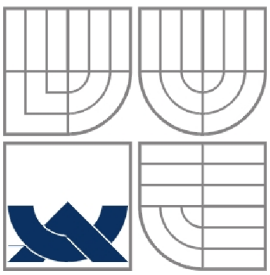
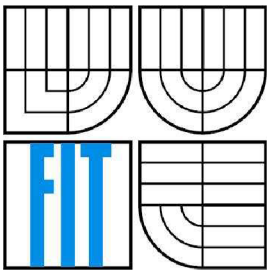


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

SYSTÉM PRO SLEDOVÁNÍ STAVU SPOJENÍ V POČÍTAČOVÉ SÍTI POMOCÍ SNMP

A SYSTEM FOR MONITORING STATUS OF NETWORK LINKS VIA SNMP

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

Peter Zubčák

VEDOUCÍ PRÁCE
SUPERVISOR

Mgr. Marek Rychlý

BRNO 2009

Abstrakt

Cílem bakalářské práce je implementovat systém pre sledování stavu spojení v počítačové síti. Systém sbíra informace ze síťových uzlů, z kterých počíta metriky a statistiky, které sú dostupné pre uživatelov v MIB. Komunikace se síťovými uzly probíha prostřednictvím protokolu SNMP. Systém je implementován v jazyce C++ jako multiplatformní služba.

Abstract

The goal of the bachelor thesis is an implementation of the system for monitoring status of network links. System collects information from network nodes, from which calculates metrics and statistics, which are available for users in MIB. The communications with network nodes is proceeded by SNMP protocol. The system is implemented in C++ language as a multiplatform service.

Klíčová slova

jednoduchý protokol pre správu síte, manager, agent, MIB, SMI, jmenný strom, síťová metrika, dostupnost, ztráta, zpoždění, šírka pásma, statistiky, agregace v čase

Keywords

simple network management protocol, manažer, agent, MIB, SMI, naming tree, network metric, availability, loss, delay, bandwidth, statistics, aggregation in time

Citace

Zubčák Peter: Systém pro sledování stavu spojení v počítačové síti pomocí SNMP, bakalářská práce, Brno, FIT VUT v Brně, 2009.

Systém pro sledování stavu spojení v počítačové síti pomocí SNMP

Prohlášení

Čestne prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením Mgr. Marka Rychlého.
Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

.....
Peter Zubčák
20.5. 2009

Poděkování

Poďakovanie za možnosť práce na tomto projekte patrí predovšetkým svojmu odbornému konzultantovi Mgr. Markovi Rychlému, ktorému patrí aj vďaka za cenné rady a pripomienky. Srdečná vďaka patrí aj mojej rodine, ktorá ma podporovala.

Obsah

1 Úvod

Za posledných pár rokov sa počet počítačov prepojených v sieti radikálne zvýšil. V dnešných komplexných počítačových sieťach zložených zo smerovačov, prepínačov, serverov sa zdá byť nemožné spravovať všetky zariadenia na sieti a zabezpečiť ich optimálny beh.

Kým sa sieťové technológie nepoužívali tak široko, správa siete nemala veľké uplatnenie, keďže nebolo čo spravovať. Ak sa objavilo nejaké zlyhanie šlo väčšinou o hardwarovú chybu, avšak dnes je hardware omnoho spoľahlivejší a chyby vznikajú dôsledkom softwarových chýb alebo nepriateľskými užívateľmi. Pre mnohých ľudí slúži sieť ako pracovný nástroj, na niektorých sieťach závisia ľudské životy a preto je nevyhnutné zabezpečiť spoľahlivosť siete. Nepredvídateľnosť a spoľahlivosť vedú k vyšším nárokom na správu siete.

1.1 Motivácia

„*V sieti Massachusetts Institute of Technology (MIT) je pripojených viac ako 30 000 počítačov, každý je pritom zapojený do portu s podporou sieťovej správy. Napriek súčasnému pokroku platí, že jeden nevhodne sa chovajúci systém môže spôsobiť závažný problém. Schopnosť dostatočne rýchlo nájsť a prípadne odpojiť problémové zariadenie je kritická*“ [2].

