VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

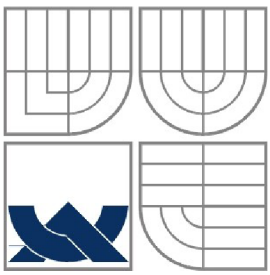
SYSTÉM PRO ZÍSKÁVÁNÍ PROVOZNÍCH ÚDAJŮ O POČÍTAČOVÉ SÍTI

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

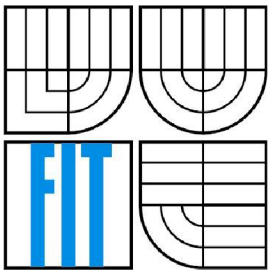
AUTOR PRÁCE
AUTHOR

BC. MILOŠ KUKLA

BRNO 2010



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

SYSTÉM PRO ZÍSKÁVÁNÍ PROVOZNÍCH ÚDAJŮ O POČÍTAČOVÉ SÍTI

SYSTEM FOR MANAGEMENT DATA ACQUISITION AT CAMPUS NETWORK

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

BC. MILOŠ KUKLA

VEDOUCÍ PRÁCE
SUPERVISOR

ING. PETR MATOUŠEK, PH.D.

BRNO 2010

Abstrakt

Práce se zabývá sběrem dat z aktivních prvků počítačové sítě a ukládáním a zobrazováním získaných údajů. Popisuje protokol SNMP, analyzuje a porovnává dostupné nástroje pro monitorování síťových zařízení. Popisuje návrh vlastního systému monitorování počítačové sítě a jeho nasazení v prostředí počítačové sítě VUT v Brně.

Abstract

This work deals with data capturing from the active network elements and storing and displaying the captured data. It describes the SNMP, analyzes and compares available tools for monitoring computer networks. It describes the design of own system for monitoring computer networks and deployment in computer network in the BUT campus.

Klíčová slova

počítačové sítě, LAN, monitorování počítačových sítí, nástroje pro monitorování, statistiky sítě, SNMP

Keywords

computer networks, LAN, monitoring of computer networks, tools for monitoring, network statistics, SNMP

Citace

Miloš Kukla: Systém pro získávání provozních údajů o počítačové síti, diplomová práce, Brno, FIT VUT v Brně, 2010

Systém pro získávání provozních údajů o počítačové síti.

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením Ing. Petra Matouška, Ph.D.
Další informace mi poskytli Ing. Tomáš Podermaňski a Ing. Matěj Grégr.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Miloš Kukla
10. května 2010

Poděkování

Touto cestou bych rád poděkoval Ing. Petrovi Matouškovi, Ph.D. za vedení, připomínky a konzultace k této práci. Také bych rád poděkoval Ing. Tomášovi Podermaňskému a Ing. Matějovi Grégrovi za konzultace a poskytnuté rady.

Obsah

1 Úvod

V posledních dvaceti letech nastal v oblasti počítačových sítí rychlý rozvoj. Těžko dnes budeme hledat firmu, která by neměla vlastní webovou prezentaci a možnost připojení k internetu. Vysoké pronikání internetu nastalo také u domácností, navíc osobní počítač už není jediné zařízení, které nám umožňuje připojení k počítačové síti. Stále více mobilních telefonů, multimediálních přehrávačů a další elektroniky využívá služeb internetu. V současné době je zařízení schopných připojení k počítačové síti tolik, že 32-bitový adresový prostor, definovaný internetovým protokolem IPv4 v roce 1981 [1], je téměř vyčerpaný a probíhá přechod na IPv6 se 128-bitovou adresou.

Rozvoj nastal také v oblasti internetových aplikací. Vedle služeb jako jsou web, mail nebo ftp tvoří v současné době velkou část přenesených dat další služby, např. streaming a peer-to-peer přenos dat nebo internetová telefonie. Podle měření na zařízeních páteřní sítě Sprint IP v roce 2003 dosahoval přenos dat typu streaming 26% procent celkového přenosu a data typu peer-to-peer přestavovala v určitých chvílích až 80% celkového vytížení [2]. Streaming a telefonování přes internet kladou na síť další požadavek – garanci kvality služby.

Takové zvyšování potřeb na přenos počítačovou sítí s sebou nese také zvyšování nároků na její administraci a monitorování. V některých případech může výpadek sítě způsobit finanční ztráty nebo dokonce ohrozit lidské životy. U rozsáhlejších sítí se statická konfigurace stala neúnosnou a začaly se používat techniky (např. dynamické směrovací protokoly), které umožňují síti flexibilně reagovat na změny. Přesto nastávají situace, které vyžadují zásah člověka spravujícího danou síť. Aby administrátor mohl efektivně vyřešit problém, potřebuje znát stav sítě v době výpadku a také v době, která výpadku předcházela.

Tato práce se zaměřuje na možnosti monitorování stavu sítě, zpracování a uložení získaných dat a jejich následnou reprezentaci. Cílem je přinést člověku spravujícímu počítačovou síť užitečné informace vedoucí k odhalení příčin problému a jeho odstranění. Dalším cílem je využití získaných dat jako vstupních hodnot při stavbě modelu počítačové sítě.

První část této práce popisuje protokol SNMP, který umožňuje monitorovat a částečně také konfigurovat zařízení zapojená v počítačové síti. Zabývá se především těmi oblastmi SNMP, které je možné využít pro získání provozních údajů z aktivních síťových prvků.

Druhá část práce přináší přehled dostupných nástrojů, určených pro monitorování síťových zařízení. Popisuje jejich vlastnosti a možnost jejich nasazení na počítačové síti VUT v Brně.

Ve třetí části je řešen návrh vlastního systému pro monitorování počítačové sítě. Je navržena jeho struktura, stanoveny teoretické základy a popsána jejich praktická realizace. V závěru této části jsou zhodnoceny výsledky testovacího provozu navrženého systému.

2 SNMP

SNMP je internetový standard pro správu zařízení v IP sítích. Je podporován v různých typech zařízení jako jsou servery, směrovače, přepínače, záložní zdroje napájení (UPS), tiskárny, pracovní stanice a další. Možnosti využití tohoto protokolu jsou rozmanité – od monitorování stavu zařízení přes zasílání zpráv monitorovacímu systému až po vzdálenou konfiguraci aktivních síťových prvků. Podklady pro tuto kapitolu čerpají ze zdrojů [3], [4], [5], [6], [7].

2.1 Architektura

Systém pro správu sítě založený na SNMP se skládá ze tří částí, nazvaných *manager*, *agent* a *database*. Vztah mezi těmito entitami je definován jako vztah klient-server, kde *manager* je klientský program, zatímco *agent* běžící na vzdáleném zařízení může být považován za server.

2.1.1 Manager

Manager, častěji nazývaný jako Network Management Station (NMS), je program spuštěný na jednom nebo více počítačích v síti a provádí tyto činnosti:

- dotazování agentů (tzv. *polling*)
- zachycení zprávy (tzv. *trap*)

Polling spočívá v zaslání dotazu agentovi (serveru, směrovači, přepínači, atd.) a obdržení odpovědi, která může být uložena a později využita při řešení problému nebo pro zobrazení statistik. Dotaz i odpověď je přenášena protokolem UDP přes port 161. Protože UDP nezaručuje doručení dat, musí toto řešit NMS např. opakováním dotazu při nedoručení odpovědi v definovaném času.

Trap je prostředek agenta, kterým může informovat NMS o nějaké události. Tato informace je zasílána asynchronně, aniž by se na ni NMS dotazoval. Ten může na základě zachycené zprávy provést předem definovanou akci, např. informovat správce sítě o problému. Tento typ zprávy je také zasílán UDP protokolem, NMS naslouchá na portu 162.

2.1.2 Agent

Agent je program běžící na síťovém zařízení, který je schopný odpovídat na SNMP dotazy. Může to být samostatný program (např. unixový démon) nebo součást operačního systému (např. IOS na Cisco směrovači). Je možné pracovat i se zařízeními, která mají místo protokolu SNMP implementovaný jiný druh protokolu pro správu. V takovém případě je nutné použít zařízení, které bude sloužit jako tzv. *proxy agent* překládající SNMP dotazy na protokol, který zařízení podporuje. Síťové zařízení může informovat NMS o událostech zasíláním asynchronních zpráv typu *trap*. Některá zařízení mohou zaslat *trap* zprávu „*clear all*“, která informuje NMS o návratu zařízení do funkčního stavu.

2.1.3 Database

Database je častěji uváděná jako Management Information Base (MIB). Tato MIB definuje, které informace můžeme získat ze spravovaného zařízení a jaké parametry je možné nastavit z NMS. Každé zařízení může obsahovat různou konfiguraci či poskytovat jiný souhrn statistických informací. Mezi tyto údaje patří např. stav portů přepínače, obsah směrovací tabulky směrovače, přiřazení IP adres k rozhraním, počet přenesených bajtů na jednotlivých portech a další. Množina těchto hodnot se označuje jako zařízením spravovaná MIB. Každý jednotlivý údaj je pak nazýván spravovaným

objektem (*managed object*) a skládá se z názvu, jednoho nebo více atributů, a množiny operací, které je možné nad objektem provádět. MIB bude do hloubky popsána v kapitole 2.3.

2.2 Verze SNMP

V současné době se můžeme setkat se třemi verzemi protokolu.

2.2.1 SNMP verze 1

Pro vytvoření spojení mezi monitorovacím systémem a agentem definuje SNMPv1 tři způsoby komunikace: *read-only*, *read-write* a *trap*. Komunikaci je možné zabezpečit heslem, pro které se používá termín *community*. Komunikace *read-only* umožňuje číst údaje od agenta, nepovoluje však nastavování parametrů agenta. Ve výchozím stavu je heslo nejčastěji nastaveno na hodnotu *public*. Komunikace *read-write* umožňuje jak čtení tak i zápis údajů, výchozí hodnota hesla bývá nastavena na *private*. Komunikace *trap* slouží pro zasílání asynchronních zpráv od agenta monitorovacímu systému. Problém hesel je v tom, že jsou sítí posílány jako nešifrovaný text, což představuje bezpečnostní riziko, především u komunikace *read-write* a *trap*.

Pro čtení a zápis údajů definuje tato verze pět příkazů: *GetRequest*, *GetNextRequest*, *SetRequest*, *GetResponse*, *Trap*. Příkazy budou popsány v kapitole 2.4.

Standard SNMPv1 definuje dokument RFC 1157 [8].

2.2.2 SNMP verze 2c

Stejně jako SNMPv1 i tato verze využívá tři způsoby komunikace a nešifrovaný přenos. Oproti verzi 1 se změnil rozsah datového typu *Counter* z 32 bitů na 64 bitů, tím se snížilo riziko zkreslených statistik u rychlých linek (např. pokud počet přenesených bajtů na síťovém rozhraní dosáhl maxima 32-bitového rozsahu verze 1, tak došlo k vynulování čítače a předávání nesprávné hodnoty do statistik).

Nově tato verze umožňuje distribuovanou správu sítě, kdy jedna NMS může zasílat zprávy jiné NMS (tato schopnost se označuje jako *manager-to-manager*). Za tímto účelem byly do MIB přidány dvě nové skupiny – *Alarm* a *Event*. Skupina *Alarm* umožňuje definovat úrovně, jejichž překročení vede k vyvolání poplašné zprávy. Druhá skupina, *Event*, určuje, kdy se má zaslat zpráva typu trap na základě jedné nebo více hodnot určitého prvku. K původním pěti příkazům SNMPv1 byly přidány dva nové: *GetBulkRequest*, *InformRequest*.

Standard SNMPv2 definují dokumenty RFC 1901, RFC 1908, RFC 3416, RFC 3417 [8].

2.2.3 SNMP verze 3

SNMPv3 řeší bezpečnostní problém předchozích verzí zavedením zabezpečené komunikace a autentizace. Bohužel tato verze není v řadě zařízení podporována.

Pro bezpečný přenos zpráv definuje tato verze bezpečnostní model založený na uživatelích (USM – User-based Security Model), pro kontrolu přístupu je definován model založený na pohledech (VACM – View-based Access Control Model). Údaje o uživatelských účtech se nacházejí v MIB ve skupině *usmUser*, pro kontrolu přístupu slouží skupiny *vacmContextTable*, *vacmSecurityToGroupTable*, *vacmAccessTable* a *vacmMIBViews*. Jako kryptografické algoritmy jsou specifikovány DES, MD5, SHA-1, HMAC.

Standard SNMPv3 definují dokumenty RFC 3410 až 3417, RFC 3584, RFC 3826 [8].

2.3 MIB

MIB je textový soubor obsahující seznam proměnných SNMP a s nimi souvisejících údajů. Soubor se zapisuje v syntaxi ASN.1 [9], což je standardní formát pro popis dat (čitelný jak člověkem tak programy). Samotné datové typy jsou standardizovány standardem SMI, definovaným v dokumentech RFC 2578 až 2580 [8].

Úkolem MIB je:

- přiřadit každé proměnné textový název
- definovat datový typ proměnné
- popsat funkci proměnné a popsat, jak daný datový typ danou funkci zajišťuje
- definovat přístupové možnosti k proměnné (zda je jen pro čtení nebo i pro zápis)

První MIB (tzv. Internet Standard MIB) vydala organizace IETF v roce 1998 jako dokument RFC 1066 [8]. Ta byla později nahrazena novější verzí označovanou jako MIB-II (RFC 1213 [8]) a jakékoliv zařízení, které dnes používá SNMP, musí mít implementováno tuto novější verzi.

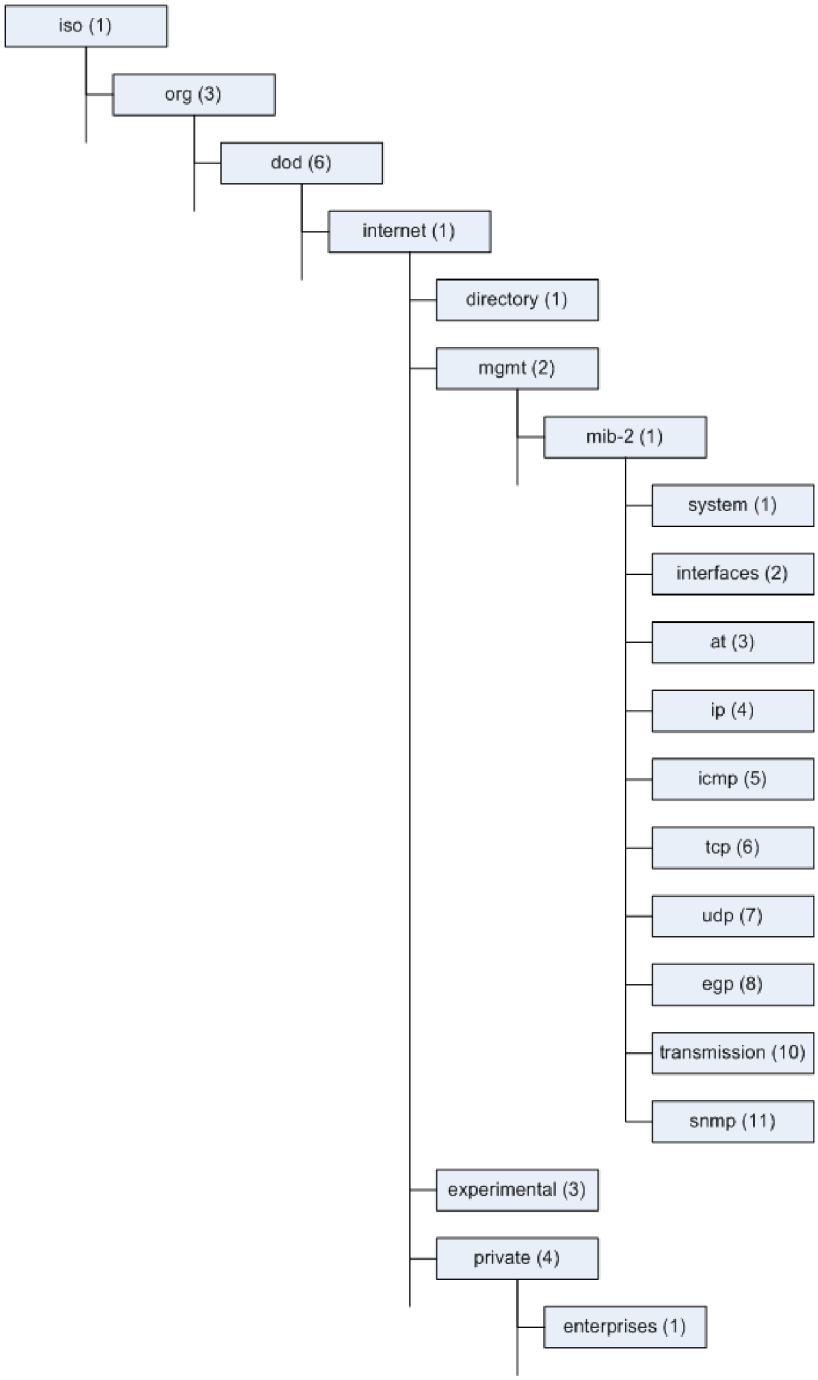
Protože je vhodné, aby jednotlivé instituce mohli spravovat vlastní skupiny proměnných, tak se všechny SNMP proměnné nenacházejí v jediném souboru, ale jsou rozděleny do množství samostatných částí. Některé části vydává IETF, další vydávají jednotliví výrobci.

Všechny spravované objekty jsou uspořádané v hierarchické či stromové struktuře. Listy stromu reprezentují aktuálně spravované objekty. Každý takový objekt má přiřazený unikátní identifikátor nazývaný *object identifier* (často se zkracuje na *OID*). Tento identifikátor se skládá z řady čísel oddělených tečkami a určuje umístění objektu ve stromové struktuře. Tato čísla se používají při přenosu zpráv prostřednictvím SNMP, textové názvy jsou určené pouze pro zvýšení čitelnosti. Hierarchický strom je zobrazen na obrázku 2.1. Kořen celé hierarchie (nemá název, značí se tečkou) má tři potomky. Jeden je určen pro ISO, druhý pro CCITT (dnes ITU), třetí je pro společné použití ISO a CCITT. Jeden z podstromů *iso* je určen pro použití dalšími organizacemi (*org*), z nichž jedna je U.S. Department of Defense (*dod*). Jedním z podstromů *dod* je *internet*, do kterého spadají všechny proměnné, které SNMP používá. OID této větve je *.1.3.6.1* (textově *.iso.org.dod.internet*), což je hodnota prefixu pro všechny SNMP proměnné.

Pod uzlem internet jsou definovány čtyři uzly:

- *directory* je rezervován pro budoucí použití s OSI Directory.
- *mgmt* obsahuje definice MIB, které byly schváleny od IAB. V současnosti je aktuální verzí MIB-II (uzel *mib-2*), která je rozdělená do několika skupin, popsaných v tabulce 2.1. Protože právě do této větve spadá správa síťových zařízení, budou její podskupiny užitečné pro správu dále popsány podrobněji.
- *experimental* slouží pro identifikaci objektů při experimentech.
- *private* se používá pro identifikaci jednostranně schválených objektů. Slouží výrobcům pro rozšíření možností správy jejich výrobků a ke sdílení těchto objektů s ostatními výrobci a uživateli.

V následujících kapitolách budou popisovány skupiny uzlu *mib-2*, které obsahují informace významné pro vytvářený monitorovací systém, jehož návrhem se zabývá kapitola 4.



Obrázek 2.1: Hierarchie identifikátorů objektů [6]

2.3.1 Skupina *system*

Tento přímý potomek uzlu *mib-2* obsahuje proměnné poskytující obecné informace o zařízení. Ve skupině je definováno sedm proměnných, jejich seznam je uveden v tabulce 2.2. Sloupec „Přístup“ udává, zda objekt umožňuje zápis hodnoty (RW) nebo pouze její čtení (RO), případně že je hodnota nepřístupná (NA).

