "Desg",
               "state": "FWD"
            },
            "Po8": {
               "role": "Desg",
               "state": "FWD"
            },
            "Te1/2/3": {
               "role": "Desg",
               "state": "FWD"
            }
         },
         "ek-sw": {
            "Te1/1": {
               "role": "Altn",
               "state": "BLK"
            },
            "Te1/2": {
               "role": "Desg",
               "state": "FWD"
            },
            "Te1/5": {
               "role": "Root",
               "state": "FWD"
            }
         }
      },
      "root": {
         "ic-sw": {
            "ip": "147.228.aaa.bbb",
            "IDHtml": "0",
            "priority": "16686",
            "rootID": "0008.E3FF.FC2C",
            "rootCost": "0",
            "rootPort": ""
         }
      }
   }
}
```

Výpis 14.11: Objekt JSON pro podrobný výpis dat.

Ukázka podrobného výpisu informací o zařízeních pro zvolenou VLAN je na obr. 14.11 Výsledná aplikace je ukázána v příloze A.4.



Obrázek 14.11: Výpis podrobných informací o stavu STP.

15 Cron

Periodické spouštění skriptu obstarává linuxový démon Cron. Časový plán spouštění jednotlivých úloh je definován v souboru `/etc/crontab`.

V souboru `/etc/crontab` je přidána úloha spouštění skriptu:

```
1 15 8,9,10,11,12,13,14,15,16,17 * * * root perl /var/scripts/stpVlan.pl > /
      dev/null
```

Výpis 15.1: Příkaz v souboru */etc/crontab*.

Skript je spouštěn každý den od 8 hodin a 15 minut do 17 hodin a 15 minut v hodinových intervalech.

16 Bezpečnost

Jelikož je celá aplikace určena výhradně pro potřeby zaměstnanců LPS CIV[50], je potřeba řešit též otázku zabezpečení přístupu pouze pro autorizované uživatele.

16.1 Firewall

Pro potřeby celé aplikace je ve firewallu serveru povolena komunikace pro následující síťové služby:

HTTPS

Přístup je povolen pouze na zabezpečený server HTTPS naslouchající na standardním portu 443. Přístup na port 80 je automaticky přesměrován na port 443. Na HTTPS serveru je nasazena webová aplikace společně s nástrojem `phpMyAdmin` pro správu databázového serveru.

SSH

Povoluje připojení k serveru a síťovým zařízením zabezpečeným protokolem SSH na portu 22. Povoleny jsou také adresní rozsahy síťových administrátorů pracoviště LPS CIV.

TFTP

Služba TFTP, která poslouchá na UDP portu 69, slouží pro příjem textových souborů z přesměrovaných výstupů síťových zařízení. Povoleny jsou pouze adresní rozsahy, ve kterých se nacházejí síťové prvky.

ICMP

Povoleno pro nástroj `ping`, sloužící k ověření dostupnosti virtuálního serveru.

Dále jsou otevřeny porty interních služeb pracoviště LPS CIV. Komunikace na ostatních TCP a UDP portech je firewallem blokována.

[50] Laboratoř počítačových systémů na Centru informatizace a výpočetní techniky při Západočeské univerzitě v Plzni.

16.2 WebAuth

Webový server Apache má doinstalovaný modul `mod_webauth`. Modul `mod_webauth` přesměruje uživatele na na WebKDC server, který daného uživatele autentizuje. V konfiguračním souboru Apache serveru je nastavena skupina uživatelů, která může na webovou aplikaci přistupovat.

16.3 TACACS+

Zabezpečení přístupu na síťové prvky je zvýšeno ověřováním přihlašovaného uživatele vůči TACACS+ serveru. V konfiguraci TACACS+ serveru je třeba nakonfigurovat seznam zařízení, ke kterým má uživatel povolený přístup. V následujícím výpisu 16.1 je ukázán příklad konfigurace autorizovaného přístupu k zařízením. Viditelné je využití regulárních výrazů k pokrytí doménových jmen skupiny zařízení.

```
1 acl = MP_dip {
2    nas dns =~ "ic-vss6509-[gs]w\\.zcu\\.cz$"
3    nas dns =~ "u[cn]-idf[1234][ab]-cat4510-sw\\.zcu\\.cz$"
4    nas dns =~ "us-n[12]-nx5548-sw\\.zcu\\.cz$"
5 }
```

Výpis 16.1: Přístup k zařízením.