Uvažujme spoločnosť, v ktorej máme niekoľko stoviek počítačov prepojených pomocou smerovačov a prepínačov, ktoré udržiavajú sieť v chode. Niekoľko počítačov slúži ako súborové serveri, na iných beží software, ktorý overuje transakcie kreditných kariet a zvyšok sú pracovné stanice. T1 okruh spája spoločnosť s internetom a nachádza sa tu privátne spojenie na systém overovania kreditných kariet. Čo sa stane ak sa zrúti jeden zo súborových serverov? Ak sa to stane v strede pracovného dňa je pravdepodobné, že si to ľudia všimnú a zavolajú administrátora, ktorý vykoná nápravu. Ale čo ak sa to stane potom ako zamestnanci odišli, vrátane administrátora alebo cez víkend? Čo ak sa zrúti spojenie so systémom pre overovanie kreditných kariet v piatok večer a nie je napravené do pondelka? Ak bolo zlyhanie spôsobené chybou na smerovači a náprava mohla byť vykonaná jeho výmenou, množstvo peňazí z internetového predaja by bolo stratených bezvýznamne. Zlyhanie T1 spojenia s internetom môže ovplyvniť množstvo predaja generovaného prístupom jednotlivcov na stránky spoločnosti a zadávaním príkazov [1].

Dokázať odhaliť vzniknutý problém v počítačovej sieti a včas naň reagovať je v dnešnej dobe veľmi dôležité. SNMP umožňuje nepretržite monitorovať sieť bez našej prítomnosti a prípadne nás informovať o vznikajúcich problémoch, na ktoré môžeme zareagovať skôr ako dôjde k závažným následkom.

1.2 Ciele projektu

Primárnym cieľom projektu je návrh a zhotovenie systému, ktorý bude monitorovať pomocou protokolu SNMP sieťové uzly, získavať z nich v pravidelných intervaloch atribúty, z ktorých následne spočíta metriky, ktorými dokážeme ohodnotiť stav uzlu alebo linky.

V projekte by som chcel predstaviť sieťové výkonnostné metriky, ktoré má zmysel získavať a definovať operácie pre ich následné spracovanie, ktoré je najužitočnejšie z pohľadu užívateľa.

Nato aby som mohol splniť tento cieľ je potrebné aby som sa oboznámil s oblasťou sieťovej správy, ako aj s existujúcimi softwarovými produktmi, ktoré sa vo svete používajú.

1.3 Prehľad dokumentu

Práca je členená na kapitoly, ktoré odzrkadľujú postup pri vytváraní tohto dokumentu. Tieto kapitoly sú ďalej členené na podkapitoly tak, aby bola dodržaná logickosť textu a úzko súvisiace pojmy boli spolu v jednej kapitole.

V prvej kapitole sa nachádza stručný úvod do problematiky, je vysvetlené prečo je potrebné vytvárať systémy, ktoré sa zaoberajú sieťovou správou, sú načrtnuté ciele projektu a popísaný prehľad.

Druhá kapitola sa zaoberá vysvetlením základných pojmov týkajúcich sa sieťovej správy, ako je protokol SNMP, MIB, SMI. Sú tu predstavené sieťové výkonnostné metriky, ktoré sa v praxi používajú a naznačené ich spracovanie.

Tretia kapitola stručne popisuje požiadavky na vytváraný systém a voľbu vhodných programových rozhraní na jeho implementáciu.

Štvrtá kapitola je venovaná implementácií jednotlivých metrík, popisuje sa tu aj ich umiestnenie v MIB databáze, testovanie a možné rozšírenie.

V záverečnej kapitole je zhrnutie a zhodnotenie celej práce.

2 Analýza problémovej oblasti

V zadaní bakalárskej práce je vyslovená požiadavka na implementáciu nástrojov, ktoré by pomocou protokolu SNMP získavali z navzájom prepojených sieťových uzlov atribúty, z ktorých by sme dokázali odvodiť isté charakteristiky (napr. strata paketov na p2p spoji).

V nasledujúcich kapitolách si objasníme základné pojmy, ktoré spadajú do oblasti sieťovej správy, aby aj čitateľ získal potrebné úvodné znalosti, ujasnil si pojmy používané v texte a mohol ďalej pokračovať v štúdiu práce.

2.1 SNMP

V nasledujúcej časti by som chcel vysvetliť čo to vlastne SNMP je, načo slúži a z akých častí sa skladá.

SNMP (Simple network management protocol), jedná sa o jednoduchý protokol, ktorý slúži ku správe zariadení v sieti.