Proměnná *sysObjectId* obsahuje unikátní identifikátor objektu nastavený výrobcem. Tento identifikátor vždy začíná řetězcem .1.3.6.1.4.1, což je prefix výrobcem definované části stromu.

Skupina	Popis
system	Souhrnné informace o systému.
interfaces	Informace o každém síťovém rozhraní na daném systému.
at	Tabulka pro překlad IP adres na MAC adresy, nyní je nahrazena obecnější tabulkou *ip.ipNetToMediaTable*.
ip	Informace vztahující se k implementaci protokolu IP na daném systému.
icmp	Informace vztahující se k implementaci protokolu ICMP na daném systému.
tcp	Informace vztahující se k implementaci protokolu TCP na daném systému.
udp	Informace vztahující se k implementaci protokolu UDP na daném systému.
egp	Informace vztahující se k implementaci protokolu EGP na daném systému.
dot3	Informace o datových přenosech a protokolech na každém rozhraní systému.
snmp	Informace vztahující se k implementaci protokolu SNMP na daném systému.

Tabulka 2.1: Skupiny objektů MIB-II [6]

Proměnná *sysUpTime* udává čas od poslední inicializace systému zařízení. Může být použita např. pro zjištění, o kolik se změnila hodnota čítačů za určitý časový úsek, nebo pro detekci výpadku zařízení.

Proměnná *sysServices* udává, které síťové vrstvy zařízení podporuje. Hodnota je interpretovaná jako sedmibitový kód, kde každý bit koresponduje s vrstvou TCP/IP nebo OSI architektury (nejméně významný bit odpovídá první vrstvě). Pokud zařízení poskytuje službu na určité vrstvě, potom má odpovídající bit hodnotu „1“. Výsledná hodnota proměnné je vypočtena jako

$$sysServices = \sum_{L \in S} 2^{L-1} \tag{1}$$

kde S je množina čísel podporovaných síťových vrstev. Např. zařízení, které poskytuje aplikační služby (např. emailový server) bude mít hodnotu 72 ($2^{(4-1)} + 2^{(7-1)}$), binárně 1001000.

Proměnná	Přístup	Popis
sysDescr	RO	Textový popis zařízení (hardware, operační systém, atd.).
sysObjectID	RO	Identifikátor SNMP objektu zařízení.
sysUpTime	RO	Doba od poslední inicializace systému v setinách sekundy.
sysContact	RW	Kontaktní údaje na správce tohoto zařízení.
sysName	RW	Správcem přiřazený název zařízení.
sysLocation	RW	Informace o fyzickém umístění zařízení.
sysServices	RO	Hodnota udávající síťové vrstvy, na kterých zařízení pracuje.

Tabulka 2.2: Proměnné skupiny system [5]

2.3.2 Skupina *interfaces*

Obsahuje informace o všech rozhraních daného zařízení, proměnná *ifNumber* udává jejich počet (včetně zařízení typu *loopback*). Samotné informace se nachází v tabulce *ifTable*, jejíž sloupce jsou

popsány v tabulce 2.3. Hodnoty z tabulky *ifTable* je možné využít pro sběr přenosových statistik jednotlivých rozhraní.

Proměnná	Přístup	Popis
ifIndex	RO	Unikátní hodnota každého rozhraní.
ifDescr	RO	Informace o rozhraní (výrobce, název, verze hardwaru, atd.).
ifType	RO	Celočíselná hodnota reprezentující protokol rozhraní.
ifMtu	RO	Nejvyšší možná hodnota MTU se kterou může zařízení pracovat.
ifSpeed	RO	Aktuální přenosová rychlost rozhraní.
ifPhysAddress	RO	Fyzická adresa druhé síťové vrstvy.
ifAdminStatus	RW	Administrativní stav rozhraní (up(1), down(2), testing(3)).
ifOperStatus	RO	Operační stav rozhraní (up(1), down(2), testing(3)).
ifLastChange	RO	Hodnota *sysUpTime* od poslední změny *ifOperStatus*.
ifInOctets	RO	Celkový počet bajtů přijatých rozhraním.
ifInUcastPkts	RO	Počet přijatých unicast paketů.
ifInNUcastPkts	RO	Počet přijatých paketů jiných než unicast.
ifInDiscards	RO	Počet zahozených příchozích paketů, přestože nebyly chybné.
ifInErrors	RO	Počet zahozených příchozích paketů, protože byly chybné.
ifInUnknownProtos	RO	Počet příchozích paketů neznámého nebo nepodporovaného protokolu.
ifOutOctets	RO	Celkový počet bajtů odeslaných rozhraním.
ifOutUcastPkts	RO	Počet odeslaných unicast paketů.
ifOutNUcastPkts	RO	Počet odeslaných paketů jiných než unicast.
ifOutDiscards	RO	Počet zahozených odchozích paketů, přestože nebyly chybné.
ifOutErrors	RO	Počet zahozených odchozích paketů, protože byly chybné.
ifOutQLen	RO	Celkový počet paketů v odchozí frontě.
ifSpecific	RO	Identifikátor objektu v závislosti na médiu.

Tabulka 2.3: Proměnné tabulky ifTable ve skupině interfaces [5], [6]

2.3.3 Skupina *ip*

Tato skupina obsahuje velké množství údajů vztahujících se k IP vrstvě, dále budou popsány pouze uzly *ipAddressTable*, *ipAddrTable*, *dot1qTpFdbTable*, *inetCidrRouteTable*, *ipNetToPhysicalTable*, *ipNetToMediaTable*, protože obsahují informace cenné pro monitorování aktuální konfigurace síťového prvku.

Tabulka *ipAddressTable* udržuje informace o IP adresách přiřazených k síťovým rozhraním. Některé její proměnné jsou určené také pro zápis, je tedy možné touto cestou zařízení také konfigurovat. Popis proměnných se nachází v tabulce 2.4. Obsahuje jak IPv4 tak IPv6 adresy a nahrazuje tabulku *ipAddrTable*, která umožňuje pracovat pouze s IPv4 adresami. Není však podporovaná na všech zařízeních, v takovém případě je nutné získávat tyto informace ze zmiňované *ipAddrTable*, jejíž proměnné popisuje tabulka 2.5.

Proměnná	Přístup	Popis
ipAddressAddrType	NA	Typ adresy proměnné *ipAddressAddr*.
ipAddressAddr	NA	Hodnota adresy.
ipAddressIfIndex	RW	Index rozhraní, ke kterému adresa náleží.
ipAddressType	RW	Typ adresy (unicast(1), anycast(2), broadcast(3) – pouze IPv4).
ipAddressPrefix	RO	Ukazatel do tabulky prefixů.
ipAddressOrigin	RO	Vznik adresy (other(1), manual(2), dhcp(4), linklayer(5), random(6)).
ipAddressStatus	RW	Stav adresy, vyjadřuje zda adresa může být použita pro komunikaci (preferred(1), deprecated(2), invalid(3), inaccessible(4), unknown(5), tentative(6), duplicate(7), optimistic(8)). Není-li dáno jinak, je IPv4 adresa vždy preffered(1).
ipAddressCreated	RO	Hodnota *sysUpTime* v čase, kdy byl tento záznam vytvořen, nebo 0, pokud existoval před poslední inicializací systému.
ipAddressLastChanged	RO	Hodnota *sysUpTime* v čase, kdy byl záznam naposledy změněn, nebo 0, pokud byl změněn před poslední inicializací systému.
ipAddressRowStatus	RW	Při tvorbě nebo mazání řádků v tabulce udává, v jakém stavu jsou údaje tohoto řádku tabulky ve vztahu k zařízení (active(1), notInService(2), notReady(3), createAndGo(4), createAndWait(5), destroy(6)).
ipAddressStorageType	RW	Způsob uložení tohoto řádku v paměti (other(1), volatile(2), nonVolatile(3), pernament(4), readOnly(5)).

Tabulka 2.4: Proměnné uzlu ipAddressTable [7]

Proměnná	Přístup	Popis
ipAdEntAddr	RO	IP adresa, ke které tento záznam náleží
ipAdEntIfIndex	RO	Unikátní index určující rozhraní, ke kterému záznam přísluší
ipAdEntNetMask	RO	Síťová maska IP adresy
ipAdEntBcastAddr	RO	Hodnota nejméně významného bitu v broadcoastové adrese
ipAdEntReasmMaxSize	RO	Největší velikost IP datagramu, který může zařízení znovu sestavit z příchozích datagramů na tomto rozhraní

Tabulka 2.5: Proměnné uzlu ipAddrTable [6]

Tabulka *dot1qTpFdbTable* ukládá informace o naučených MAC adresách a jejich přiřazení k portům zařízení a příslušnost k VLAN. Nahrazuje tabulku *dot1dTpFdbTable*, která neobsahovala informace o VLAN. Popis proměnných se nachází v tabulce 2.6.

Proměnná	Přístup	Popis
dot1qTpFdbAddress	NA	Unicast MAC adresa, o které má zařízení informaci.
dot1qTpFdbPort	RO	Hodnota portu, ze kterého byla adresa naučena, nebo 0, pokud adresa nebyla naučena ale zařízení o ní má informaci.
dot1qTpFdbStatus	RO	Status záznamu (other(1), invalid(2), learned(3), self(4), mgmt(5)).

Tabulka 2.6: Proměnné uzlu dot1qTpFdbTable [7]

Tabulka *inetCidrRouteTable* je potomkem uzlu *ipForward* a nahrazuje tabulky *ipRouteTable* a *ipForwardTable*. Obsahuje informace o směrování nezávisle na verzi IP. Popis proměnných se nachází v tabulce 2.7.

Proměnná	Přístup	Popis
inetCidrRouteDestType	NA	Typ cílové IP adresy.
inetCidrRouteDest	NA	Hodnota cílové IP adresy.
inetCidrRoutePfxLen	NA	Maska pro logický AND s cílovou adresou před jejím porovnáním s
inetCidrRoutePolicy	NA	Index pro rozlišení vícenásobných hodnot vztahujících se ke stejné destinaci.
inetCidrRouteNextHopType	NA	Typ adresy příštího skoku.
inetCidrRouteNextHop	NA	Pro vzdálené cíle adresa příštího uzlu, jinak řetězec nulové délky.
inetCidrRouteIfIndex	RW	Index rozhraní pro příští skok nebo 0.
inetCidrRouteType	RW	Typ směrování (other(1), reject(2), local(3), remote(4), blackhole(5)).
inetCidrRouteProto	RO	Způsob, jakým byla tato cesta naučena.
inetCidrRouteAge	RO	Počet sekund od poslední aktualizace nebo ověření cesty.
inetCidrRouteNextHopAS	RW	Číslo autonomního systému příštího skoku nebo 0.
inetCidrRouteMetric1	RW	Primární metrika této cesty nebo -1.
inetCidrRouteMetric2	RW	Náhradní metrika této cesty nebo -1.
inetCidrRouteMetric3	RW	Náhradní metrika této cesty nebo -1.
inetCidrRouteMetric4	RW	Náhradní metrika této cesty nebo -1.
inetCidrRouteMetric5	RW	Náhradní metrika této cesty nebo -1.
inetCidrRouteStatus	RW	Při tvorbě nebo mazání řádků v tabulce udává, v jakém stavu jsou údaje tohoto řádku tabulky ve vztahu k zařízení (active(1), notInService(2), notReady(3), createAndGo(4), createAndWait(5), destroy(6)).

Tabulka 2.7: Proměnné uzlu inetCidrRouteTable [7]

Tabulka *ipNetToMediaTable* slouží k překladu IP adres na hardwarové adresy. V případě Ethernetu to znamená nalezení příslušné ethernetové adresy v tabulce protokolu ARP. Popis proměnných se nachází v tabulce 2.8. V současné době je nahrazena tabulkou *ipNetToPhysicalTable*, která obsahuje jak IPv4 tak IPv6 adresy, není však podporována na všech zařízeních. Popis proměnných se nachází v tabulce 2.9.

Proměnná	Přístup	Popis
ipNetToMediaIfIndex	RW	Index rozhraní s danou adresou.
ipNetToMediaPhysAddress	RW	Fyzická adresa.
ipNetToMediaNetAddress	RW	IP adresa příslušející k fyzické adrese.
ipNetToMediaType	RW	Způsob mapování (other(1), invalid(2), dynamic(3), static(4)).

Tabulka 2.8: Proměnné uzlu ipNetToMediaTable [6]

Proměnná	Přístup	Popis
ipNetToPhysicalIfIndex	NA	Index rozhraní s danou adresou.
ipNetToPhysicalNetAddressType	NA	Typ IP adresy (unknown(0), ipv4(1), ipv6(2), ipv4z(3), ipv6z(4), dns(16)).
ipNetToPhysicalNetAddress	NA	IP adresa příslušející k fyzické adrese.
ipNetToPhysicalPhysAddress	RW	Fyzická adresa.
ipNetToPhysicalLastUpdated	RO	Hodnota *sysUpTime* v čase, kdy byl záznam naposledy změněn, nebo 0, pokud byl změněn před poslední inicializací systému.
ipNetToPhysicalType	RW	Způsob mapování (other(1), invalid(2), dynamic(3), static(4), local(5)).
ipNetToPhysicalState	RO	Stav dostupnosti souseda na rozhraní, ke kterému se mapování vztahuje (reachable(1), stale(2), delay(3), probe(4), invalid(5), unknown(6), incomplete(7)). Pokud se detekce dostupnosti nepoužívá (např. u IPv4), je stav vždy unknown(6).
ipNetToPhysicalRowStatus	RW	Při tvorbě nebo mazání řádků v tabulce udává, v jakém stavu jsou údaje tohoto řádku tabulky ve vztahu k zařízení (active(1), notInService(2), notReady(3), createAndGo(4), createAndWait(5), destroy(6)).

Tabulka 2.9: Proměnné uzlu ipNetToPhysicalTable [7]

2.4 Příkazy SNMP

Každý SNMP paket přenáší kromě samotných administrativních informací ještě další tři údaje. Prvním je číslo verze (pro SNMPv1 je to nula), druhým údajem je hodnota *community* a třetí údaj určuje typ zprávy, kterou si agent vyměňuje s monitorovacím systémem – tento údaj se označuje jako Protocol Data Unit (PDU). Pro každý z níže popsaných příkazů je definován standardní formát PDU.

- **GetRequest** je příkaz zaslaný NMS agentovi. Můžeme ho použít k přečtení jedné proměnné nebo seznamu proměnných z MIB agenta. Tento příkaz, stejně jako ostatní příkazy SNMP, vyžaduje jako parametr adresu dotazovaného systému. Protože SNMP příkazy jsou přenášeny prostřednictvím IP protokolu, je parametrem IP adresa cílového systému.
- **GetNextRequest** je podobný předchozímu příkazu s jedním rozdílem – příkaz GetRequest vrátil hodnotu proměnné, na kterou se dotazujeme, tento příkaz vrátí hodnotu proměnné, která následuje za dotazovanou proměnnou. Zadáváním sekvence příkazů GetNextRequest tak můžeme procházet celou MIB agenta.
- **SetRequest** je stejně jako předchozí dva příkazy zasílán od NMS agentovi. Jedná se o žádost agentovi o nastavení některého ze spravovaných objektů na požadovanou hodnotu.
- **GetResponse** je příkaz zaslaný agentem pro NMS. Jedná se o mechanismus, kterým může agent odpovídat na příkazy GetRequest, GetNextRequest a SetRequest. Informace vrácené tímto příkazem umožňují určit, zda byl obdržený příkaz úspěšně zpracován a pokud ano, vrací se seznam spravovaných objektů, které byly zpracovány společně s jejich aktuálními hodnotami.
- **Trap** umožňuje agentovi aktivně informovat NMS o vzniku nějaké předem definované události, např. o změně stavu ethernetového portu.
- **GetBulkRequest** (SNMPv2c, SNMPv3) umožňuje získat od agenta naráz více MIB objektů. Klasický GetRequest příkaz se může také pokusit získat více než jeden objekt, velikost zprávy je však omezena možnostmi agenta, pokud agent nedokáže zaslat všechny odpovědi, vrátí pouze chybovou zprávu a žádná data. Oproti tomu příkaz GetBulkRequest říká agentovi, ať pošle tolik odpovědí, kolik dokáže. To znamená, že i nekompletní odpověď bude doručena. Příkaz vyžaduje kromě seznamu dotazovaných objektů zadat další dva parametry – *nonrepeaters* a *max-repetitions*. *Nonrepeaters* říká, že prvních *N* objektů má být získáno jednorázovým voláním příkazu GetNextRequest. *Max-repetitions* říká, že pro každý zbývající ze zadaných objektů se má zavolat *M* příkazů GetNextRequest – jedná se tedy dotaz na vícerozměrné veličiny (např. do tabulky s informacemi o portech přepínače). Celkový počet získaných odpovědí je pak dán rovnicí $N + (M * R)$, kde R je počet vícerozměrných veličin.
- **InformRequest** (SNMPv2c, SNMPv3) slouží pro vzájemnou komunikaci mezi NMS. Cílový systém po obdržení této zprávy zasílá odesilateli odpověď s potvrzením o přijetí. Takto lze vytvořit víceúrovňovou hierarchii NMS, kde každý segment sítě je spravován lokálním NMS a tyto lokální systémy předávají získané informace jednomu globálnímu NMS, který udržuje informace o celé síti.

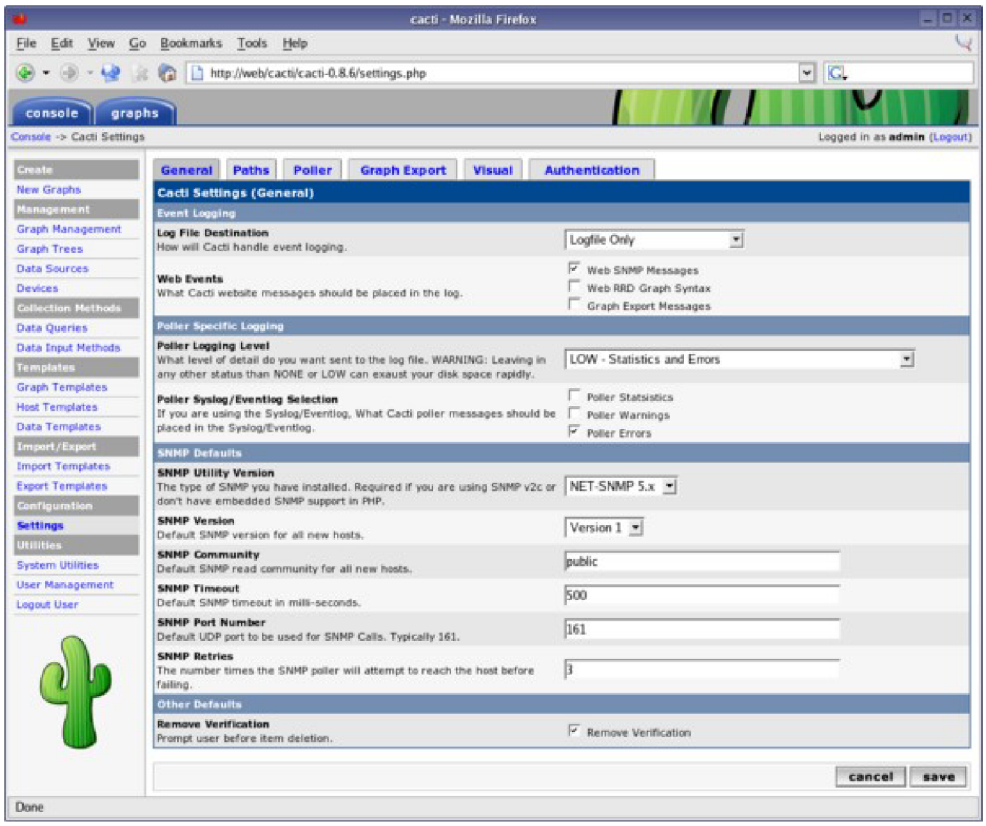
3 Dostupné nástroje NMS

Tato kapitola se zabývá přehledem dostupných nástrojů pro monitorování a správu počítačové sítě. V závěru kapitoly bude zhodnocena možnost použití popisovaných nástrojů v prostředí počítačové sítě VUT v Brně.