Povolení vykonávání skupiny jednotlivých příkazů pro konkrétního uživatele se konfiguruje podle výpisu 16.2. Ve skupině příkazů pojmenované `MP-dip` je třeba nejdříve přiřadit úroveň oprávnění do takové úrovně, ve které je možné vykonávat potřebné příkazy. Hodnota `priv-lvl` zajistí přepnutí do privilegovaného režimu po přihlášení uživatele. Omezení množiny autorizovaných příkazů (zde `terminal`, `show`, `exit` a `logout`) se provádí regulárními výrazy. Pro každý příkaz jsou ve složených závorkách uvedeny přípustné parametry příkazu.

```
group = MP-dip {
   default service = deny
   service = shell {
      default command = deny
      set priv-lvl = 15
      cmd = terminal {
          permit length.*
      }
      cmd = show {
         permit vlan.*
         permit span.*
         permit int.*status
         deny .*
      }
      cmd = exit {
         permit .*
      }
      cmd = logout {
         permit .*
      }
   }
}
```

Výpis 16.2: Autorizované příkazy uživatele.

Uživatel, kterému je povolen přístup, má v závorkách uložen otisk přihlašovacího hesla a členství v access listu. Access list určuje vztah mezi povolenými příkazy a síťovými zařízeními, na kterých může povolené příkazy spustit, tak jak je uvedeno ve výpisu 16.3

```
user = vmandak {
   login = crypt ...
   pap = crypt ...
   member acl MP_dip = MP-dip
}
```

Výpis 16.3: Autorizovaný uživatel s přihlašovacími údaji.

17 Zhodnocení výsledků

V této kapitole jsou uvedeny poznatky, které byly získány při závěrečném testování výsledné aplikace.

17.1 Získávání dat

Připojování k síťovým zařízením pomocí protokolu SSH bylo časově ovlivněno hodnotou timeoutu. Skript, který získával data ze síťových zařízení současně měřil i časové prodlevy logických částí kódu. Odesílání požadavků společně se získáním odpovědi trvalo řádově stovky milisekund. Pokud mělo být na jedno zařízení odesláno více příkazů, byl výsledný čas vykonávání zdržen o prodlevu timeoutu pro každý příkaz.

17.2 Databáze

Zvolený databázový systém MySQL plně dostačuje potřebám práce. Ukládání dat ze síťových zařízení do příslušných tabulek trvalo v nejhorších případech řádově desítky milisekund. Integrita dat v databázi byla zajištěna použitím transakcí a integritních omezení klíčů v tabulkách.

17.3 AJAX

Jelikož je interaktivita webové aplikace z velké části založena na technologii AJAX, je dobré zde zmínit i výsledky dosahované při odesílání AJAX požadavků na server. Zpracování AJAX požadavků na straně serveru vykonává SQL dotazy nad databází. Přestože jsou používány složité SQL dotazy obsahující několikanásobné propojení více tabulek klíčovým slovem `INNER JOIN`, tak jsou data vrácena opět v řádech desítek milisekund. Delší prodlevy se vyskytovaly během konverze vrácených dat z databáze do formátu JSON. Konverze vyžaduje programovou manipulaci s aso-

ciativními poli. V průběhu testování aplikace jsem nezaznamenal výrazné zpomalení vlivem vyřizování AJAX požadavků. Během zpracovávání AJAX požadavků se v aplikaci zobrazují příslušné animované spinnery informující uživatele o nově načítaném obsahu.

17.4 Webová aplikace

Webové rozhraní by mělo vyhovovat požadavkům zadavatele z funkčního hlediska. Samotný grafický design stránky už vychází z mého uvážení. Vysoký důraz byl kladen na přehlednost, jednoduchost a intuitivnost ovládání. Aplikace byla testována v následujících prohlížečích:

- Internet Explorer 9+
- Mozilla Firefox
- Google Chrome
- Safari
- Opera

Tyto prohlížeče byly v době vývoje aplikace podporovány jak samotnými výrobci, tak i vývojáři webových technologií. Prohlížeče Internet Explorer ve verzi 9 a vyšší, vykazovaly drobné nepřesnosti při vykreslování objektů knihovny *jsPlumb*. Prohlížeče Google Chrome, Safari a Opera, postavené na open source jádře *WebKit*[51], tyto problémy neměly.