3.1 Cacti

Cacti [10] je kompletní grafické prostředí pro monitorování zařízení v síti vyvíjené pod licencí GPL. Funguje jako nadstavba nástroje RRDtool v podobě webové aplikace vytvořené v PHP a MySQL. Získávání dat se standardně provádí pomocí vestavěného PHP skriptu, ten je však možné nahradit aplikací Spine, což je velmi rychlý dotazovací nástroj napsaný v jazyku C a je vhodné ho použít při sběru dat z velkého množství zařízení. Pro dotazování podporuje především protokol SNMP, dají se však používat i vlastní skripty a pracovat tak se zařízeními, která tento protokol nepodporují. Výchozí interval dotazování je pět minut, ten je možné pouze snižovat na minimální hodnotu deseti sekund.

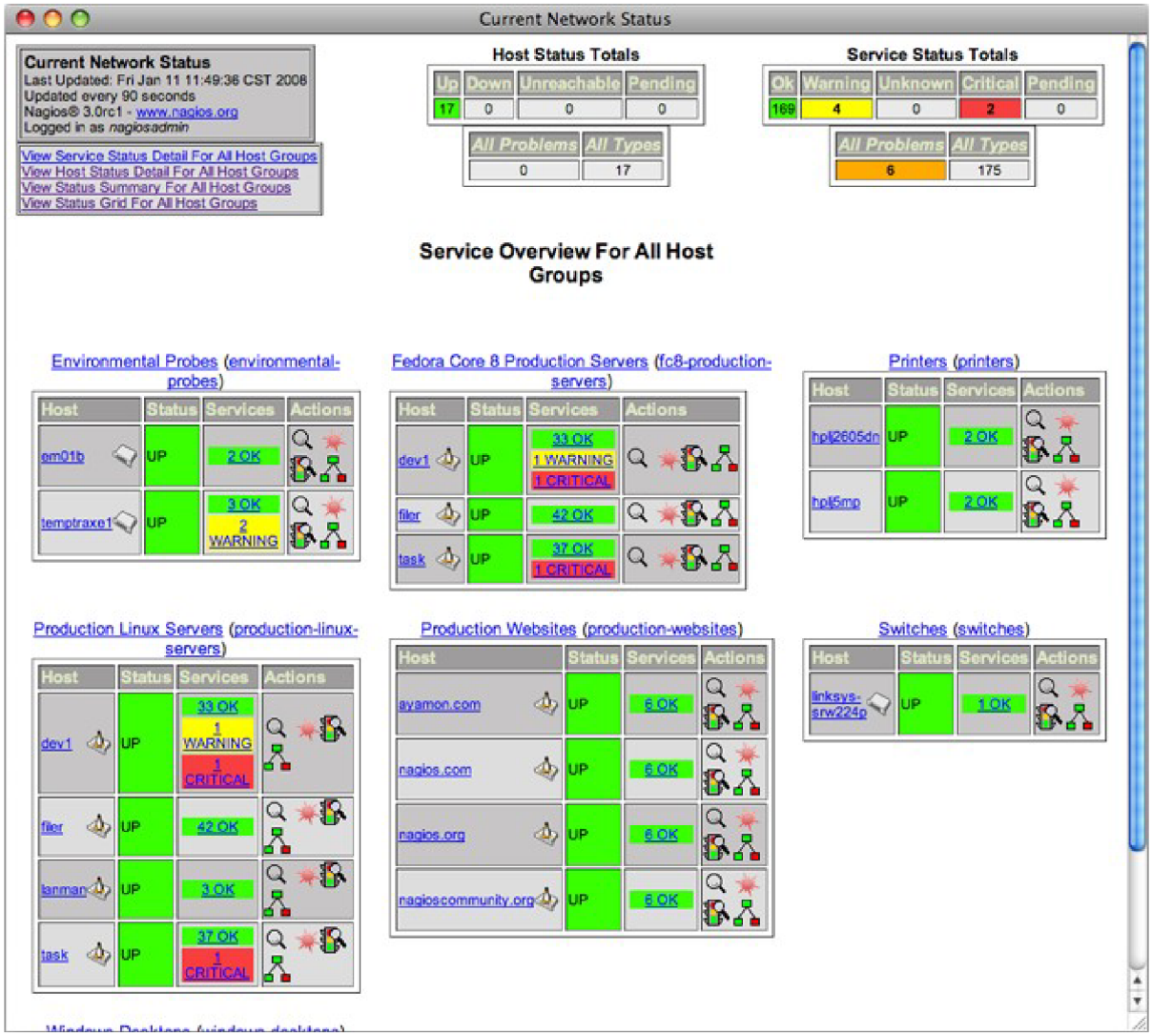
V základní instalaci je možné pomocí webového rozhraní přidávat nová zařízení a vytvářet grafy ze získaných údajů. Pro přidávání zařízení a grafů jsou připravené sady šablon, prostředí poskytuje možnost vytvářet a importovat vlastní šablony. Systém také nabízí správu uživatelských účtů a nastavení oprávnění uživatelů pomocí rolí. Pokročilé funkce (např. vyhledání zařízení podle určitých podmínek, zasílání zpráv správci) v základní instalaci chybí, funkčnost systému se však dá rozšířit pomocí různých doplňků, ať už vlastních nebo vytvářených komunitou soustředěnou kolem Cacti.



Obrázek 3.1: Cacti - ukázka aplikace [10]

3.2 Nagios

Nagios [11] je výkonný NMS vyvíjený pod licencí GPL. Umožňuje monitorovat různé typy zařízení od síťových tiskáren přes aktivní síťové prvky až po koncové stanice a servery. Lze nastavit sledování různých služeb (smtp, pop3, snmp), kontrolovat systémové prostředky (vytížení procesoru, kapacitu pevného disku, přihlášení do systému) a upozorňovat na chyby různými cestami (email, SMS, VoIP). Silnou stránkou je možnost tvorby vlastních doplňků pro monitorování podle potřeb uživatele, podporováno je množství programovacích jazyků (C++, Perl, Python, Ruby, PHP a další). Hotových a dostupných rozšíření existuje velké množství a jsou spravované vývojovou skupinou Nagiosu. Nevýhodou tohoto systému je složitá konfigurace, za účelem jejího zjednodušení začaly vznikat různá rozšíření, jedním z nich je např. Centreon.

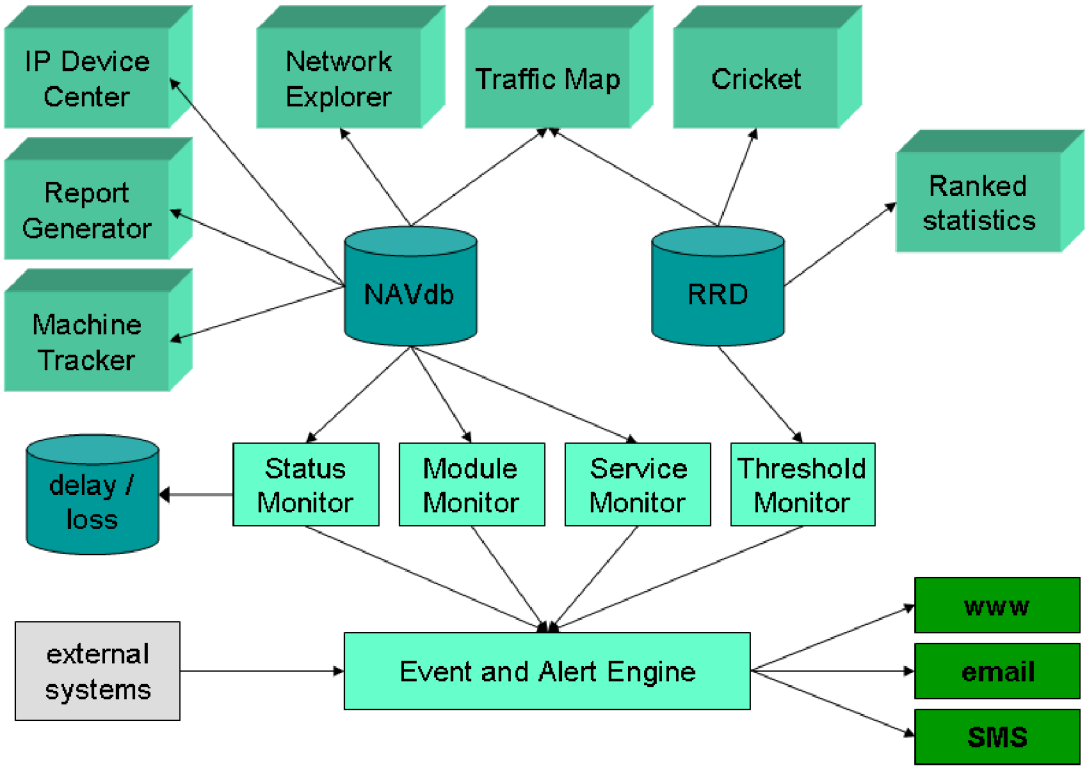


Obrázek 3.2: Nagios - ukázka aplikace [11]

3.3 NAV

Network Administration Visualised (NAV) [12] je nástroj určený pro monitorování rozsáhlých počítačových sítí vyvíjený pod licencí GPL. Jeho vývoj začal v roce 1999 na Norwegian University of Science and Technology (NTNU), od roku 2006 převzala vývoj společnost UNINETT. Systém automaticky zjišťuje topologii sítě, sleduje její vytížení a výpadky.

Systém samotný se skládá z množství propojených modulů, struktura je na obrázku 3.3. Centrálním prvkem je databáze NAVdb (využívá PostgreSQL), v horní části jsou moduly tvořící webové rozhraní, v dolní části jsou procesy starající se o sběr dat a fungování systému. Moduly jsou implementovány v jazycích Java, Python, Perl a PHP.



Obrázek 3.3: Architektura systému NAV [12]

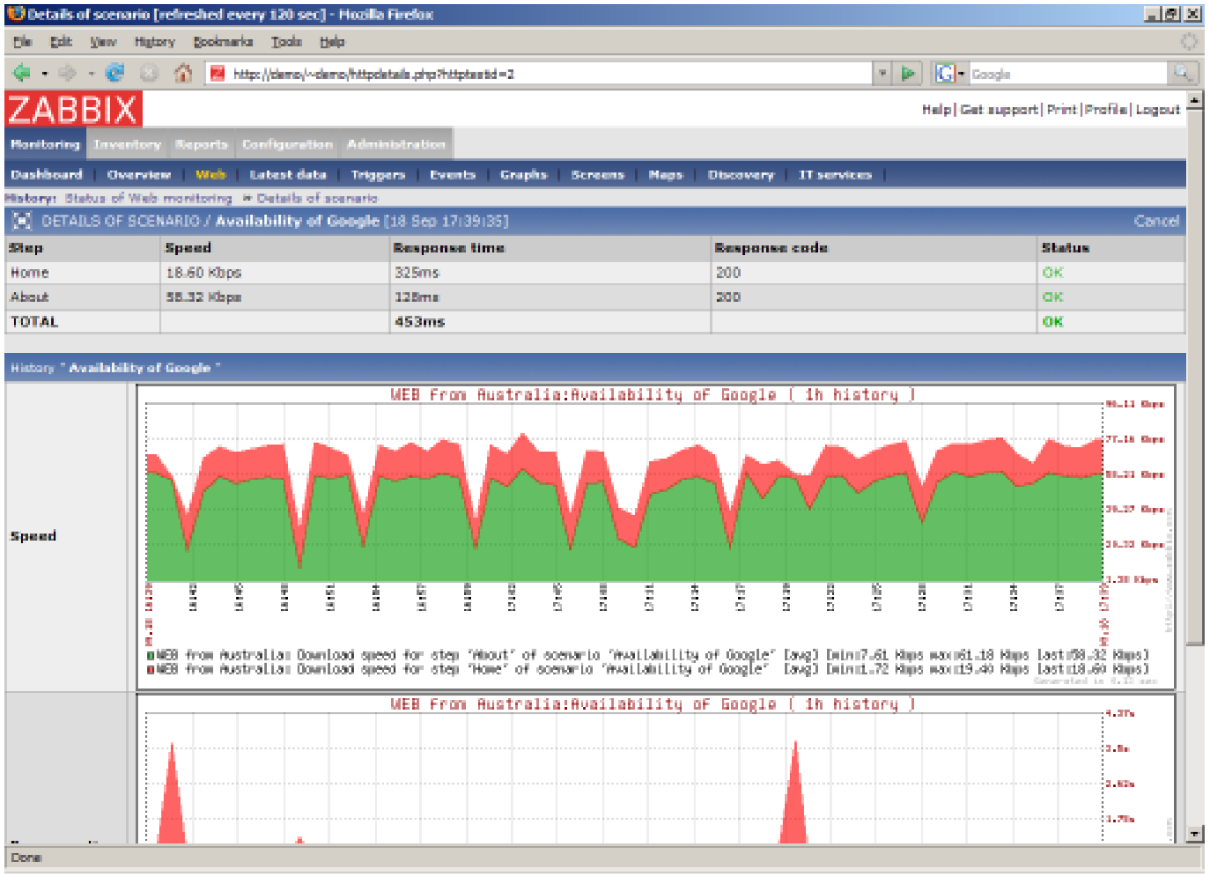
3.4 ZABBIX

Zabbix [13] je systém pro správu sítě šířený pod licencí GPL. Skládá se ze tří částí – serverová část je napsána v C, uživatelské webové rozhraní v PHP a pro ukládání dat využívá SQL databázi, podporuje MySQL, PostgreSQL, SQLite a Oracle. Nabízí několik metod pro sběr údajů ze zařízení:

- dotazování pomocí protokolu SNMP
- testování funkčnosti určitých služeb, např. http, ftp, smtp
- dotazování vlastního agenta běžícího na monitorovaném zařízení

První dvě metody nevyžadují na monitorovaném prvku žádný dodatečný software kromě podpory sledovaných služeb, pro použití třetí metody je nutné nainstalovat na monitorované zařízení agenta, se kterým poté Zabbix komunikuje. Agent je napsaný v C a je k dispozici pro různé platformy a operační systémy a umožňuje monitorovat např. zatížení procesoru, využití diskového prostoru nebo síťový provoz.

Zjištěné hodnoty prezentuje v podobě tabulek a grafů, pro lepší přehled nad monitorovanou sítí nabízí možnost vytvoření síťového diagramu přímo ve webovém rozhraní. Na vzniklé chyby nebo definované události může upozornit např. prostřednictvím emailu, sms či protokolu Jabber.



Obrázek 3.4: Zabbix - ukázka aplikace [13]

3.5 Zhodnocení

Většina zde popsaných NMS umožňuje doplnit a přizpůsobit funkčnost pomocí rozšíření v podobě modulů. To by mělo umožnit při jejich nasazení v počítačové síti VUT v Brně splnění konkrétních požadavků této sítě. Z hlediska výkonnosti a znovupoužitelnosti považuji za vhodné vytvořit vlastní systém pro sběr dat, který bude co nejlépe odpovídat podmínkám počítačové sítě VUT v Brně a který bude v případě potřeby možné použít s existujícími NMS pomocí rozšiřujícího modulu. Návrhem takového systému se zabývá kapitola 4.

Tabulka 3.1 obsahuje porovnání popisovaných NMS. Hodnota „modul“ znamená, že daná vlastnost není v systému dostupná v základní instalaci, je však možné ji doplnit externím modulem. Význam jednotlivých sloupců je následující:

- NMS – název systému
- Modulární – zda je architektura založena na množství samostatných modulů
- Úložiště dat – jaké způsoby systém používá pro zaznamenání monitorovaných dat

- SNMP – zda systém podporuje SNMP
- IPv6 – zda je možné systém nasadit v síti provozující IPv6
- Zasílání zpráv – zda je systém schopný informovat administrátora o výskytu určitých událostí
- Zjištění topologie – zda systém automaticky detekuje zařízení, která se v síti vyskytují
- Mapa topologie – zda systém umožňuje grafické znázornění síťové topologie
- Licence – licence, pod kterou je systém distribuován

NMS	Modulární	Úložiště dat	SNMP	IPv6	Zasílání zpráv	Zjištění topologie	Mapa topologie	Licence
Cacti	ano	RRDtool, MySQL	ano	ano	ano	modul	modul	GPL
Nagios	ano	flat file, SQL	modul	ano	ano	modul	ano	GPL
NAV	ano	RRDtool, Cricket, PostgreSQL	ano	ano	ano	ano	ano	GPL
Zabbix	ano	Oracle, MySQL, PostgreSQL, SQLite	ano	ano	ano	ano	ano	GPL

Tabulka 3.1: Porovnání NMS

4 Analýza a návrh vlastního NMS

Návrh systému pro správu a monitorování síťového provozu je řešen v rámci projektu Analýza počítačových sítí (ANSA). Projekt ANSA se zabývá studiem automatizovaných metod verifikace počítačových sítí, založených na znalosti síťové topologie a konfigurace síťových zařízení. Kromě teoretického výzkumu je cílem projektu vývoj nástrojů, využívající získané teoretické znalosti.

Skupina Analýza počítačových sítí je rozdělena na tři podskupiny s těmito cíli:

a) monitoring a správa sítě
   - nástroje pro online a offline analýzu provozních dat
   - NetFlow, SNMP a další techniky
   - modelování provozu VUT v Brně
   - sledování kvality provozu

b) multicast a kvalita služeb (QoS)
   - sledování, návrh, modelování multicastu v síti s důrazem na kvalitu provozu
   - multicast pro IPv4 a IPv6

c) modelování a analýza
   - metody pro modelování, simulaci a analýzu sítě vzhledem k požadovaným vlastnostem typu garance služeb

Návrh systému pro získávání provozních údajů o počítačové síti, popsaný v následujících kapitolách, spadá do kategorie monitoring a správa sítě. Kromě samotného monitorování sítě je dalším cílem poskytnutí informací o síťových prvcích pro ostatní skupiny, především za účelem modelování sítě.

4.1 Návrh struktury systému

NMS má za úkol sběr dat z aktivních prvků, jejich uchovávání ve vhodné struktuře a poskytnutí uživatelského rozhraní, které umožní operátorovi efektivně získávat informace, které potřebuje. Návrh struktury takového systému se tedy bude sestávat ze tří částí:

- modul pro dotazování aktivních prvků a ukládání získaných dat
- modul databáze uchovávající získaná data
- modul uživatelského rozhraní pro dotazování nad daty a jejich reprezentaci

Aby bylo možné takový systém v případě potřeby rozšiřovat nebo propojit s jinými existujícími systémy, je vhodné řešit tři výše popsané části jako samostatné, vzájemně nezávislé programy. Je předpokládáno nasazení NMS na operačním systému typu unix, tomu budou odpovídat použité programovací nástroje. Návrh jednotlivých modulů bude popisován v následujících kapitolách.

4.2 Modul pro sběr dat

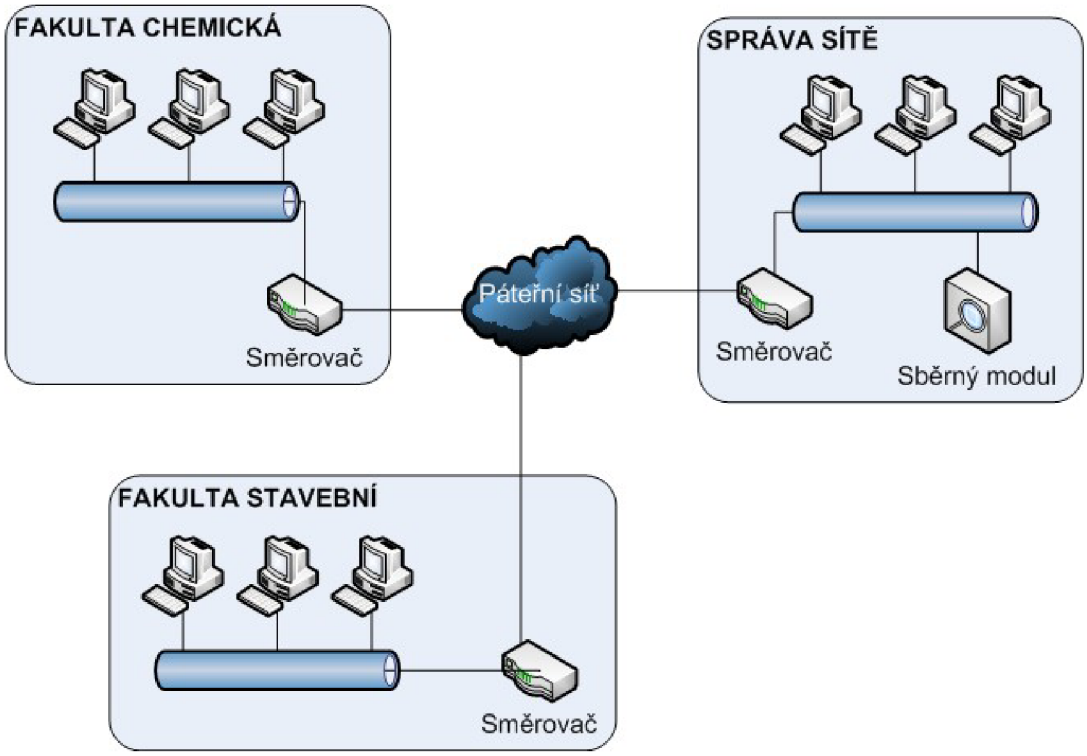
Pro sběr provozních údajů z aktivních prvků je vhodné využít protokol SNMP popisovaný v kapitole 2. Je široce podporován na různých typech aktivních prvků, umožňuje získat jak informace o počtech přenesených dat, užitečných pro výpočet statistik, tak i aktuální konfigurace síťových zařízení, které

je možné využít pro modelování sítě nebo pro dohledání informací užitečných při řešení různých problémů. Pro identifikaci zařízení a získání základních údajů slouží SNMP proměnné popsané v tabulce 2.2, údaje o přenesených datech ukládá SNMP v proměnných popsaných v tabulce 2.3. Proměnné pro zjištění konfigurace síťových zařízení (IP adresy, směrovací tabulky, ARP tabulky) jsou popsané v tabulkách 2.4, 2.5, 2.6, 2.7, 2.8, 2.9.

4.2.1 Architektura

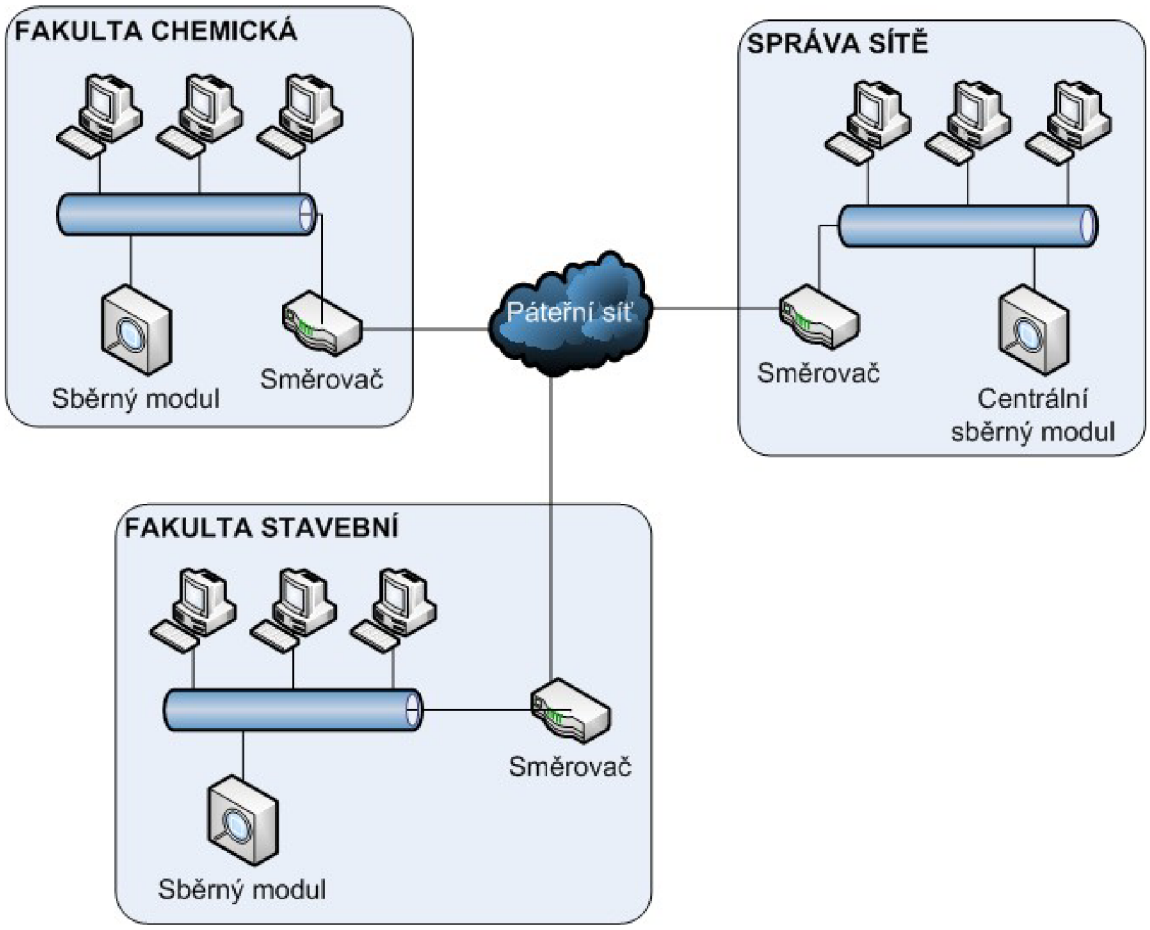
Umístění sběrného modulu v počítačové síti je možné řešit dvěma způsoby:

- použití jednoduché architektury, kde v celé počítačové síti je jediný modul sbírající údaje ze všech prvků (viz obrázek 4.1)
- použití distribuované architektury, kde každý segment sítě má vlastní modul, který shromažďuje data, a po určitém čase předá naráz několik dávek dat centrálnímu sběrnému modulu (viz obrázek 4.2)



Obrázek 4.1: Jednoduchá architektura sběru dat

Výhodou jednoduché architektury je snadnost implementace a přehledná konfigurace seznamu prvků určených k monitorování, protože celý seznam se nachází na jednom místě. Ze stejného důvodu je také snadná aktualizace softwaru při vydání nové verze. Nevýhody této architektury se mohou projevit při nasazení v rozsáhlé síti rozdělené na několik podsítí. Zde může působit problémy zpoždění při přenosu od vzdálených prvků ke sběrači či výpadek určité části sítě, který znemožní doručení dat od některých prvků. Také je nutná dostupnost všech monitorovaných prvků ze segmentu, ve kterém je umístěn sběrný modul, což nemusí být mezi různými podsítěmi vždy splněno.



Obrázek 4.2: Hierarchická architektura sběru dat

Hierarchická architektura výše popsané problémy do jisté míry eliminuje, např. při výpadku sítě mezi lokálním a centrálním sběrným prvkem mohou být data dočasně uložena na lokálním prvku a při obnovení funkčnosti sítě odeslána centrálnímu prvku. Nevýhodou této architektury je složitější implementace a náročnost na konfiguraci a aktualizaci softwaru. Pro zajištění komunikace mezi jednotlivými stanicemi pro sběr dat je možné použít přímo SNMP příkaz *InformRequest* (popsaný v kapitole 2.4) nebo vytvořit vlastní způsob předávání dat na úrovni transportní vrstvy.

Pro implementaci považuji za vhodnější jednoduchý model z několika důvodů:

- jednoduchost implementace
- přehlednost konfigurace
- snadnost aktualizace
- SNMP přenáší relativně malé objemy dat a frekvence dotazování nemusí být vysoká (řádově minuty), z hlediska efektivity přenosu dat počítačovou sítí není tedy hierarchický model nutný.
- Výpadek sítě a nedoručení určité části dat není natolik kritické, protože pro identifikaci příčiny výpadku jsou vhodnější jiné zdroje dat než SNMP, např. NetFlow.

4.2.2 Frekvence sběru dat

Pro frekvenci dotazování není jednoduché určit jediný univerzální interval, možnosti a potřeby sběru dat se pro různá prostředí liší. Záleží na rozloze a vytížení počítačové sítě. Časté SNMP dotazy mohou natolik zatížit aktivní prvky, že u vytížených sítí může kvůli tomu docházet k výpadkům. Problém také může nastat při ukládání dat – vzhledem k tomu, že během jednoho sběru dat z jediného zařízení můžeme získat řádově i tisíce záznamů, bude jejich zpracování a ukládání do databáze trvat dobu, která může být delší než je interval mezi jednotlivými sběry dat, zejména při větším množství monitorovaných zařízení. V takovém případě by docházelo ke spouštění nové sběrné operace před dokončením jejích předchozích běhů, což by postupně vedlo k vyčerpání prostředků sběrné stanice a stavu vyžadujícího zásah správce.

Vzhledem k charakteru dat nepovažuji za nutné opakovat dotazování častěji než každých pět minut (statistiky přenesených dat není nutné sbírat v menších intervalech, změny ARP tabulek by měl tento interval zachytit a změny konfigurací také). Horní mez intervalu by neměla být vyšší než 15 minut, aby nedošlo ke ztracení některé důležité informace (např. MAC adresy škůdce, který se připojil jen na krátkou dobu za účelem provedení určité akce). Přesnou hodnotu z intervalu 5 až 15 minut bude nutné zvolit pro konkrétní prostředí na základě experimentování nebo zkušeností. Nástroje Cacti a Zabbix popsané v kapitole 3 umožňují nastavovat interval dotazování od desítek sekund až po několik minut (Cacti až pět minut, Zabbix až deset minut). Tyto nástroje však nejsou tolik zaměřené na sběr dat z rozsáhlých tabulek jako jsou neighbour cache a tabulka CAM, proto není možné se jejich intervaly pro sběr dat stoprocentně řídit. Nástroj NAV sbírá údaje z neighbour cache a tabulek CAM každých patnáct minut.

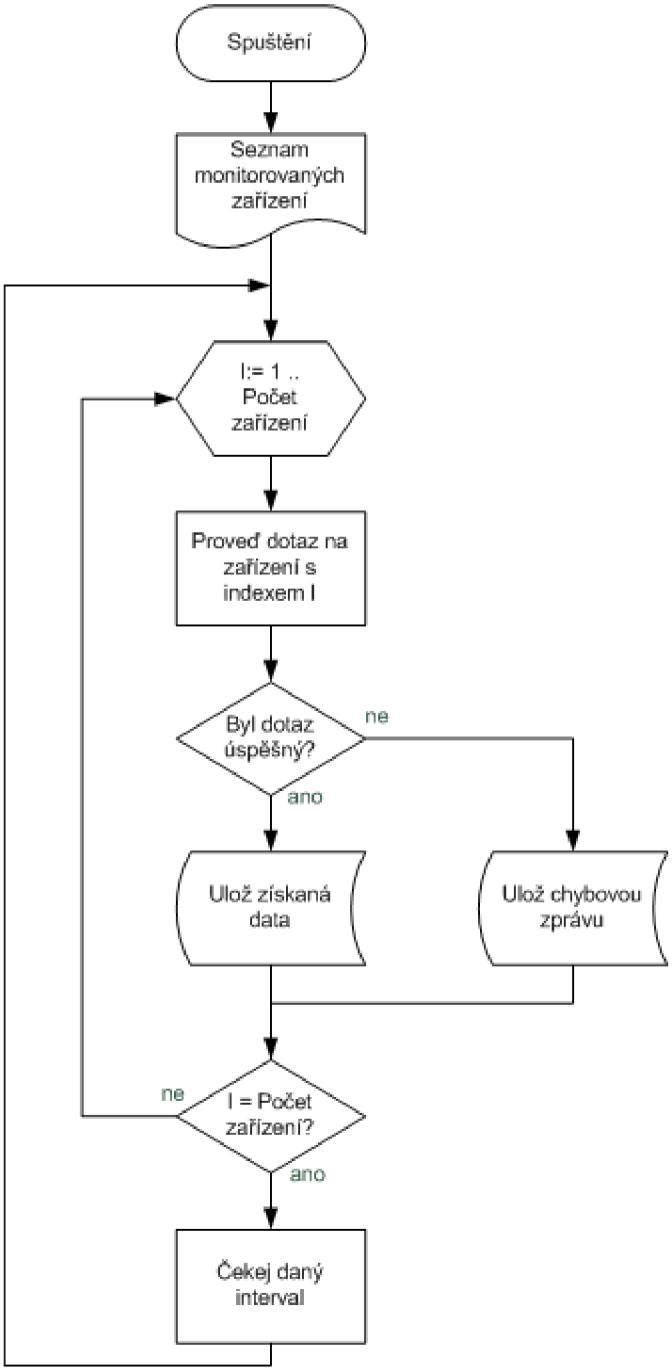
4.2.3 Implementace

Pro implementaci modulu pro sběr dat jsem se rozhodl využít protokol SNMP ve verzi 2c, oproti verzi 1 přináší nový příkaz *GetBulkRequest* a rozšířené čítače a oproti verzi 3 má v současné době širokou podporu ve většině zařízení. Za programovací jazyk jsem zvolil Perl. S jeho podporou v operačních systémech typu unix nebývá problém a pro komunikaci protokolem SNMP obsahuje kvalitní knihovnu `Net::SNMP`, také pro ukládání získaných dat je Perl vybaven modulem `DBI`, který funguje jako univerzální rozhraní mezi aplikací a databázovým strojem, nezávisle na použitém typu databáze.

Vývojový diagram funkčnosti sběrného modulu se nachází na obrázku 4.3. Blok „Seznam monitorovacích zařízení“ načítá z databáze adresy všech prvků určených k monitorování a hodnotu jejich komunity pro čtení dat. Blok „Proveď dotaz na zařízení s indexem I“ volá SNMP příkazy *GetBulkRequest* pro získání proměnných popsaných v tabulkách 2.2 až 2.9. Získané údaje zapíše do databáze, v případě neúspěšného dotazu nebo uložení do databáze zaznamená chybovou zprávu. Po provedení dotazů nad všemi definovanými zařízeními se čeká určený čas a poté se blok dotazování a ukládání dat opakuje.

Pro funkčnost skriptu pro sběr dat je nutné do prostředí Perlu nainstalovat moduly `Net::SNMP`, `DBI` (s ovladačem `DBD::Pg`) a `Parallel::ForkManager`. Pro zvýšení přehlednosti a znovupoužitelnosti skriptu jsou určité části kódu realizovány formou samostatných tříd, které logickou strukturou odpovídají struktuře dotazovaných SNMP tabulek. Jedná se o třídy `Device`, `Interface`, `Ip`, `Mac` a `NetToPhysical`, které jsem navrhl sám. Popis použitých modulů a vlastních tříd následuje níže. Pro přehlednost je na obrázku 4.4 znázorněn diagram použití jednotlivých modulů a tříd ve skriptu.

V příloze B se nachází ukázka SNMP dotazu prováděného skriptem v jazyku Perl, která získá ze serveru `isa.fit.vutbr.cz` popis zařízení, unikátní ID, dobu běhu systému a podporované síťové vrstvy.



Obrázek 4.3: Vývojový diagram sběrného modulu

Modul Net::SNMP

Modul `Net::SNMP` implementuje objektově orientované rozhraní pro práci s protokolem SNMP. Každá instance třídy představuje individuální mapování mezi tímto objektem a vzdáleným zařízením podporujícím protokol SNMP. Modul v použité verzi 6.0.0 podporuje SNMP ve verzích 1, 2c, 3 a umožňuje komunikovat se zařízením buď blokujícím nebo neblokujícím způsobem. Při použití blokujícího způsobu se běh skriptu po odeslání dotazu zastaví, dokud neobdrží odpověď (výsledky dotazu jsou vráceny jako reference na hash) nebo nevyprší časový limit. Tento způsob má výhodu ve snadné implementaci a vyšší přehlednosti výsledného kódu, pokud však potřebujeme na zařízení

odeslat větší množství dotazů, je výhodnější použít neblokující způsob. Ten zařadí každý dotaz do fronty a pokračuje ve vykonávání kódu. K odeslání všech požadavků dojde hromadně po zavolání metody `snmp_dispatcher`. Odpovědi jsou poté předány uživatelské metodě (zadané při definici dotazu jako jeden z povinných parametrů) ve formě atributu obsahujícího referenci na objekt. Samotné výsledky získáme zavoláním metody `var_bind_list` objektu. Ta vrací referenci na hash – klíče hashe jsou identifikátory SNMP objektu v tečkové notaci. Pokud metoda `var_bind_list` vrátí nedefinovanou hodnotu, potom během vykonávání dotazu došlo k chybě. Pro zjištění o jakou chybu se jedná slouží metoda `error` objektu.

Spojení s cílovým zařízením se inicializuje metodou `session`, která umožňuje zadat parametry pro adresu zařízení, komunitu (u SNMPv3 autentizační údaje), zda se bude komunikovat blokujícím či neblokujícím způsobem, počet opakování při neúspěchu, čas pro timeout a další.

Pro komunikaci se zařízením je využívána výhradně metoda `get_bulk_request`. Ta umožňuje v rámci jediného dotazu získat hodnoty většího množství SNMP objektů, má však jistá omezení. Je možné dotazovat libovolné množství jednorozměrných veličin (jejich počet je dán parametrem `nonrepeaters`) a navíc volitelně jednu vícerozměrnou veličinu (tabulku). Není však možné v rámci jednoho volání funkce dotazovat více než jednu vícerozměrnou veličinu. V takovém případě nedojde k získání všech požadovaných výsledků (aniž by o tom bylo informováno např. chybovou zprávou). Potřeba dotazovat několik vícerozměrných veličin naráz nastala při snaze získávat z některých SNMP tabulek pouze určité části. Např. u tabulky *ipNetToPhysicalTable*, která zaznamenává mapování síťových adres na fyzické, nebylo nutné získávat všech pět objektů, které tabulka k dané adrese uchovává. Pro získání potřebných informací stačí hodnoty pouze tří z těchto pěti objektů. Protože tato tabulka může obsahovat až tisíce záznamů, je dotazování pouze určitých částí významné pro snížení přenášených objemů dat. V modulu `Net::SNMP` je nutné pro každou část tabulky volat novou metodu `get_bulk_request` s požadovaným OID.

Užitečnou metodou je `oid_base_match`, která za parametry očekává dvě hodnoty OID v tečkové notaci a vrací pravdivou hodnotu, pokud druhé OID je shodné nebo potomek prvního v MIB. Při získávaní údajů pomocí `get_bulk_request` je tato metoda využívána pro kontrolu, zda další získaný objekt již neleží mimo rozsah požadované části MIB stromu.

Kompletní popis modulu `Net::SNMP` se nachází v manuálové stránce uvedené ve zdroji [14], odkud také byly čerpány informace pro tuto kapitolu.