Zajímavým výsledkem bylo odhalení nepřesností v konfiguraci některých přepínačů. V síti existovaly pro některé VLAN více než jeden přepínač, který byl označen jako Root Bridge pro jednu instanci RPVST+ protokolu, přestože by v síti měl být pro každou VLAN právě jeden kořenový přepínač. Grafické znázornění této situace je v příloze A.2.

[51] Webové stránky projektu *WebKit* - `https://www.webkit.org/`.

18 Možnosti rozšíření

Potencionálních možností pro další rozšiřování funkčnosti je několik. Získávaná data ze síťových zařízení se nemusí týkat pouze funkčnosti protokolu STP. Komunikace po SSH protokolu umožňuje odesílat prakticky libovolné příkazy a zpracovávat jejich výstupy. Další získávané informace by mohly být:

Směrovací tabulky

Ze zařízení pracujících na L3 vrstvě ISO/OSI modelu by bylo zajímavé získávat směrovací tabulky. Daly by se tím odstranit některé problémy se směrováním komunikace v síti.

L2 informace o připojených stanicích

Pro detekování připojených koncových zařízení. Možnost detekce zablokovaných stanic atd.

Vytížení linek

Získávání informací o vytížení linek může odhalit neoptimální chování síťových prvků nebo stanice vytěžující síť.

Automatická detekce topologie

Pro získání topologie celé sítě se dá použít standardizovaný protokol LLDP nebo proprietární protokol CDP. Uživatel by nemusel ve webovém rozhraní vytvářet model mapy, ale měl by možnost si určit, které prvky chce vykreslit a všechna propojení by se vykreslila automaticky.

19 Závěr

Úkolem této práce bylo vytvořit programové vybavení, které bude umožňovat vytvoření mapy sítě ve webovém rozhraní. Vytvořenou mapu lze uložit do databáze a následně shromažďovat data o stavu STP protokolu na jednotlivých portech síťových prvků. Takto získaná data je možné vizualizovat ve webovém rozhraní.

Podařilo se mi vytvořit programové vybavení, které se připojuje k síťovým zařízením a získává z nich potřebné informace o stavu STP. Získané informace jsou zpracovány a uloženy. Webové rozhraní poskytuje editační mód, ve kterém si administrátor vytvoří mapu jakékoliv části univerzitní sítě. Ve vizualizačním módu si uživatel může zobrazit konkrétní VLAN společně s informacemi o stavu STP protokolu na jednotlivých portech.

Výsledná aplikace pomohla odhalit nesrovnalosti v konfiguraci síťových prvků. Například pro některé VLAN byl v síti přítomen více než jeden kořenový přepínač STP protokolu. Tento stav nastával v případě, kdy v konfiguraci trunku nebyla omylem povolena daná VLAN, přestože byla přítomna v databázi VLAN obou přepínačů. Důsledkem byla volba více kořenových přepínačů.

Zadavatel byl s výsledkem práce spokojen a hodlá ji začlenit do svých nástrojů pro správu a dohled sítě. V neposlední řadě lze tuto aplikaci využít i pro výukové účely. Student může vidět, jak funguje STP v reálné síti.

Svým rozsahem byla tato práce byla pro mne velkým přínosem. Seznámil jsem se s pro mě dosud neznámým jazykem Perl. Dále jsem poznal užitečné nástroje a technologie pro tvorbu webových aplikací. Rozšířil jsem si znalosti o jazyce HTML a JavaScript. Naučil jsem se pracovat s knihovnou *jQuery* a frameworkem *UIkit*.