Modul DBI

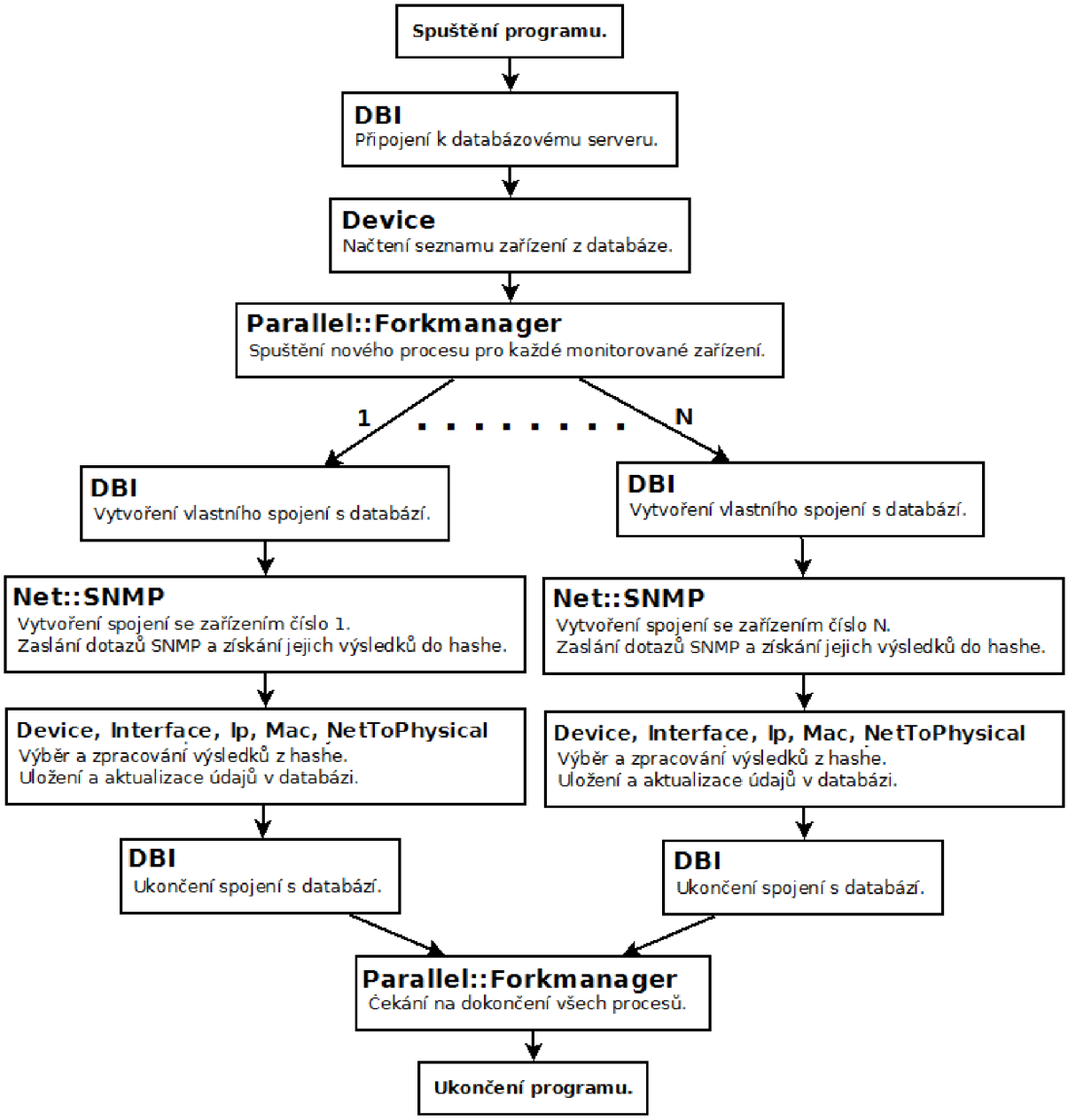
Modul poskytuje rozhraní pro práci s databází, které je nezávislé na typu použité databáze. Pro fungování s konkrétním typem databáze je nutné zvolit vhodný ovladač a zavést ho v konstruktoru při vytváření instance třídy `DBI` modulu. V našem případě se jedná o ovladač `DBD::Pg` pro komunikaci s databázovým serverem PostgreSQL. Využití nezávislého modulu `DBI` přináší v případě potřeby možnost snadné změny využívaného typu databázového serveru. Např. při přechodu databázový server MySQL stačí pouze zavést ovladač `DBD::mysql` a sběrný modul bude bez nutnosti dalších úprav pracovat s touto databází. Jediné problémy může přinést reprezentace datových typů pro IP adresy a MAC adresy – pro tyto údaje PostgreSQL nabízí nativní datové typy `inet`, `cidr` a `mac`, kdežto u jiných databázových serverů se podpora těchto typů může lišit. To však není nutné řešit v rámci sběrného modulu ale při vytváření databázových tabulek. Tato problematika je podrobněji rozebrána v kapitole 4.3.

Pro komunikaci s databází jsou využívány tyto metody `DBI` modulu:

- `connect` a `clone` pro vytvoření spojení na databázi
- `do`, `prepare` a `execute` pro zaslaní SQL dotazu
- `selectrow_hashref`, `fetchrow_hashref` pro získání výsledků dotazu
- `last_insert_id` pro získání identifikátoru posledního řádku vloženého do tabulky

- `disconnect` pro ukončení spojení s databází

Kompletní popis modulu `DBI` a ovladače `DBD::Pg` se nachází v manuálových stránkách, uvedených ve zdrojích [15] a [16], ze kterých byly čerpány informace pro tuto kapitolu.



Obrázek 4.4: Schéma použití modulů a tříd ve skriptu pro sběr dat

Modul Parallel::Forkmanager

Tento modul poskytuje vyšší úroveň abstrakce nad použitím funkce Perlu `fork`. Umožňuje provádět souběžně ty části kódu, které jsou vhodné pro paralelní zpracování, především cykly bez vzájemných závislostí (pro každou smyčku spouští kopii kódu s vlastními instancemi proměnných). Ve sběrném modulu je pro paralelní zpracování vhodná část, kde se v cyklu provádí výběr monitorovaného zařízení ze seznamu zařízení, jeho dotazování a ukládání získaných dat. S využitím tohoto modulu je možné provádět dotazování prvků téměř ve stejnou chvíli, kdežto u sériového zpracování je nutné čekat na dokončení sběru a ukládání dat jednoho prvku, než je možné dotazovat prvek další. To může

vést v případě sledování většího množství prvků i k několikaminutovým rozdílům mezi časem platnosti dat prvního a posledního prvku a působit problémy při reprezentaci určitých údajů.

Konstruktor třídy `Parallel::Forkmanager` očekává jako parametr číslo, udávající maximální počet souběžně spuštěných procesů. Pro vytvoření nového procesu slouží metoda `start`, pro jeho ukončení slouží metoda `finish`. Rodičovský proces volá za tělem paralelizovaného cyklu metodu `wait_all_children`, která umožní pokračovat rodičovskému procesu v běhu až po dokončení běhu všech potomků. Důležité je správné nastavení maximálního počtu paralelně spuštěných procesů, které závisí na výkonu sběrné stanice a databázového serveru. V případě nastavení příliš vysokého počtu procesů může dojít k vyčerpání systémových zdrojů, čím nižší bude počet procesů, tím více poroste rozdíl platnosti dat v rámci jednoho sběru mezi prvním a posledním monitorovaným prvkem.

Informace o tomto modulu byly čerpány z manuálových stránek modulu [17].

Třída Device

Třída zapouzdřuje údaje vztahující se k právě jednomu monitorovanému prvku. Známé údaje jsou nejprve načteny z databáze, po provedení SNMP dotazu aktualizovány a uloženy zpět do databáze.

Atributy třídy jsou:

- `id` – identifikátor řádku v databázové tabulce zařízení
- `ip_address` – IP adresa pro zasílání SNMP dotazů
- `hostname` – hostname pro zasílání SNMP dotazů
- `community` – heslo, které má zařízení nastaveno pro SNMP komunikaci
- `object_id` – identifikátor SNMP objektu zařízení (OID *sysObjectID*)
- `description` – textový popis zařízení (OID *sysDescr*)
- `contact` – kontaktní údaje na správce zařízení (OID *sysContact*)
- `name` – správcem přiřazený název zařízení (OID *sysName*)
- `location` – informace o fyzickém umístění zařízení (OID *sysLocation*)
- `services` – síťové vrstvy, na kterých zařízení pracuje (OID *sysServices*)
- `id_device_check` – identifikátor do tabulky, udržující informace o provedených SNMP dotazech, ukazující na aktuální sběr dat
- `id_device_last_check` – identifikátor do tabulky, udržující informace o provedených SNMP dotazech, ukazující na sběr dat předcházející tomuto sběru
- `id_device_check_status` – status, se kterým skončil aktuálně prováděný sběr dat

Pro přístup k atributům slouží metoda `get`, která za parametry očekává názvy požadovaných atributů a vrací skalár, pokud je volána s jedním parametrem, pokud je volána s více parametry tak vrací jejich hodnoty v poli.

Minimálně jeden z atributů `ip_address` nebo `hostname` musí být zadaný, pokud jsou zadané oba je dána přednost atributu `ip_address`.

Pro použití SNMP ve verzích 1 a 2c obsahuje tato třída všechny potřebné atributy, v případě použití SNMP verze 3 by bylo nutné třídu rozšířit o atributy reprezentující autentizační údaje.

Třída Interface

Třída udržuje údaje, týkající se jednoho síťového rozhraní na daném zařízení v okamžiku sběru dat. Všechny údaje jsou získány během SNMP dotazu a vloženy do databáze, v případě existence daného rozhraní v databázi jsou jeho hodnoty aktualizovány. Pro přístup k atributům a jejich nastavení opět slouží metody `get` a `set` jako u předchozí třídy.

Atributy třídy jsou:

- `id` – identifikátor řádku v databázové tabulce rozhraní
- `index` – index síťového rozhraní (OID *ifIndex*)
- `description` – informace o rozhraní (OID *ifDescr*)
- `id_type` – celočíselná hodnota reprezentující protokol rozhraní (OID *ifType*)
- `speed` – aktuální přenosová rychlost rozhraní (OID *ifSpeed*)
- `phys_address` – fyzická adresa druhé síťové vrstvy (OID *ifPhysAddress*)
- `id_admin_status` – administrativní stav rozhraní (OID *ifAdminStatus*)
- `id_oper_status` – operační stav rozhraní (OID *ifOperStatus*)
- `in_octets` – celkový počet bajtů přijatých rozhraním (OID *ifInOctets*)
- `in_ucast_pkts` – počet přijatých unicast paketů (OID *ifInUcastPkts*)
- `in_nucast_pkts` – počet přijatých paketů jiných než unicast (OID *ifInNUcastPkts*)
- `in_discards` – počet zahozených příchozích paketů, přestože nebyly chybné (OID *ifInDiscards*)
- `in_errors` – počet zahozených příchozích paketů, protože byly chybné (OID *ifInErrors*)
- `in_unknown_protos` – počet příchozích paketů neznámého nebo nepodporovaného protokolu (OID *ifInUnknownProtos*)
- `out_octets` – celkový počet bajtů odeslaných rozhraním (OID *ifOutOctets*)
- `out_ucast_pkts` – počet odeslaných unicast paketů (OID *ifOutUcastPkts*)
- `out_nucast_pkts` – počet odeslaných paketů jiných než unicast (OID *ifOutNUcastPkts*)
- `out_discards` – počet zahozených odchozích paketů, přestože nebyly chybné (OID *ifOutDiscards*)
- `out_errors` – počet zahozených odchozích paketů, protože byly chybné (OID *ifOutErrors*)
- `out_qlen` – celkový počet paketů v odchozí frontě (OID *ifOutQLen*)

Třída Ip

Třída udržuje údaje o konfiguraci IP adresy právě jednoho rozhraní na daném zařízení v okamžiku sběru dat. Všechny údaje jsou získány během SNMP dotazu a vloženy do databáze. V případě, že daná konfigurace již v databázi existuje a má konec intervalu platnosti roven času posledního sběru dat, potom je tento interval platnosti prodloužen do času aktuálního sběru dat. Pro přístup k atributům a jejich nastavení slouží metody `get` a `set`.

Atributy třídy jsou:

- `id` – identifikátor řádku v databázové tabulce konfigurací IP adres
- `id_device` – identifikátor řádku v databázové tabulce zařízení, ke kterému se konfigurace vztahuje
- `id_interface` – identifikátor řádku v databázové tabulce rozhraní, ke kterému se konfigurace vztahuje
- `index_interface` – index rozhraní (OID *ipAddressIfIndex*, případně *ipAdEntIfIndex* pokud zařízení nepodporuje tabulku *ipAddressTable*)
- `id_type` – typ adresy (OID *ipAddressType*, případně „1“ pokud zařízení nepodporuje tabulku *ipAddressTable*, ve které hodnota „1“ odpovídá IPv4 adrese)
- `ip_address` – hodnota IP adresy získaná ze SNMP OID proměnné *ipAddressIfIndex* (případně přímo hodnota SNMP proměnné *ipAdEntAddr* pokud zařízení nepodporuje tabulku *ipAddressTable*)

Třída Mac

Třída obsahuje informace, které odpovídají právě jednomu řádku tabulky CAM zařízení v době sběru dat. Získané údaje jsou do databáze buď vloženy jako nové nebo dojde k aktualizaci existujícího záznamu (za stejných podmínek jako u třídy `Ip`). Pro přístup k atributům a jejich nastavení slouží metody `get` a `set`.

Atributy třídy jsou:

- `id` – identifikátor řádku v databázové tabulce MAC adres
- `id_device` – identifikátor řádku v databázové tabulce zařízení
- `id_interface` – identifikátor řádku v databázové tabulce rozhraní
- `port` – hodnota portu, ze kterého byla adresa naučena (OID *dot1qTpFdbPort*)
- `vlan_id` – hodnota VLAN_ID získaná ze SNMP OID proměnné *dot1qTpFdbPort*
- `id_status` – status záznamu (OID *dot1qTpFdbStatus*)
- `mac_address` – hodnota MAC adresy získaná ze SNMP OID proměnné *dot1qTpFdbPort*

Třída NetToPhysical

Třída obsahuje údaje, které odpovídají právě jednomu řádku tabulky, ve které zařízení udržuje informace o mapování síťových adres na fyzické. Údaje jsou opět do databáze vkládány jako nové nebo dochází k aktualizaci již existujících záznamů a to za stejných podmínek jako v případě třídy `Ip`. Pro přístup k atributům a jejich nastavení slouží metody `get` a `set`.

Atributy třídy jsou:

- `id` – identifikátor řádku v databázové tabulce mapující síťové adresy na fyzické
- `id_device` – identifikátor řádku v databázové tabulce zařízení
- `id_interface` – identifikátor řádku v databázové tabulce rozhraní
- `index_interface` – hodnota indexu rozhraní získaná ze SNMP OID proměnné *ipNetToPhysicalPhysAddress* (nebo *ipNetToMediaPhysAddress* pokud zařízení nepodporuje tabulku *ipNetToPhysicalTable*)

- `id_address_type` – typ adresy získaný ze SNMP OID proměnné *ipNetToPhysicalPhysAddress* (nebo hodnota „1“ pokud zařízení nepodporuje tabulku *ipNetToPhysicalTable*, ve které hodnota „1“ odpovídá IPv4 adrese)
- `id_type` – způsob vzniku mapování získaný ze SNMP proměnné *ipNetToPhysicalType* (nebo *ipNetToMediaType* pokud zařízení nepodporuje tabulku *ipNetToPhysicalTable*)
- `id_state` – stav dostupnosti souseda na rozhraní, ke kterému se mapování vztahuje (OID *ipNetToPhysicalState*, případně hodnota „6“ pokud zařízení nepodporuje tabulku *ipNetToPhysicalTable*, ve které hodnota „6“ odpovídá stavu „unknown“)
- `mac_address` – hodnota MAC adresy získaná ze SNMP proměnné *ipNetToPhysicalPhysAddress* (nebo hodnota proměnné *ipNetToMediaPhysAddress* pokud zařízení nepodporuje tabulku *ipNetToPhysicalTable*)
- `ip_address` – hodnota IP adresy získaná ze SNMP OID proměnné *ipNetToPhysicalPhysAddress* (nebo z OID proměnné *ipNetToMediaPhysAddress* pokud zařízení nepodporuje tabulku *ipNetToPhysicalTable*)

4.2.4 Získávání dat o síti do databáze

Výsledky získané z dotazovaného zařízení pomocí protokolu SNMP a reprezentované v Perlu modulem `Net::SNMP` je nutné před jejich uložením do databáze vhodně upravit a doplnit o určité vazby tak, aby po jejich uložení bylo možné reprezentovat požadované informace ve vyhovující podobě. Pokračování této kapitoly se bude věnovat popisu struktury dat získaných ze SNMP dotazů a budou popsány kroky, které sběrný skript provádí, aby data upravil do vyhovující podoby. Protože běh skriptu lze logicky rozdělit do několika bloků, kde každý blok zpracovává určitou skupinu údajů (reprezentovanou třídami popsanými v předchozí kapitole), bude i zbytek této kapitoly rozdělen do částí podle jednotlivých tříd.

Získání informací o zařízeních (třída Device)

Informace o zařízení jsou získávány ze SNMP tabulky *system* (viz tabulka 2.2, str. 8). Pro získání všech proměnných se provede jedno volání funkce `get_bulk_request` se všemi objekty OID v tabulce *system* (jedná se o jednorozměrné veličiny) a výsledky odpovídající dotazovaným OID již není nutné upravovat a je možné je rovnou použít pro aktualizaci databáze.

Získání informací o síťových rozhraních (třída Interface)

Pro získání údajů o rozhraních monitorovaného zařízení je dotazována tabulka *ifTable* (viz tabulka 2.3, str. 9) a tento dotaz je zaslán společně s voláním funkce `get_bulk_request` na proměnné tabulky *system*, kde OID tabulky *ifTable* figuruje jako jediná vícerozměrná veličina. Z výsledků je nejprve na základě všech proměnných *ifIndex* vytvořen seznam indexů existujících rozhraní. Když jsou známy indexy rozhraní, tak je možné zbývající proměnné tabulky *ifTable* přiřadit ke konkrétním rozhraním na základě jejich indexů (rozšířením OID zbývajících proměnných o hodnotu indexu, např. OID *ifDescr.5* vrátí popis rozhraní s indexem 5). Všechny takto získané hodnoty je možné rovnou použít pro aktualizaci databáze s výjimkou jediné – *ifPhysAddress*, tu modul `Net::SNMP` reprezentuje v binární podobě a před uložením do databáze je nutné ji převést na řetězec pomocí funkce Perlu `unpack` s parametrem `H*`.

Získání informací o konfiguraci IP adres (třída Ip)

Konfiguraci IP adres je možné získat z několika různých SNMP tabulek. Aktuální a preferovaná je tabulka *ipAddressTable*, která obsahuje jak IPv4 tak IPv6 adresy. Ta však nemusí být na každém

zařízení podporovaná (při testování takový případ nastal u zařízení 3Com Switch 4800G s verzí softwaru 5.20). V takovém případě je možné dotazovat tabulku *ipAddrTable*, která však obsahuje pouze IPv4 adresy a v současné době má status *deprecated*. Také způsob zpracování získaných údajů je u každé z tabulek trochu odlišný.

Nejprve bude popsáno zpracování tabulky *ipAddressTable* (její struktura je detailně popsána v tabulce 2.4, str. 10). Ve sběrném modulu není dotazována celá tabulka pomocí OID *ipAddressTable*, protože pro získání potřebných informací stačí dotazovat pouze dvě z jejích devíti proměnných. Proto je dvakrát volána funkce `get_bulk_request`, jednou s OID *ipAddressIfIndex*, podruhé s OID *ipAddressPrefix*. Hodnotu IP adresy není možné dotazovat přímo pomocí OID *ipAddressAddr* (to má metodu přístupu definovanou jako *not-accessible*). Její hodnota je však vrácena v odpovědi jako část řetězce, připojeného za OID proměnné *ipAddressIfIndex* a její získání probíhá takto:

- vrácené OID *ipAddressIfIndex* pro IPv4 adresu má podobu *1.3.6.1.2.1.4.34.1.3.T.L.A.B.C.D*, kde T je typ IP adresy (1 pro IPv4, 2 pro IPv6), L označuje délku zbývající části OID (4 pro IPv4, 16 pro IPv6) a A.B.C.D je hodnota samotné IPv4 adresy decimálně
- konkrétní hodnoty mohou být např. *1.3.6.1.2.1.4.34.1.3.1.4.147.229.252.85*
- po odstranění řetězce odpovídajícího OID *ipAddressIfIndex* zůstane řetězec *1.4.147.229.252.85*
- odebráním hodnot 1 a 4 ze začátku řetězce zjistíme typ adresy a její délku a zbývající řetězec je hledaná hodnota IP adresy
- v případě, že by se jednalo o IPv6 adresu, je navíc nutné ji před dalším zpracováním převést z decimální do hexadecimální podoby
- pro zjištění síťové masky této konkrétní IP adresy je proveden dotaz na OID *ipAddressPrefix.147.229.252.85*, výsledkem je odkaz do tabulky prefixů, který má podobu *1.3.6.1.2.1.4.32.1.5.921.1.4.147.229.252.85.30*, délce prefixu odpovídá poslední decimální hodnota, zbytek je zahozen
- výsledná podoba této IP adresy, která bude vložena do databáze je tedy 147.229.252.85/30

Zpracování tabulky *ipAddrTable* (detailně popsána v tabulce 2.5, str. 10) je vlivem odlišné struktury rozdílné. Funkce `get_bulk_request` je volána třikrát a to pro OID *ipAdEntAddr*, *ipAdEntIfIndex* a *ipAdEntNetMask*. Hodnotu IP adresy je možné získat rovnou z výsledku proměnné *ipAdEntAddr* (ta má oproti proměnné *ipAddressAddr* z tabulky *ipAddressTable* metodu přístupu *read-only*). Hodnoty indexu a síťové masky získáme také přímo z hodnot proměnných *ipAdEntIfIndex* a *ipAdEntNetMask*, síťovou masku je však před dalším zpracováním třeba upravit z tvaru A.B.C.D na tvar /X.

Získání MAC adres ze síťových zařízení (třída Mac)

Pro získání přepínací tabulky CAM zařízení obsahující přiřazení naučených MAC k portům je jednou volána funkce `get_bulk_request` s dotazem na celou tabulku (OID *dot1qTpFdbTable*, detailně popsána v tabulce 2.6, str. 11). Cílem je získat hodnoty MAC adres, jejich přiřazení do VLAN, porty ke kterým jsou přiřazeny a způsob, jakým byly naučeny. Přiřazení k portu a způsob naučení lze zjistit rovnou jako hodnotu proměnných *dot1qTpFdbPort* a *dot1qTpFdbStatus*. MAC adresu nelze získat přímo z proměnné *dot1qTpFdbAddress*, protože ta má metodu přístupu definovanou jako *not-accessible*. Stejně tak údaj o VLAN není možné zjistit přímo, protože tabulka *dot1qTpFdbTable* nemá pro tento údaj definované žádné OID. Jak MAC adresu, tak přiřazení do VLAN je však možné získat jako podřetězec, připojený v odpovědi za OID proměnné *dot1qTpFdbPort*.

Postup získání hodnoty MAC adresy je následující:

- vrácené OID *dot1qTpFdbPort* má podobu *1.3.6.1.2.1.17.7.1.2.2.1.2.V.A.B.C.D.E.F*, kde V je údaj o přiřazení do VLAN a A.B.C.D.E.F je hodnota MAC adresy decimálně
- konkrétní hodnoty mohou být např. *1.3.6.1.2.1.17.7.1.2.2.1.2.8.0.28.46.146.3.128*
- po odstranění řetězce odpovídajícího OID *dot1qTpFdbPort* zůstane řetězec *8.0.28.46.146.3.128*
- odebráním hodnoty 8 ze začátku řetězce zjistíme příslušnost do VLAN a zbývající část řetězce je MAC adresa, kterou je ještě třeba převést do hexadecimálního tvaru, výsledná hodnota MAC adresy potom bude 00:1C:2E:92:03:80

Se získaným údajem o přiřazení do VLAN je však spojena ještě jedna komplikace. U některých zařízení vyjadřuje získaná hodnota přímo identifikátor VLAN, tedy VLAN_ID, u jiných zařízení se však jedná pouze o index, identifikující řádek v SNMP tabulce *dot1qVlanCurrentTable*, kde je možné k danému identifikátoru zjistit skutečné VLAN_ID z OID proměnné *dot1qVlanFdbId*. První způsob byl pozorován u zařízení 3Com, druhý způsob u zařízení HP. U zařízení HP je tedy navíc nutné volat ještě jednou funkci `get_bulk_request` s OID *dot1qVlanFdbId* pro zjištění skutečné hodnoty VLAN_ID.

Mapování IP adres na MAC adresy (třída NetToPhysical)

Informace o mapování síťových adres na fyzické je možné získat z několika SNMP tabulek. Aktuální a preferovaná je *ipNetToPhysicalTable* (popsaná v tabulce 2.9, str. 12), která obsahuje jak IPv4 adresy tak IPv6 adresy. Ta však nemusí být na všech zařízeních podporovaná (takový případ opět pozorován u zařízení 3Com), potom je možné dotazovat tabulku *ipNetToMediaTable* (popsaná v tabulce 2.8, str. 12), která však obsahuje pouze IPv4 adresy a v současné době má status *deprecated*.

Nejprve bude popsáno zpracování tabulky *ipNetToPhysicalTable*. Protože celá tabulka je poměrně rozsáhlá, jsou dotazovány pouze tři z jejích pěti přístupných proměnných. Volaní funkce `get_bulk_request` proběhne třikrát pro OID *ipNetToPhysicalPhysAddress*, *ipNetToPhysicalType* a *ipNetToPhysicalState*. Cílem je získat hodnotu IP adresy a příslušející MAC adresy, index rozhraní ke kterému náležejí, typ IP adresy, způsob vzniku mapování a stav dostupnosti sousedního zařízení. MAC adresu, způsob mapování a stav dostupnosti je možné získat přímo z proměnných *ipNetToPhysicalPhysAddress*, *ipNetToPhysicalType* a *ipNetToPhysicalState*, jen hodnotu MAC adresy je třeba převést do textové podoby pomocí funkce Perlu `unpack` s parametrem `H*`. Zbývající informace je nutné získat z řetězce připojeného k vrácenému OID proměnné *ipNetToPhysicalPhysAddress*, protože jim odpovídající SNMP proměnné *ipNetToPhysicalIfIndex*, *ipNetToPhysicalNetAddress* a *ipNetToPhysicalNetAddressType* mají metodu přístupu definovanou jako *not-accessible*. Získání probíhá tímto způsobem:

- vrácené OID *ipNetToPhysicalPhysAddress* má tvar *1.3.6.1.2.1.4.35.1.4.R.T.L.A.B.C.D*, kde R je index rozhraní, T je typ IP adresy (1 pro IPv4, 2 pro IPv6), L je délka IP adresy (4 pro IPv4, 16 pro IPv6) a A.B.C.D je hodnota IPv4 adresy v decimální podobě (IPv6 by byla zapsána stejným způsobem – 16 decimálních číslic oddělených tečkami)
- konkrétní hodnoty mohou být např. *1.3.6.1.2.1.4.35.1.4.961.1.4.147.229.252.125*
- po odstranění řetězce odpovídajícího OID *ipNetToPhysicalPhysAddress* zůstane řetězec *961.1.4.147.229.252.125*, kde vyjmutím hodnoty 961 získáme index rozhraní (vysoké číslo naznačuje, že se nejedná o fyzické rozhraní ale spíše virtuální VLAN rozhraní), po vyjmutí 1 a 4 získáme typ adresy a její délku a zbytkem je hledaná IP adresa
- výslednou adresu 147.229.252.125 můžeme rovnou použít pro aktualizaci databáze, v případě, že by se jednalo o IPv6 adresu tak by bylo nutné ji nejprve převést do hexadecimálního tvaru a podoby akceptované databázovým datovým typem `inet`

Pro zpracování tabulky *ipNetToMediaTable* je dvakrát volána funkce `get_bulk_request` (pro OID *ipNetToMediaPhysAddress* a *ipNetToMediaType*). Hodnotu MAC adresy je opět možné získat přímo z výsledku SNMP proměnné *ipNetToMediaPhysAddress* (opět je nutný převod do textové podoby voláním funkce `unpack`), stejně tak způsob mapování z hodnoty proměnné *ipNetToMediaType*. Typ adresy bude v tomto případě vždy „1“, stav dostupnosti bude pokaždé hodnota „6“ (tuto hodnotu má IPv4 adresa v tabulce *ipNetToPhysicalTable* vždy). Hodnotu IP adresy a index rozhraní získáme z řetězce OID proměnné *ipNetToMediaPhysAddress* podobným způsobem, jako v předchozím případě:

- vrácené OID *ipNetToMediaPhysAddress* má tvar *1.3.6.1.2.1.4.22.1.2.R.A.B.C.D*, kde R je index rozhraní a A.B.C.D je IPv4 adresa
- konkrétní hodnoty mohou být např. *1.3.6.1.2.1.4.22.1.2.31.147.229.255.209*
- odstraněním řetězce odpovídajícího OID *ipNetToMediaPhysAddress* zůstane řetězec *31.147.229.255.209*
- vyjmutím první hodnoty 31 získáme index rozhraní a zbytek je IP adresa, kterou je možné rovnou použít pro aktualizaci databáze

4.3 Modul databáze

Informace získané pomocí SNMP mají strukturu vhodnou pro uložení těchto dat v relační SQL databázi. V unixových operačních systémech jsou dostupné databázové systémy MySQL a PosgreSQL. U obou těchto systémů je potřeba vyřešit otázku uložení hodnot IP adres a MAC adres.

Při využití základních datových typů připadají pro uložení IPv4 v úvahu tyto možnosti:

1) Jeden sloupec s datovým typem `VARCHAR(15)` je nejjednodušší řešení. Formát adresy není třeba upravovat, složitost struktury databáze se tím nezvýší a údaje budou i pro člověka snadno čitelné. Velkou nevýhodou je náročnost na prostor a neefektivní indexování. V systémech, které operují s velkým počtem takových údajů povede tento způsob uložení ke snížení výkonu oproti způsobům následujícím.
2) Čtyři sloupce s datovým typem `UNSIGNED TINYINT` efektivně využijí prostor a zachovají dobrou čitelnost údajů, nevýhodou je potřeba upravit data před uložením a čtením. Také struktura tabulek se tímto zkomplikuje.
3) Jeden sloupec s datovým typem `UNSIGNED INT` má nejlepší vlastnosti z hlediska využití prostoru a indexování dat, vyžaduje však složitější úpravy formátu při ukládání a čtení dat. Také čitelnost takto uložených dat je pro člověka obtížná. Pro systémy pracující s velkými objemy takových dat je tento typ nejvhodnější. Databáze MySQL nabízí za tímto účelem funkce `INET_ATON()` a `INET_NTOA()`, které slouží k převodu IP adresy zapsané jako řetězec na číslo a naopak.
4) Databáze PostgreSQL disponuje datovými typy `CIDR` a `INET`, které slouží k uložení IP adres. Jejich velikost je však sedm či devatenáct bajtů v PostgreSQL 8.4 [18], což oproti čtyřem bajtům datového typu `INT` znamená horší využití prostoru a efektivity operací. Výhodou těchto datových typů je současná podpora pro IPv4 i IPv6.

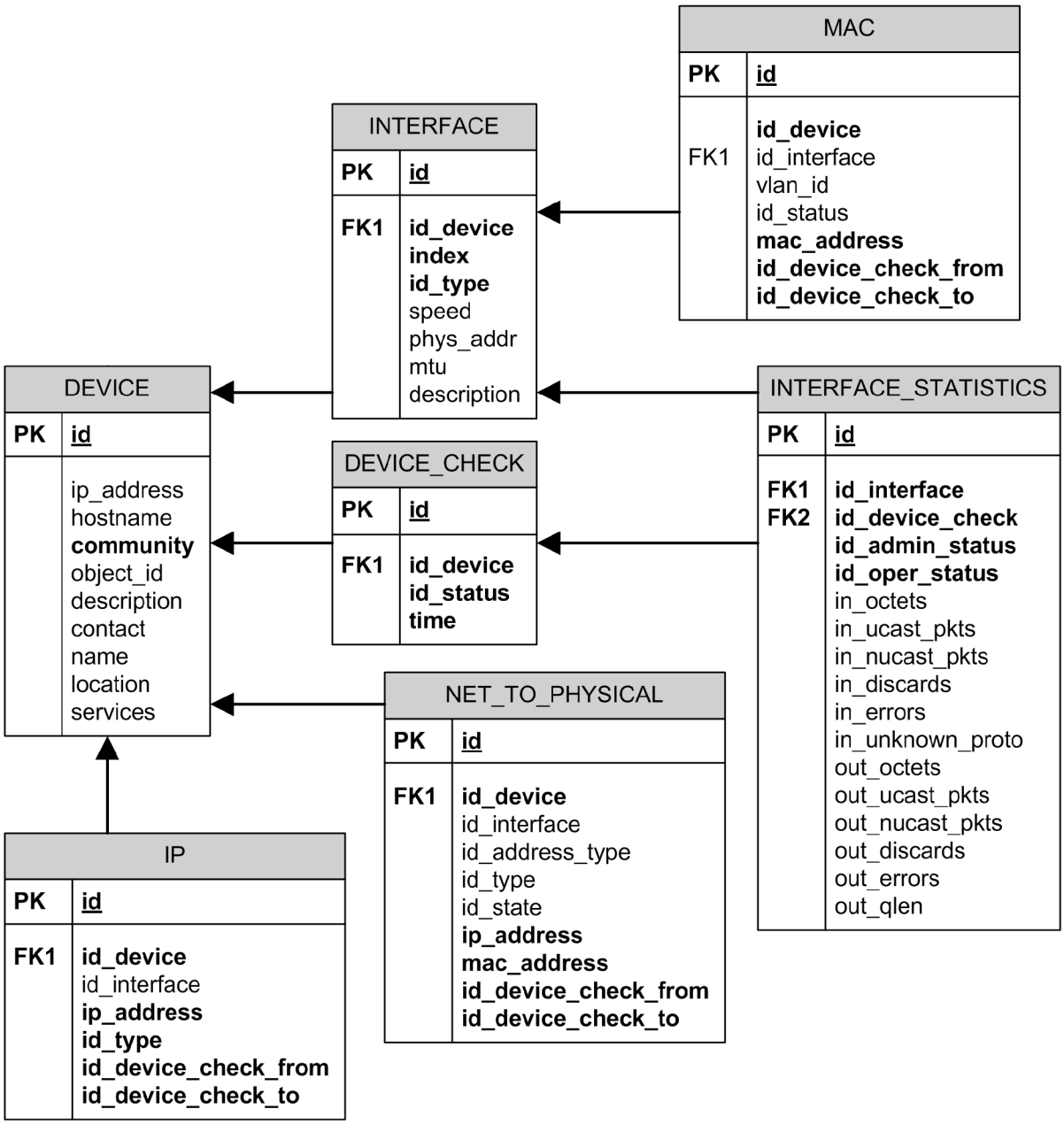
Složitější situace nastává v případě adresy IPv6, kterou nepojme ani největší celočíselný datový typ – osmibajtový `BIGINT`. V případě MySQL je nutné řešit ukládání jiným způsobem, např. využitím nějakého nečíselného datového typu, případně použít dva sloupce typu `BIGINT`. V PostgreSQL jsou k dispozici výše zmíněné datové typy `CIDR` a `INET`.

Ukládání MAC adresy je možné v MySQL vyřešit podobným způsobem jako u IPv4 (jen místo typu `INT` je nutné použít `BIGINT`), v PostgreSQL pro tento účel existuje datový typ `MACADDR`, o velikosti šest bajtů.

Rozhodl jsem se pro využití databáze PostgreSQL. Hlavním důvodem je podpora datových typů pro práci s IP a MAC adresami, dalším důvodem je BSD licence, pod kterou je PostgreSQL vydávána.

4.3.1 Relační schéma

Zobrazení relačního schématu databáze se nachází na obrázku 4.5. Tučně označené položky nesmějí obsahovat hodnotu `NULL`.



Obrázek 4.5: Relační schéma databáze

Nyní popíši podrobně jednotlivé tabulky. Databázová tabulka `DEVICE` obsahuje seznam monitorovaných zařízení, jejich popis, název, kontakt na správce a údaj o podporovaných síťových

vrstvách. Jednotlivé sloupce a datové typy popisuje tabulka 4.1. Povinnými údaji jsou buď IP adresa nebo hostname zařízení a hodnota komunity. Použité datové typy vycházejí buď z definice typů SNMP proměnných nebo v případě hostname z definice RFC 1035 [8].

Název	Datový typ	Vlastnosti	Popis
id	serial	primary key	Primární klíč.
ip_address	inet	null	IPv4 nebo IPv6 adresa zařízení.
hostname	varchar(255)	null	Hostname zařízení.
community	varchar(255)	not null	Hodnota SNMP komunity pro komunikaci se zařízením.
object_id	varchar(255)	null	Hodnota SNMP proměnné *sysObjectID*.
description	varchar(255)	null	Hodnota SNMP proměnné *sysDescr*.
contact	varchar(255)	null	Hodnota SNMP proměnné *sysContact*.
name	varchar(255)	null	Hodnota SNMP proměnné *sysName*.
location	varchar(255)	null	Hodnota SNMP proměnné *sysLocation*.
services	smallint	null	Hodnota SNMP proměnné *sysServices*.

Tabulka 4.1: Databázová tabulka DEVICE

Databázová tabulka `DEVICE_CHECK` zaznamenává informace o provedených SNMP dotazech na zařízeních, jejich výsledek a čas kontroly. Tato tabulka je důležitá pro generování statistik a reprezentování časových vztahů mezi IP a MAC adresami a zařízeními, u kterých se adresy vyskytovaly. Jednotlivé sloupce a datové typy popisuje tabulka 4.2. Pro volbu identifikátoru jako datového typu `SERIAL` (rozsah čtyři bajty, je však možné použít pouze kladné hodnoty, tedy jen 2^{31} hodnot [18]) jsem vycházel z následujícího předpokladu: při monitorování 100 zařízení v intervalu pět minut bude v tabulce přibývat každou hodinu 1 200 záznamů, za rok 10 512 000 záznamů – při rozsahu typu `SERIAL` by došlo k zaplnění tabulky za více než 200 let.

Název	Datový typ	Vlastnosti	Popis
id	serial	primary key	Primární klíč.
id_device	integer	not null	Reference do tabulky `DEVICE`.
id_status	smallint	not null	Status provedeného sběru dat.
time	timestamp	not null	Čas provedení sběru dat.

Tabulka 4.2: Databázová tabulka DEVICE_CHECK

Tabulka `INTERFACE` obsahuje seznam síťových rozhraní s referencí do tabulky `DEVICE`, informaci o typu rozhraní, jeho rychlosti, fyzické adrese a indexu. Jednotlivé sloupce a datové typy popisuje tabulka 4.3.

Název	Datový typ	Vlastnosti	Popis
id	serial	primary key	Primární klíč.
id_device	integer	not null	Reference do tabulky `DEVICE`.
index	smallint	not null	Hodnota SNMP proměnné *ifIndex*.
id_type	smallint	not null	Hodnota SNMP proměnné *ifType*.

speed	bigint	null	Hodnota SNMP proměnné *ifSpeed*.
phys_addr	macaddr	null	Hodnota SNMP proměnné *ifPhysAddress* reprezentující MAC adresu rozhraní.
mtu	smallint	null	Hodnota SNMP proměnné *ifMtu*.
description	varchar(255)	null	Hodnota SNMP proměnné *ifDescription*.