Seznam použitých zkratek

AAA	Authentication, Authorization and Accounting
ACL	Access List
AES	Advanced Encryption Standard
AJAX	Asynchronous JavaScript and XML
ASN.1	Abstract Syntax Notation One
BPDU	Bridge Protocol Data Unit
CAM	Content-addressable memory
CDP	Cisco Discovery Protocol
CFI	Canonical Format Identifier
CIV	Centrum informatizace a výpočetní techniky
CPAN	Comprehensive Perl Archive Network
CSS	Cascading Style Sheets
DES	Data Encryption Standard
DOM	Document Object Model
DTP	Dynamic Trunking Protocol
ERA	Entity-Relationship-Attribute model
FTP	File Transfer Protocol
GUI	Graphical User Interface
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
HTTPS	HyperText Transfer Protocol Secure
IEEE	Institute of Electrical and Electronics Engineers
IOS	Internetwork Operating System
IP	Internet Protocol
ISL	Inter-Switch Link
ISO/OSI	International Organization for Standardization / Open Systems Interconnection model
JSON	JavaScript Object Notation
LACP	Link Aggregation Control Protocol
LAN	Local Area Network
LLDP	Link Layer Discovery Protocol
LPS	Laboratoř počítačových systémů
MAC	Medium Access Control
MAN	Metropolitan Area Network
MD5	Message Digest algorithm 5
MIB	Management Information Base
NvRAM	Non-volatile Random Access Memory
NX OS	Nexus Operating System
OID	Object Identifier
OVF	Open Virtualization Format
PAgP	Port Agregation Protocol
PDU	Protocol Data Unit
PHP	PHP: Hypertext Preprocessor
PRI	Priotiry value

Perl	Practical Extension and Report Language
PoE	Power over Ethernet
PVSTP	Per VLAN Spanning Tree Protocol
PVLAN	Private VLAN
RPVST+	Rapid Per VLAN Spanning Tree Protocol+
RSTP	Rapid Spanning Tree Protocol
SCP	Secure Copy Protocol
SHA	Secure Hash Algorithm
SMI	Structure of Management Information
SNMP	Simple Network Management Protocol
SQL	Structured Query Language
SSH	Secure Shell
STP	Spanning Tree Protocol
TCI	Tag Control Information
TCP	Transport Connection Protocol
TFTP	Trivial File Transfer Protocol
UDP	User Datagram Protocol
VID	VLAN ID
VTP	VLAN Trunking Protocol
VLAN	Virtual Local Area Network
XHTML	Extensible HyperText Markup Language

Seznam obrázků

Seznam tabulek

Seznam ukázek kódu

Použité zdroje

[1] Campus Network for High Availability Design Guide [online]. 21. 5. 2008, informační zdroj dostupný z URL: http://www.cisco.com/c/en/us/td/docs/solutions/Enterprise/Campus/HA_campus_DG/hacampusdg.html, (cit.1.5.2015).

[2] SNMP Tutorial Part 1 - The MIB, the manager, the agents... [online]. 21. 7. 2014, informační zdroj dostupný z URL: http://www.dpstele.com/snmp/tutorial-what-is.php, (cit.25.5.2015).

[3] Cisco IOS [online]. 14. 5. 2015, informační zdroj dostupný z URL: https://cs.wikipedia.org/wiki/Cisco_IOS, (cit.1.6.2015).

[4] How to make Ajax request using jQuery [online]. 1999-2015, informační zdroj dostupný z URL: http://1stwebmagazine.com/how-to-make-ajax-request-using-jquery, (cit.15.6.2015).

[5] DBI and MySQL [online]. 28. 10. 2002, informační zdroj dostupný z URL: http://1stwebmagazine.com/how-to-make-ajax-request-using-jquery, (cit.19.6.2015).

[6] CCNP SWITCH/Implementing VLANs in Campus Networks [online]. 3. 4. 2014, informační zdroj dostupný z URL: http://mars.tekkom.dk/mediawiki/index.php/CCNP_SWITCH/Implementing_VLANs_in_Campus_Networks, (cit.1.5.2015).

[7] End-to-End vs. Local VLAN Models [online]. 8. 2. 2013, informační zdroj dostupný z URL:https://www.nlogic.co/end-to-end-vs-local-vlan-models/, (cit.1.5.2015).

[8] Spanning Tree Introduction: Switching Loops | NLogic Training [online]. 12. 5. 2015, informační zdroj dostupný z URL: https://www.nlogic.co/spanning-tree-introduction-switching-loops/, (cit.22.5.2015).

[9] SC Labs | Networking notes (CCNA R/S, CCNA Sec, CCNP): Ex3 Chapter 5 - STP (+RSTP +Etherchannel) [online]. 20. 4. 2010, informační zdroj dostupný z URL: http://sclabs.blogspot.cz/2010/04/ex3-chapter-3-stp.html, (cit.24.5.2015).

[10] Spanning Tree Protocol Configuration :: Chapter 7. Spanning Tree Protocol (STP) :: Lan switching first-step :: Networking :: eTutorials.org [online]. 2015, informační zdroj dostupný z URL: http://etutorials.org/Networking/Lan+switching+first-step/Chapter+7.+Spanning+Tree+Protocol+STP/Spanning+Tree+Protocol+Configuration/, (cit.24.5.2015).