Tabulka 4.3: Databázová tabulka INTERFACE

Tabulka `INTERFACE_STATISTICS` zaznamená při každé úspěšné kontrole aktuální počty přenesených dat na rozhraních a stavy rozhraní, pro udržení časových informací obsahuje referenci do tabulky `DEVICE_CHECK`. Jednotlivé sloupce a datové typy popisuje tabulka 4.4. Použité datové typy vycházejí z definic typů SNMP proměnných. Komplikace nastává u SNMP typu `COUNTER`, který má od verze 2c rozsah 64 bitů a je neznaménkový. V PostgreSQL je pro tyto hodnoty možné použít datový typ `BIGINT`, který má také rozsah 64 bitů, je však pouze znaménkový. Před každým uložením a čtením takových hodnot z nebo do databáze je nutné provést jejich úpravu přičtením či odečtením maximální kladné hodnoty typu `BIGINT`. Pro datový typ `BIGSERIAL` (osm bajtů, možné využít 2^{63} hodnot) jako identifikátor záznamu jsem se rozhodl na základě stejného předpokladu, jako u tabulky `DEVICE_CHECK`, přibývání záznamů je nutné navíc vynásobit počtem rozhraní na zařízení. Pro počet rozhraní jsem zvolil hodnotu 100, potom k zaplnění tabulky dojde přibližně za dva roky při použití typu `SERIAL`, z toho důvodu je výhodnější využít typ `BIGSERIAL`.

Název	Datový typ	Vlastnosti	Popis
id	bigserial	primary key	Primární klíč.
id_interface	integer	not null	Reference do tabulky `INTERFACE`.
id_device_check	integer	not null	Reference do tabulky `DEVICE_CHECK`.
id_admin_status	smallint	not null	Hodnota SNMP proměnné *ifAdminStatus*.
id_oper_status	smallint	not null	Hodnota SNMP proměnné *ifOperStatus*.
in_octets	bigint	null	Hodnota SNMP proměnné *ifInOctets*.
in_ucast_pkts	bigint	null	Hodnota SNMP proměnné *ifInUcastPkts*.
in_nucast_pkts	bigint	null	Hodnota SNMP proměnné *ifInNUcastPkts*.
in_discards	bigint	null	Hodnota SNMP proměnné *ifInDiscards*.
in_errors	bigint	null	Hodnota SNMP proměnné *ifInErrors*.
in_unknown_protos	bigint	null	Hodnota SNMP proměnné *ifInUnknownProtos*.
out_octets	bigint	null	Hodnota SNMP proměnné *ifOutOctets*.
out_ucast_pkts	bigint	null	Hodnota SNMP proměnné *ifOutUcastPkts*.
out_nucast_pkts	bigint	null	Hodnota SNMP proměnné *ifOutNUcastPkts*.
out_discards	bigint	null	Hodnota SNMP proměnné *ifOutDiscards*.
out_qlen	integer	null	Hodnota SNMP proměnné *ifOutQLen*.

Tabulka 4.4: Databázová tabulka INTERFACE_STATISTICS

Tabulka `IP` udržuje jak aktuální konfiguraci IP adres tak i všechny předchozí konfigurace od počátku sledování zařízení. Jednotlivé sloupce a datové typy popisuje tabulka 4.5. Interval platnosti je

vyjádřen referencemi do tabulky DEVICE_CHECK. Adresy jsou získávány ze SNMP tabulky *ipAddressTable*. U některých zařízení není tabulka *ipAddressTable* podporována, v takovém případě jsou data získávána ze SNMP tabulky *ipAddrTable*.

Název	Datový typ	Vlastnosti	Popis
id	serial	primary key	Primární klíč.
id_device	integer	not null	Reference do tabulky DEVICE.
id_interface	integer	null	Reference do tabulky INTERFACE.
id_type	smallint	not null	Hodnota SNMP proměnné *ipAddressType*.
ip_address	inet	not null	Hodnota IP adresy získaná ze SNMP OID proměnné *ipAddressIfIndex*.
id_device_check_from	integer	not null	Reference do tabulky DEVICE_CHECK.
id_device_check_to	integer	not null	Reference do tabulky DEVICE_CHECK.

Tabulka 4.5: Databázová tabulka IP

Pomocí údajů z tabulky MAC je možné zjistit zařízení a rozhraní, na kterém se která MAC adresa vyskytovala v jakém časovém intervalu – časové závislosti jsou opět řešeny vazbou do tabulky DEVICE_CHECK. Jednotlivé sloupce a datové typy popisuje tabulka 4.6. Adresy jsou získávány ze SNMP tabulky *dot1qTpFdbTable*.

Název	Datový typ	Vlastnosti	Popis
id	bigserial	primary key	Primární klíč.
id_device	integer	not null	Reference do tabulky DEVICE.
id_interface	integer	null	Reference do tabulky INTERFACE.
vlan_id	smallint	null	Hodnota VLAN_ID získaná ze SNMP OID proměnné *dot1qTpFdbPort*.
id_status	smallint	null	Hodnota SNMP proměnné *dot1qTpFdbStatus*.
mac_address	macaddr	not null	Hodnota MAC adresy získaná ze SNMP OID proměnné *dot1qTpFdbStatus*.
id_device_check_from	integer	not null	Reference do tabulky DEVICE_CHECK.
id_device_check_to	integer	not null	Reference do tabulky DEVICE_CHECK.

Tabulka 4.6: Databázová tabulka MAC

Tabulka NET_TO_PHYSICAL obsahuje mapování IP adres na fyzické adresy. Zdrojem dat je SNMP tabulka *ipNetToPhysicalTable*, pokud tato není zařízením podporována, použije se *ipNetToMediaTable*. Intervaly platnosti záznamů jsou řešeny vazbou do tabulky DEVICE_CHECK. Jednotlivé sloupce a datové typy popisuje tabulka 4.7.

Název	Datový typ	Vlastnosti	Popis
id	bigserial	primary key	Primární klíč.
id_device	integer	not null	Reference do tabulky DEVICE.

id_interface	integer	null	Reference do tabulky `INTERFACE`.
id_address_type	smallint	null	Typ IP adresy získaný z hodnoty SNMP proměnné *ipNetToPhysicalNetAddressType*.
id_type	smallint	null	Způsob vzniku mapování získaný z hodnoty SNMP proměnné *ipNetToPhysicalType*.
id_state	smallint	null	Stav dostupnosti souseda získaný ze SNMP proměnné *ipNetToPhysicalState*.
ip_address	inet	not null	Hodnota IP adresy získaná ze SNMP OID proměnné *ipNetToPhysicalPhysAddress*.
mac_address	macaddr	not null	Hodnota MAC adresy získaná ze SNMP proměnné *ipNetToPhysicalPhysAddress*.
id_device_check_from	integer	not null	Reference do tabulky `DEVICE_CHECK`.
id_device_check_to	integer	not null	Reference do tabulky `DEVICE_CHECK`.

Tabulka 4.7: Databázová tabulka NET_TO_PHYSICAL

4.4 Modul uživatelského rozhraní

Data v systému mají různý charakter a podle typu informace se liší způsob, jakým je vhodné je reprezentovat uživateli. Statistiky o počtech přenesených dat je vhodné prezentovat v podobě grafu či tabulky, konfigurační údaje síťových zařízení budou snadno čitelné ve formě tabulek. Pro vyhledávání konkrétních záznamů musí existovat možnost zadat omezující podmínky – buď pomocí vhodného formuláře nebo dotazovacího jazyku. Jednou z možností pro splnění všech těchto potřeb je vytvoření uživatelského rozhraní formou webové aplikace. Webový server Apache v kombinaci se skriptovacím jazykem PHP umožňuje vytvořit přehledné rozhraní s podporou generování grafů, tabulek a dotazovacích formulářů. Také podpora SQL databáze, ve které jsou data uložena, je v případě použití PHP zajištěna. Další výhodou těchto technologií je jejich podpora v unixových systémech.

Ne vždy je ale použití webového rozhraní vhodné. Např. při vyhledávání informací není zadávání podmínek pomocí grafických prvků pro zkušeného uživatele efektivní, ne vždy musí mít člověk hledající určitou informaci po ruce počítač nebo přístup k internetu. Z těchto důvodů je vhodné nespoléhat se pouze na jediné uživatelské rozhraní v podobě webové aplikace, ale poskytnout uživateli alternativu v podobě dotazovacího nástroje, který bude schopen pracovat i v prostředí bez grafického uživatelského rozhraní. Další možností, která odstraňuje nutnost dostupného internetového připojení, je vytvoření aplikace pro některý z „chytrých“ mobilních telefonů s operačním systémem (Symbian, Windows Mobile, Android, a další).

V rámci této práce byly vytvořeny dva nástroje, první z nich je konzolová aplikace, která slouží pouze pro vyhledávání údajů, druhý nástroj je webová aplikace, která umožňuje kromě vyhledávání také spravovat monitorovaná zařízení. Oba nástroje a jejich použití bude popsáno v následujících kapitolách.

4.4.1 Konzolová aplikace

Tato aplikace se skládá ze dvou programů, jedním z nich je server, který očekává požadavky a zasílá odpovědi druhému programu – klientovi. Oba programy jsou psány v jazyku Perl, takže pro spuštění

vyžadují interpret Perlu s přítomnou knihovnou `IO::Socket::INET`, která obstarává síťovou komunikaci. Server navíc požaduje modul `DBI` s ovladačem `DBD::Pg`, protože se připojuje k databázovému serveru shromažďujícímu získané SNMP údaje.

Server

Může být spuštěn na libovolném stroji, který má konektivitu na databázový server, z hlediska výkonu je vhodné spustit ho přímo na databázovém serveru. Program se spouští bez parametrů, po spuštění zůstává aktivní a očekává požadavky od klientů na portu 23501 (hodnota byla vybrána ze seznamu neobsazených portů [19]). Jakmile přijde požadavek od klienta, který obsahuje buď IP nebo MAC adresu a počáteční a koncový čas platnosti záznamů, tak provede dotaz na databázový server se zadanými podmínkami a výsledek odešle zpět klientovi jako textový řetězec. Tento řetězec obsahuje hlavičku s popisem dat a samotná data, jako oddělovač řádků je použit znak roura („|“), za oddělovač sloupců slouží znak středník. Pokud nebyly nalezeny žádné vyhovující údaje tak je vrácena pouze hlavička.

Klient

Program je možné spustit s následujícími parametry:

- `-help` – vypíše nápovědu
- `-host` – adresa serveru, na který je zasílán dotaz, výchozí hodnota je localhost
- `-ip` – IP adresa, která se má vyhledat
- `-mac` – MAC adresa, která se má vyhledat
- `-from` – od jakého času se má hledat (nepovinný parametr)
- `-to` – do jakého času se má hledat (nepovinný parametr)
- `-f` – vypíše výsledek ve formátu tabulky (nepovinný parametr)

Musí být zadán minimálně jeden z parametrů `-ip` nebo `-mac`, jinak program neodešle žádný dotaz ale pouze vypíše nápovědu na standardní výstup. V opačném případě je zaslán požadavek na server, obsahující zadané parametry a získaný výsledek je vypsán na standardní výstup.

Ukázka výstupu programu se nachází na obrázku 4.6. Byl odeslán dotaz na vyhledání IP adresy 147.229.192.26 v časovém intervalu od 5. 5. 2010 00:00:00 až do 9. 5. 2010 23:59:59. Dotazu vyhovuje šest záznamů, význam jednotlivých sloupců je následující:

- `device_name` – jméno síťového zařízení, u kterého se hledaná adresa vyskytovala v tabulce mapování síťových adres na fyzické
- `port` – číslo portu, ke kterému se mapování vztahuje
- `int_desc` – popis rozhraní, ke kterému se mapování vztahuje
- `vlan_id` – VLAN, do které hledaná adresa náleží
- `mac_address` – fyzická adresa, která odpovídá hledané síťové adrese
- `ip_address` – hledaná síťová adresa
- `from` – počátek intervalu, během kterého se hledaná adresa vyskytovala v tabulce zařízení při každém sběru dat prováděném v průběhu tohoto intervalu
- `to` – konec intervalu, během kterého se hledaná adresa vyskytovala v tabulce zařízení při každém sběru dat prováděném v průběhu tohoto intervalu

```
xkukla05@pcuifs2:~/snmpterminal$
xkukla05@pcuifs2:~/snmpterminal$ ./client.pl -ip 147.229.192.6 -f -from 2010-05-05\ 00:00:00 -to 2010-05-09\ 23:59:59
device_name | port | int_desc | vlan_id | mac_address       | ip_address      | from                | to
=============================================================================================================
hp-pur      | 26   | B2       | 210     | 00:15:60:c4:1e:a8 | 147.229.192.6   | 2010-05-05 01:30:27 | 2010-05-05 01:45:01
hp-pur      | 26   | B2       | 210     | 00:15:60:c4:1e:a8 | 147.229.192.6   | 2010-05-05 19:34:40 | 2010-05-06 08:00:29
hp-pur      | 26   | B2       | 210     | 00:15:60:c4:1e:a8 | 147.229.192.6   | 2010-05-06 09:30:01 | 2010-05-06 20:32:39
hp-pur      | 26   | B2       | 210     | 00:15:60:c4:1e:a8 | 147.229.192.6   | 2010-05-06 20:48:04 | 2010-05-06 21:20:39
hp-pur      | 26   | B2       | 210     | 00:15:60:c4:1e:a8 | 147.229.192.6   | 2010-05-06 21:38:53 | 2010-05-07 00:45:02
hp-pur      | 26   | B2       | 210     | 00:15:60:c4:1e:a8 | 147.229.192.6   | 2010-05-07 08:32:38 | 2010-05-09 18:45:02
xkukla05@pcuifs2:~/snmpterminal$
```

Obrázek 4.6: Konzolová aplikace - klient

4.4.2 Webová aplikace

Aplikace ke svému běhu vyžaduje webový sever Apache a skriptovací jazyk PHP. Úkolem aplikace je umožnit vyhledávání v údajích získaných sběrným modulem, spravovat monitorovaná zařízení, spravovat uživatelské účty a umožnit čistou instalaci databázových tabulek na nové databázi. Práce s aplikací je umožněna pouze registrovaným uživatelům, definovány jsou dva typy uživatelských rolí – běžný uživatel a administrátor, který má oprávnění spravovat uživatelské účty a monitorovaná zařízení. Aplikace se skládá z několika částí, které budou popsány v následujících kapitolách.

Instalační skript

Pokud není při spuštění aplikace nalezena databázová tabulka uživatelských účtů, tak je uživateli nabídnut instalační skript. Ten vytvoří databázové tabulky, které sběrný modul vyžaduje pro svou funkčnost, založí tabulku uživatelů a vloží do ní prvního uživatele s oprávněním administrátora, uživatelským jménem „admin“ a heslem „admin“. Pokud v průběhu instalace nastanou nějaké chyby, je o nich uživatel informován, v opačném případě je vypsána zpráva o úspěšné instalaci a umožněno přihlášení do systému.

Správa zařízení

Tato část je přístupná pouze uživatelům s oprávněním administrátora. Vypíše seznam aktuálně monitorovaných zařízení, umožňuje mazat a přidávat zařízení nebo editovat jejich parametry. Při zakládání nebo editaci zařízení je možné vyplnit pouze jeho IP adresu, hostname a SNMP komunitu, zbývající údaje jsou automaticky doplněny na základě výsledku SNMP dotazů. Při mazání zařízení dojde také k odstranění všech záznamů z ostatních tabulek, které se na dané zařízení odkazují.

Správa uživatelů

Tato část je přístupná pouze uživatelům s oprávněním administrátora. Vypíše seznam existujících uživatelů, umožňuje mazat a přidávat uživatele nebo editovat jejich parametry.

Hledat podle IP

Slouží pro vyhledání údajů o konkrétní IPv4 nebo IPv6 adrese. V zobrazeném formuláři je nutné vyplnit v editačním poli „IP adresa“ hodnotu hledané adresy a v editačních polích „Od:“ a „Do:“ volitelně interval, ve kterém se má hledat. Nalezené výsledky jsou potom vypsány v podobě tabulky.

Ukázka získaných výsledků pro IP adresu 2001:718:802:c0d2::93e5:c003 v časovém intervalu od 4. 5. 2010 00:00:00 až do 9. 5. 2010 23:59:59 je zobrazena v tabulce 4.8. Význam jednotlivých sloupců je následující:

- Zařízení – jméno síťového zařízení, u kterého se hledaná adresa vyskytovala v tabulce mapování síťových adres na fyzické
- Port – číslo portu, ke kterému se mapování vztahuje
- Popis portu – popis portu, ke kterému se mapování vztahuje

- VLAN – VLAN, do které hledaná adresa náleží
- MAC – fyzická adresa, která odpovídá hledané síťové adrese
- IP – hodnota hledané síťové adresy
- Výskyt od – počátek intervalu, během kterého se hledaná adresa vyskytovala v tabulce zařízení při každém sběru dat prováděném v průběhu tohoto intervalu
- Výskyt do – konec intervalu, během kterého se hledaná adresa vyskytovala v tabulce zařízení při každém sběru dat prováděném v průběhu tohoto intervalu

Zařízení	Port	Popis portu	VLAN	MAC	IP	Výskyt od	Výskyt do
hp-kn66	1	1	210	00:50:56:93:1a:6a	2001:718:802:c0d2::93e5:c003	2010-05-04 16:00:01	2010-05-06 20:30:02
hp-kn66	1	1	210	00:50:56:93:1a:6a	2001:718:802:c0d2::93e5:c003	2010-05-06 20:47:03	2010-05-06 21:19:37
hp-kn66	1	1	210	00:50:56:93:1a:6a	2001:718:802:c0d2::93e5:c003	2010-05-06 21:37:54	2010-05-08 00:45:01
hp-kn66	1	1	210	00:50:56:93:1a:6a	2001:718:802:c0d2::93e5:c003	2010-05-08 01:00:01	2010-05-09 20:46:12

Tabulka 4.8: Výsledek hledání IPv6 adresy

Hledat podle MAC

Slouží pro vyhledání údajů o konkrétní MAC adrese. Funkčně je tento blok shodný s předchozím blokem. Ukázka získaných výsledků pro MAC adresu 00:50:56:93:1a:6a v časovém intervalu od 4. 5. 2010 až do 6. 5. 2010 23:59:59 je vypsána v tabulce 4.9. Význam sloupců je stejný jako v předchozím bloku. Z výsledků je vidět, že hledané síťové rozhraní bylo konfigurováno s IPv4 i s IPv6 adresou.

Zařízení	Port	Popis portu	VLAN	MAC	IP	Výskyt od	Výskyt do
hp-kn66	1	1	210	00:50:56:93:1a:6a	2001:718:802:c0d2::93e5:c003	2010-05-04 16:00:01	2010-05-06 20:30:02
hp-pur	292	Trk3	210	00:50:56:93:1a:6a	147.229.192.3	2010-05-04 16:02:19	2010-05-06 20:32:39
hp-kn66	1	1	210	00:50:56:93:1a:6a	2001:718:802:c0d2::93e5:c003	2010-05-06 20:47:03	2010-05-06 21:19:37
hp-pur	292	Trk3	210	00:50:56:93:1a:6a	147.229.192.3	2010-05-06 20:48:04	2010-05-06 21:20:39

Tabulka 4.9: Výsledek hledání MAC adresy

4.5 Zhodnocení zkušebního provozu

Tato kapitola popisuje chování navrženého systému na základě provedeného zkušebního provozu. Poskytuje informace o parametrech testovacího prostředí a popisuje chování jednotlivých částí systému (sběrného modulu, databáze, uživatelského rozhraní). V této kapitole jsou také popsány a řešeny problémy, které se během provozu objevily.

4.5.1 Parametry provozu a chování systému

Navržený systém byl spuštěn v testovacím provozu dne 30. 4. 2010 v čase 20:20. Monitorováno bylo celkem šest síťových prvků, konkrétně dvě zařízení HP ProCurve J8629A Switch 3500yl-24G, tři zařízení HP ProCurve J8697A Switch 5406zl a jedno zařízení 3Com switch 4800G 24-Port. Zpočátku byl sběr dat prováděn každých 10 minut, dne 4. 5. 2010 v 16:00 byl tento interval prodloužen na 15 minut, protože doba zpracování dat téměř dosahovala původního intervalu sběru dat. Po vytvoření vhodných databázových indexů byl tento problém odstraněn a celková doba získání, zpracování a uložení údajů se ustálila na době kratší než 1 minuta. Interval sběru byl ponechán na hodnotě 15 minut. Tabulka 4.10 zobrazuje počty záznamů v databázových tabulkách ke dni 10. 5. 2010 10:30, kdy již bylo provedeno 1 104 sběrů dat na každém zařízení (celkem tedy 6 624 sběrů dat).

Databázová tabulka	Počet záznamů
device	6
device_check	6 624
interface	435
interface_statistics	480 240
ip	62
mac	333 665
net_to_physical	78 212

Tabulka 4.10: Velikost databázových tabulek

Pro stanovení odhadu růstu databáze byl zjišťován nárůst databázových tabulek během 24-hodinových intervalů za období 7. 5. 2010 00:00 až 13. 5. 2010 23:59. Počet nových záznamů v databázových tabulkách během těchto intervalů zobrazuje tabulka 4.11. Zkoumány byly pouze tabulky `interface_statistics`, `mac` a `net_to_physical`, které během dosavadního provozu zaznamenaly největší zvyšování objemu. Výsledky u tabulky `interface_statistics` jsou podle očekávání téměř konstantní, její růst je závislý pouze na počtu monitorovaných zařízení a jejich rozhraní, který byl v průběhu celého provozu konstantní. Drobné rozdíly mezi jednotlivými dny jsou způsobeny občasnými výpadky nebo opakováním sběru dat mimo plánovaný čas, ke kterému docházelo z důvodu úprav v monitorovacím systému. Zajímavé výsledky lze pozorovat u tabulek `mac` a `net_to_physical`, kde v průběhu pracovního týdne je počet záznamů velmi podobný, v pátek začíná znatelně klesat, v sobotu je nejnižší a v neděli opět narůstá. To odpovídá sníženému počtu koncových zařízení v počítačové síti během doby volna. Z uvedených týdenních souhrnů lze odhadovat, že při současném provozu po dobu jednoho roku bude databázová tabulka `interface_statistics` obsahovat přes 15 milionů záznamů, tabulka `mac` téměř 12 milionů a tabulka `net_to_physical` okolo 3,8 milionů záznamů. U takto vysokých čísel je pravděpodobné, že bude potřeba podniknou určité kroky pro udržení efektivního chodu monitorovacího systému. Řešení může být například na úrovni databázového serveru, kde pravidelně spouštěná úloha bude snižovat počet záznamů v tabulkách spojováním intervalů platnosti záznamů

a promazáváním záznamů spadajících do těchto spojených intervalů. Toto však jistě není jediné řešení a otevírá se zde prostor pro návrhy dalších vylepšení.

Tabulka	Den	Počet nových záznamů	Týdenní souhrn
interface_statistics	Pá 7. 5. 2010	41 325	
interface_statistics	So 8. 5. 2010	41 760	
interface_statistics	Ne 9. 5. 2010	42 195	
interface_statistics	Po 10. 5. 2010	41 760	293 625
interface_statistics	Út 11. 5. 2010	41 760	
interface_statistics	St 12. 5. 2010	43 065	
interface_statistics	Čt 13. 5. 2010	41 760	
mac	Pá 7. 5. 2010	28 027	
mac	So 8. 5. 2010	22 122	
mac	Ne 9. 5. 2010	25 094	
mac	Po 10. 5. 2010	36 715	227 901
mac	Út 11. 5. 2010	40 247	
mac	St 12. 5. 2010	39 604	
mac	Čt 13. 5. 2010	36 092	
net_to_physical	Pá 7. 5. 2010	9 009	
net_to_physical	So 8. 5. 2010	6 207	
net_to_physical	Ne 9. 5. 2010	7 576	
net_to_physical	Po 10. 5. 2010	11 630	72 752
net_to_physical	Út 11. 5. 2010	13 283	
net_to_physical	St 12. 5. 2010	13 071	
net_to_physical	Čt 13. 5. 2010	11 976	

Tabulka 4.11: Růst databázových tabulek

4.5.2 Údaje získané během zkušebního provozu

Z údajů získaných během testovacího provozu bylo vytvořeno několik statistik. Tabulka 4.12 zobrazuje počet unikátních fyzických a síťových adres získaných z tabulek CAM a neigbour cache monitorovaných zařízení ke dni 10. 5. 2010. V tabulkách 4.8 a 4.9 jsou jsou vypsány údaje, které je možné zjistit ze systému při hledaní konkrétní adresy.

Typ adresy	Počet unikátních záznamů
IPv4	6 406
IPv6	3 952
MAC	9 958

Tabulka 4.12: Počty unikátních adres

Při vyhledávání konkrétních síťových a fyzických adres se pro určitá zadání objevily výsledky vyvolávající otázky. Např. fyzická adresa 00:1c:2e:d6:ea:00, které odpovídá síťová adresa 147.229.255.7, náleží podle výsledků získaných z databáze do několika různých VLAN. Výsledky takového dotazu zobrazuje tabulka 4.13.

Zařízení	Port	Popis portu	VLAN	MAC	IP	Výskyt od	Výskyt do
hp-man	23	23	504	00:1c:2e:d6:ea:00	147.229.255.7	06.05.10 21:37	10.05.10 17:45
hp-pur	117	E21	504	00:1c:2e:d6:ea:00	147.229.255.7	06.05.10 21:38	08.05.10 03:31
hp-pur	2	A2	532	00:1c:2e:d6:ea:00	147.229.255.7	06.05.10 21:38	10.05.10 17:30
hp-kol	124	F4	504	00:1c:2e:d6:ea:00	147.229.255.7	06.05.10 21:42	10.05.10 17:30
hp-kol	2	A2	532	00:1c:2e:d6:ea:00	147.229.255.7	06.05.10 21:42	10.05.10 17:30

Tabulka 4.13: Adresa vyskytující se v několika VLAN

Pátrání po příčině tohoto chování odhalilo, že chyba je ve způsobu reprezentace získaných dat. Pro získání hodnot uvedených v tabulce 4.13 je provedena operace `INNER JOIN` mezi databázovými tabulkami `MAC` a `NET_TO_PHYSICAL`, kde klíčem pro spojení je hodnota MAC adresy. Problém nastane, pokud má nějaké zařízení rozhraní, které náleží do více VLAN (pro danou fyzickou adresu existuje v tabulce `MAC` více záznamů) ale síťovou adresu má nakonfigurovanou pouze pro jedinou VLAN (v tabulce `NET_TO_PHYSICAL` existuje jeden záznam). Potom kvůli vlastnostem operace `INNER JOIN` dojde k přiřazení IP adresy i těm záznamům v tabulce `MAC`, které nemají odpovídající protějšek v tabulce `NET_TO_PHYSICAL`. Pro odstranění tohoto chování je potřeba rozšířit podmínku operace `INNER JOIN` tak, aby MAC adresám, které mají pro danou VLAN přiřazenou IP adresu tuto adresu přiřadila, ale k ostatním nenašla odpovídající záznam v tabulce s IP adresami. K tomu lze využít jednu specifickou vlastnost tabulek `MAC` a `NET_TO_PHYSICAL`. Jde o to, že údaje v tabulce `MAC` se vážou ke konkrétnímu fyzickému rozhraní, kdežto údaje v tabulce `NET_TO_PHYSICAL` se vážou k virtuálnímu rozhraní určité VLAN. Z toho lze poznat, které MAC adresy nemají pro danou VLAN nakonfigurovanou IP adresu – pro tuto VLAN neexistuje záznam v tabulce `NET_TO_PHYSICAL`. Operaci `INNER JOIN` tedy stačí doplnit o podmínku rovnosti VLAN_ID mezi oběma tabulkami.

Pro realizaci této operace byla tabulka `INTERFACE` rozšířena o sloupec `VLAN_ID`, jehož hodnotu lze u některých zařízení získat ze SNMP proměnné *ifDescr*, u jiných je nutné ji získat ze SNMP tabulky *dot1qVlanStaticName*, do které je klíčem hodnota proměnné *ifDescr*. Výsledky pro adresu 00:1c:2e:d6:ea:00 po provedení těchto úprav zobrazuje tabulka 4.14.

Zařízení	Port	Popis portu	VLAN	MAC	IP	Výskyt od	Výskyt do
hp-list	99	E3	504	00:1c:2e:d6:ea:00		06.05.10 21:37	10.05.10 17:45
hp-man	23	23	504	00:1c:2e:d6:ea:00	147.229.255.7	06.05.10 21:37	10.05.10 17:45
hp-list	99	E3	532	00:1c:2e:d6:ea:00		06.05.10 21:37	10.05.10 17:45
hp-pur	117	E21	504	00:1c:2e:d6:ea:00	147.229.255.7	06.05.10 21:38	08.05.10 03:31
hp-pur	2	A2	532	00:1c:2e:d6:ea:00		06.05.10 21:38	10.05.10 17:45
hp-kol	124	F4	504	00:1c:2e:d6:ea:00	147.229.255.7	06.05.10 21:42	10.05.10 17:45
hp-kol	2	A2	532	00:1c:2e:d6:ea:00		06.05.10 21:42	10.05.10 17:45

Tabulka 4.14: Vyhledání MAC adresy po opravě reprezentace údajů

5 Závěr

V průběhu práce na tomto diplomovém projektu jsem získal přehled o možnostech monitorování počítačových sítí, především hlubší teoretické znalosti o protokolu SNMP. Rozboru protokolu SNMP jsem věnoval velkou část této práce, protože ho považuji za vhodný prostředek při tvorbě NMS splňujícího požadavky na monitorování sítě VUT v Brně. Jeho praktické použití jsem úspěšně otestoval na jednoduchých příkladech. Získané zkušenosti mi společně se znalostí architektury protokolu umožnily navrhnout vlastní systém pro sběr provozních údajů z aktivních síťových prvků a vytvořit databázovou strukturu vhodnou pro uchování a reprezentaci dat, která vypovídají o stavech počítačové sítě v různých okamžicích.

Přínos vytvořeného řešení vidím v poskytnutí nového pohledu do fungování počítačové sítě z hlediska druhé a třetí síťové vrstvy. Shromáždění informací o stavu tabulek CAM a neighbour cache ze síťových prvků v centrální databázi a zaznamenání časové platnosti těchto údajů umožňuje efektivně vyhledávat výskyt určitých informací o síťových nebo fyzických adresách, ať už se jedná o vyhledání konkrétní adresy nebo celkovou statistiku přidělených síťových adres v určitém období. Dále mohou získané údaje pomoci odhalit některé nesrovnalosti v konfiguraci počítačové sítě. Za úspěch považuji také vyřešení problematiky týkající se podpory IPv6, vytvořený systém je schopný bez dalších úprav nebo dodatečné konfigurace fungovat v počítačové síti využívající jak IPv4 tak IPv6 protokol.

Při vytváření a testování systému se objevily další otázky a náměty k řešení. U testovaných zařízení byla objevena určitá nekompatibilita ohledně podpory SNMP objektů, která vyžadovala odlišný přístup k zasílání SNMP dotazů. Tento problém se podařilo v rámci testovaných zařízení vyřešit, testování však probíhalo na malém počtu zařízení různých značek a typů. V případě potřeby monitorovat další modely zařízení nebo zařízení s odlišnými verzemi softwaru bude nutné otestovat jejich funkčnost se sběrným modulem a případné další výjimky ve skriptu ošetřit.

Dalším problémem k řešení může být chování systému po delší době aktivní činnosti, zde se dá očekávat, že po dosažení určité velikosti databáze nastanou problémy, které znemožní efektivní fungování monitorovacího systému.

V případě dalšího využívání systému se zde nabízí možnost jeho zapojení do jiného, komplexního NMS, kterou by bylo možné provést vytvořením vhodného rozšiřujícího modulu.

Literatura

[1] *Internet protocol* [online]. 1981 [cit. 29. 11. 2009]. Dostupný z WWW: <http://www.ietf.org/rfc/rfc791.txt>

[2] Fraleigh, Ch., Moon, S., Lyles, B., Cotton, Ch., Khan, M., Moll, D., Rockell, R., Seely, T., Diot, Ch: *Packet-Level Traffic Measurements from the Sprint IP Backbone*, IEEE Network Magazine, 2003

[3] Mauro, D., R., Schmidt, K., J.: *Essential SNMP*, Cambridge, O'Reilly & Associates, 2001, ISBN 0-596-00020-0

[4] Held, G.: *LAN Management with SNMP and RMON*, New York, Wiley Computer Publishing, 1996, ISBN 0-471-14736-2

[5] Kretchmar, J., M., Dostálek, L.: *Administrace a diagnostika sítí pomocí OpenSource utilit a nástrojů*, Brno, Computer Press, 2004, ISBN 80-251-0345-5

[6] Stallings, W.: *SNMP, SNMPv2, SNMPv3, and RMON 1 and 2, Third Edition*, USA, Addison-Wessley, 1999, ISBN 0-201-48534-6

[7] *Cisco SNMP Object Navigator* [online]. 2010 [cit. 25. 3. 2010]. Dostupný z WWW: <http://tools.cisco.com/Support/SNMP/do/BrowseOID.do>

[8] *Request for Comments (RFC)* [online]. 2010 [cit. 18. 5. 2010]. Dostupný z WWW: <http://www.ietf.org/rfc.html>

[9] *Information technology - Abstract Syntax Notation One (ASN.1): Specification of basic notation* [online]. 2002 [cit. 18. 5. 2010]. Dostupný z WWW: <http://www.itu.int/ITU-T/studygroups/com17/languages/X.680-0207.pdf>

[10] *Cacti* [online]. 2009 [cit. 5. 1 .2010]. Dostupný z WWW: <http://www.cacti.net>

[11] *Nagios* [online]. 2009 [cit. 5. 1. 2010]. Dostupný z WWW: <http://www.nagios.org>

[12] *NAV features at a glance* [online]. 2007 [cit. 5. 1. 2010]. Dostupný z WWW: <https://metanav.uninett.no/navfeatures>

[13] *ZABBIX* [online]. 2009 [cit. 5. 1. 2010]. Dostupný z WWW: <http://www.zabbix.com>

[14] *Net::SNMP - Object oriented interface to SNMP* [online]. 2009 [cit. 7. 5. 2010]. Dostupný z WWW: <http://search.cpan.org/~dtown/Net-SNMP-v6.0.0/lib/Net/SNMP.pm>

[15] *DBI - Database independent interface for Perl* [online]. 2010 [cit. 8. 5. 2010]. Dostupný z WWW: <http://search.cpan.org/~timb/DBI-1.611/DBI.pm>

[16] *DBD::Pg - PostgreSQL database driver for the DBI module* [online]. 2010 [cit. 8. 5. 2010]. Dostupný z WWW: <http://search.cpan.org/~turnstep/DBD-Pg-2.17.1/Pg.pm>

[17] *Parallel::ForkManager - A simple parallel processing fork manager* [online]. 2000 [cit. 8. 5. 2010]. Dostupný z WWW: <http://search.cpan.org/~dlux/Parallel-ForkManager-0.7.5/ForkManager.pm>

[18] *PostgreSQL 8.4.3 Documentation* [online]. 2009 [cit. 23. 3. 2010]. Dostupný z WWW: <http://www.postgresql.org/docs/8.4/static/index.html>

[19] *Port numbers* [online]. 2010 [cit. 18. 5. 2010]. Dostupný z WWW: <http://www.iana.org/assignments/port-numbers>

Seznam příloh

Příloha A: Seznam zkratek

ARP	Address Resolution Protocol
ASN.1	Abstract Syntax Notation One
CAM	Content Addressable Memory
CCITT	Comite Consultatif Internationale de Telegraphie et Telephonie
EGP	Exterior Gateway Protocol
IAB	Internet Architecture Board
ICMP	Internet Control Message Protocol
IETF	Internet Engineering Task Force
IOS	Internetwork Operating systém
IP	Internet Protocol
ISO	International Organization for Standardization
ITU	International Telecommunications Union
MIB	Management Information Base
NMS	Network Management Station
OID	Object Identifier
PDU	Protocol Data Unit
SMI	Structure Management Information
SNMP	Simple Network Management Protocol
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
USM	User-based Security Model
VASM	View-based Access Control Model

Příloha B: Ukázka SNMP dotazu v Perlu

Ukázkový program v jazyku Perl získá ze serveru `isa.fit.vutbr.cz` hodnoty SNMP proměnných pro popis zařízení, unikátní ID, dobu běhu systému a podporované síťové vrstvy.

```
#! /usr/local/bin/perl
use strict;
use warnings;
use Net::SNMP;

my $hostname = 'isa.fit.vutbr.cz';      # adresa zařízení
my $community = 'public';               # komunita pro čtení

# OID dotazovaných proměnných
my $OID_sysDescr =      '1.3.6.1.2.1.1.1';
my $OID_sysObjectId =   '1.3.6.1.2.1.1.2';
my $OID_sysUpTime =     '1.3.6.1.2.1.1.3';
my $OID_sysServices =   '1.3.6.1.2.1.1.7';

# vytvoření session
my($session, $error) = Net::SNMP->session(
    -hostname    => $hostname,
    -community   => $community,
    -nonblocking => 1,
    -version     => 'snmpv2c',
);

if (!defined $session) {
    printf ("ERROR: %s.\n", $error);
    exit 1;
}

# volání SNMP funkce get_bulk_request, výsledek se předá funkci
# get_bulk_callback
my $result = $session->get_bulk_request(
    -varbindlist    => [   $OID_sysDescr,
                           $OID_sysObjectId,
                           $OID_sysUpTime,
                           $OID_sysServices ],
    -callback       => [ \&get_bulk_callback ],
    -nonrepeaters   => 4,
    -maxrepetitions => 0,
);

if (!defined $result) {
    printf ("ERROR: %s\n", $session->error());
    $session->close();
    exit 1;
}

snmp_dispatcher();  # neblokující smyčka
```

```
$session->close();

exit 0;

# funkce volaná při předání výsledku
sub get_bulk_callback
{
    my ($session) = @_;

    # reference na hash s výsledky
    my $result = $session->var_bind_list();

    if (!defined $result) {
        printf ("ERROR: %s\n", $session->error());
        return;
    }

    # seznam vrácených OID
    my @oids = $session->var_bind_names();

    # výpis vrácených výsledku
    foreach my $oid (@oids) {
        printf ("%s = %s\n", $oid, $result->{$oid});
    }

    return;
}
```

Po úspěšném provedení vypíše program tyto hodnoty:

```
1.3.6.1.2.1.1.1.0 = FreeBSD isa.fit.vutbr.cz 7.2-STABLE FreeBSD 7.2-
STABLE    #0:    Wed    Sep         9    14:34:48    CEST    2009
root@isa.fit.vutbr.cz:/usr/obj/usr/src/sys/ISA i386
1.3.6.1.2.1.1.2.0 = 1.3.6.1.4.1.8072.3.2.8
1.3.6.1.2.1.1.3.0 = 16 days, 08:26:35.01
1.3.6.1.2.1.1.7.0 = 72
```

Z uvedených výsledků můžeme zjistit, že na dotazovaném systému je spuštěn operační systém FreeBSD, SNMP agent má identifikaci „1.3.6.1.4.1.8072.3.2.8“, od poslední inicializace systému uplynulo více než 16 dní a že zařízení poskytuje aplikační služby.