[11] CCNP SWITCH/Implementing Spanning Tree [online]. 5. 2. 2014, informační zdroj dostupný z URL: http://mars.tekkom.dk/mediawiki/index.php/CCNP_SWITCH/Implementing_Spanning_Tree, (cit.24.5.2015).

[12] Chapter 3 - Implementing Spanning Tree [online]. 10. 7. 2014, informační zdroj dostupný z URL: https://ls-a.org/doku.php?id=school:ccnp_1a_chapter3, (cit.24.5.2015).

[13] SNMP – začínáme [online]. 18. 5. 2014, informační zdroj dostupný z URL: http://m-linux.cz/2014/05/snmp-zaciname/, (cit.25.5.2015).

[14] Andrea, D.: Cisco IOU Web Interface [online]. 2015, informační zdroj dostupný z URL: http://www.routereflector.com/cisco/cisco-iou-web-interface/, (cit.15.6.2015).

[15] CASTRO, E.; HYSLOP, B.: *HTML5 a CSS3 Názorný průvodce tvorbou WWW stránek*. Computer Press, a.s., listopad 2012, ISBN 978-80-251-3733-8, 440 s.

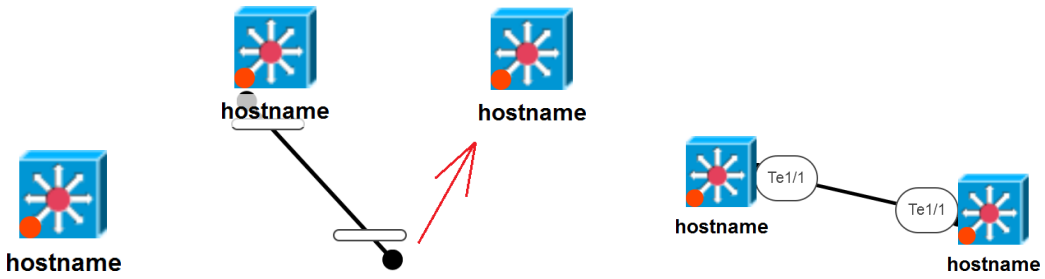
[16] CHAFFER, J.; SWEDBERG, K.: *Mistrovství v jQuery*. Brno: Computer Press, a.s., srpen 2013, ISBN 978-80-251-4103-8, 384 s.

[17] FROOM, R.; SIVASUBRAMANIAN, B.; FRAHIM, E.: *Implementing Cisco IP Switched Networks (SWITCH) Foundation Learning Guide: Foundation learning for SWITCH 642-813*. Foundation Learning Guides, Cisco Press, červen 2010, ISBN 978-1-58705-884-4, 560 s.

[18] KUROSE, J. F.; ROSS, K. W.: *Počítačové sítě*. Brno: Computer Press, a.s., první vydání, červenec 2014, ISBN 978-80-251-3825-0, 624 s.

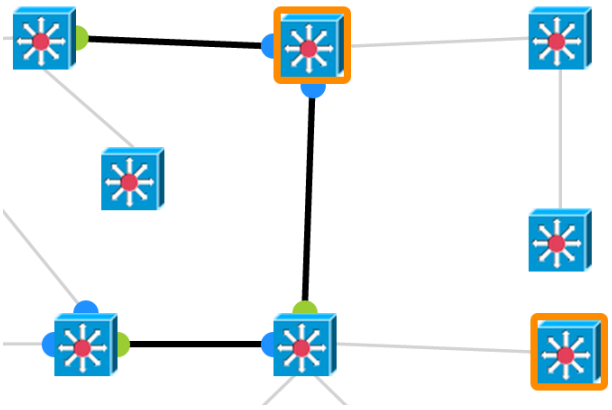
[19] LUBBERS, P.; ALBERS, B.; SALIM, F.: *HTML5 Programujeme moderní webové aplikace*. Brno: Computer Press, a.s., srpen 2011, ISBN 978-80-251-3539-6, 304 s.

A Přílohy

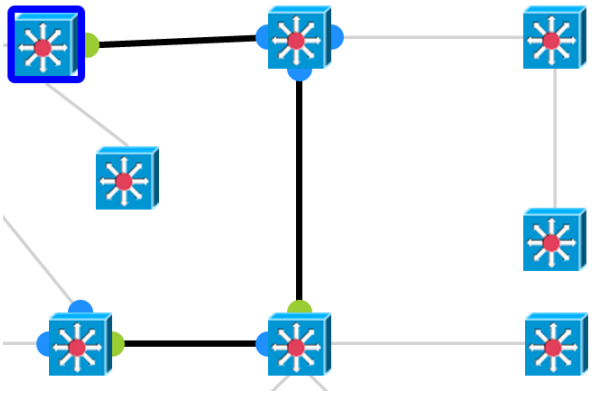


(a) (b) (c)

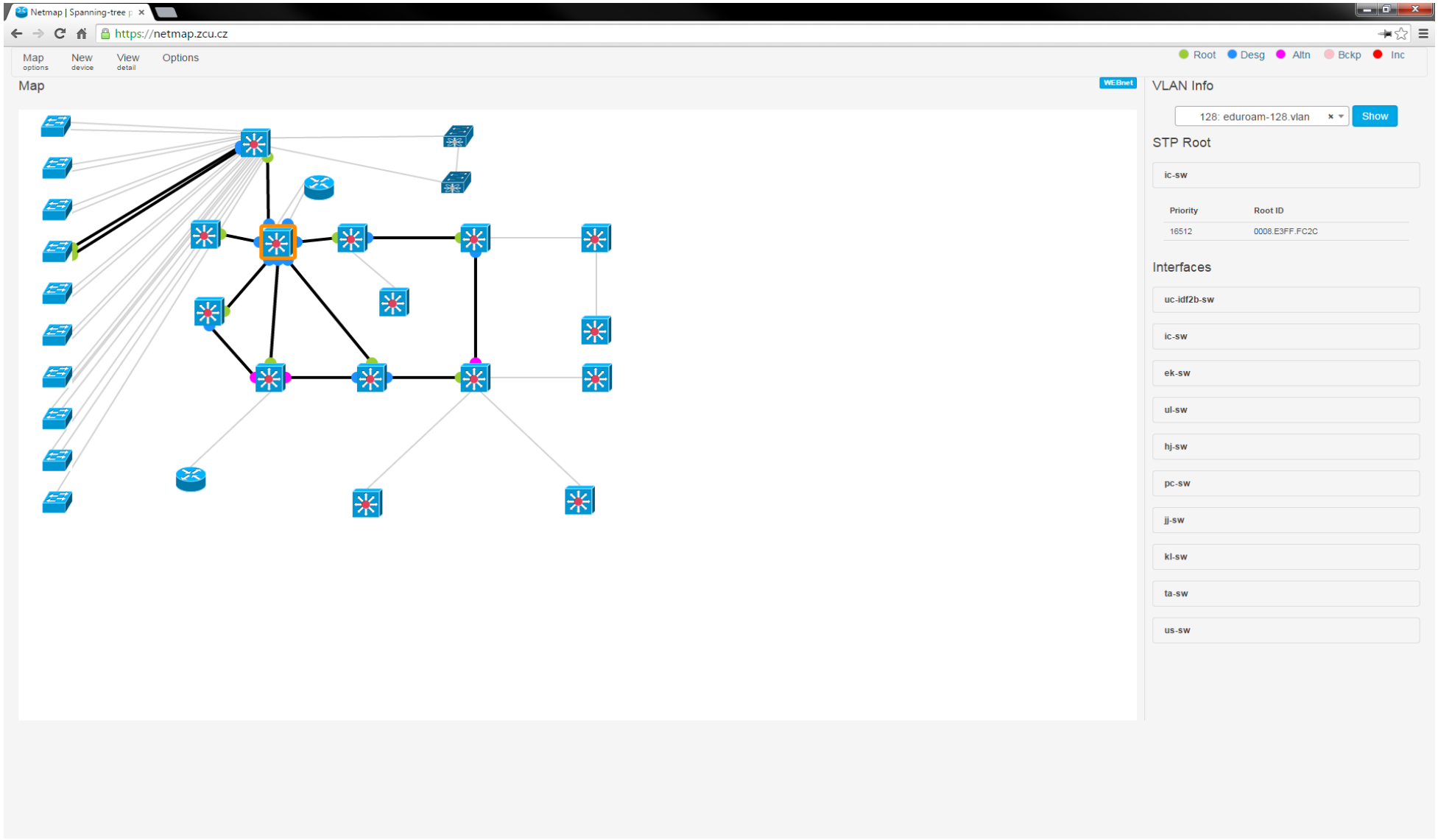
Obrázek A.1: Propojování elementů .



Obrázek A.2: Přítomnost dvou kořenových přepínačů.



Obrázek A.3: Nejbližší přepínač ke kořenovému přepínači.



Obrázek A.4: Ukázka webové aplikace.