SNMP sa snaží splniť nasledujúce ciele:

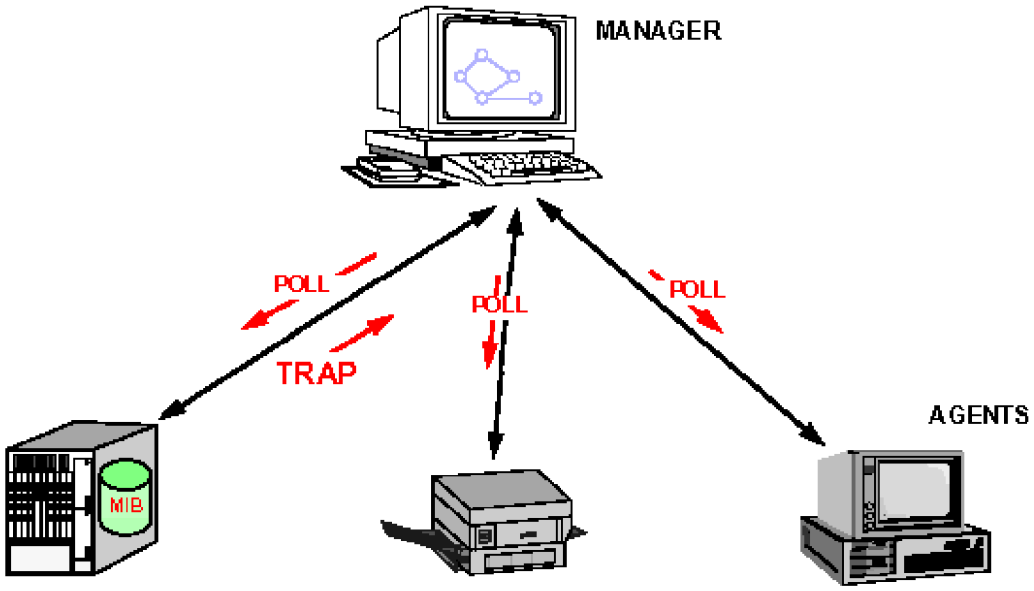
- **všadeprítomnosť** - SNMP by malo bežať na každom systéme (server, pracovná stanica, switch, router, atd.). Každé zariadenie, ktoré ovláda protokol IP. Internetový štandard robí zo SNMP nástroj, ktorý má výhodu v prostredí kde je nasadených viacero zariadení od rôznych výrobcov. Každé zariadenie môže vo vnútri pracovať inak, ale je dostupné operátorovi cez spoločné rozhranie.
- **jednoduché pridávanie SNMP funkcionality** - Nato aby sme dokázali spravovať systém (agent), je potreba pridať malé množstvo kódu, čím získame obmedzenú funkčnosť. Manažér môže mať komplexný kód. Spravovaný systém nie je schopný sám kontrolovať či funguje správne, ak tak obmedzene.
- **jednoduché rozšírenie správy** - Informácie, ktoré spravujeme majú modulárnu štruktúru a preto je jednoduché ich rozšíriť pridaním nového modulu.
- **robustná správa** - Aj keď je sieť zaťažená, správa siete by mala stále fungovať.

Vo svete SNMP existujú dva druhy entít: manažéri a agenti (vid obr. 2.1 prevzatý z [7]).

- **Manažér** je server, na ktorom beží určitý typ softwaru, ktorý dokáže spravovať úlohy týkajúce sa správy siete, často bývajú označovaný pojmom NMS (Network Management Station). NMS je zodpovedný za polling (dotazovanie) a prijímanie trapov (pascí) od agentov v sieti. Poll (dotaz) je v kontexte sieťovej správy, akt dotazovania sa agenta (router, switch, Unix server, atd.) o nejaký kúsok informácie.

Táto informácia môže byť neskôr použitá na predikciu katastrofických udalostí. Trap je spôsob ako môže agent oznámiť manažérovi, že došlo k nejakej udalosti. Pasce sú posielané asynchrónne, nie ako odpoveď na dotaz od manažéra [2].

- **Agent** je software, ktorý beží na sieťových zariadeniach, ktoré spravujeme. Môže to byť daemon (služba), alebo môže byť vstavaný do operačného systému (Cisco IOS). Agent sleduje rôzne operačné stavy zariadenia, na ktorom sa nachádza a poskytuje ich manažérovi. Napríklad agent umiestnený na routri môže sledovať aktuálne prenosy na jednotlivých rozhraniach. Manažér je schopný dotazovať sa routra na prichádzajúce množstvá dát konkrétneho rozhrania a vykonať nápravu ak sa ich počet neúmerne zvyšuje. Ak dôjde k zmene stavu nejakého rozhrania môže agent poslať manažérovi pascu [2].



Obr. 2.1: Princíp fungovania manažérov a agentov

Samotné SNMP je tvorené niekoľkými štandardami:

1. SNMP protokol - definuje ako sa informácie prenášajú
2. SMI - definuje syntax
3. MIB - definuje aká informácia pre správu existuje

2.1.1 SNMP protokol

V súčasnosti existujú 3 verzie tohto protokolu:

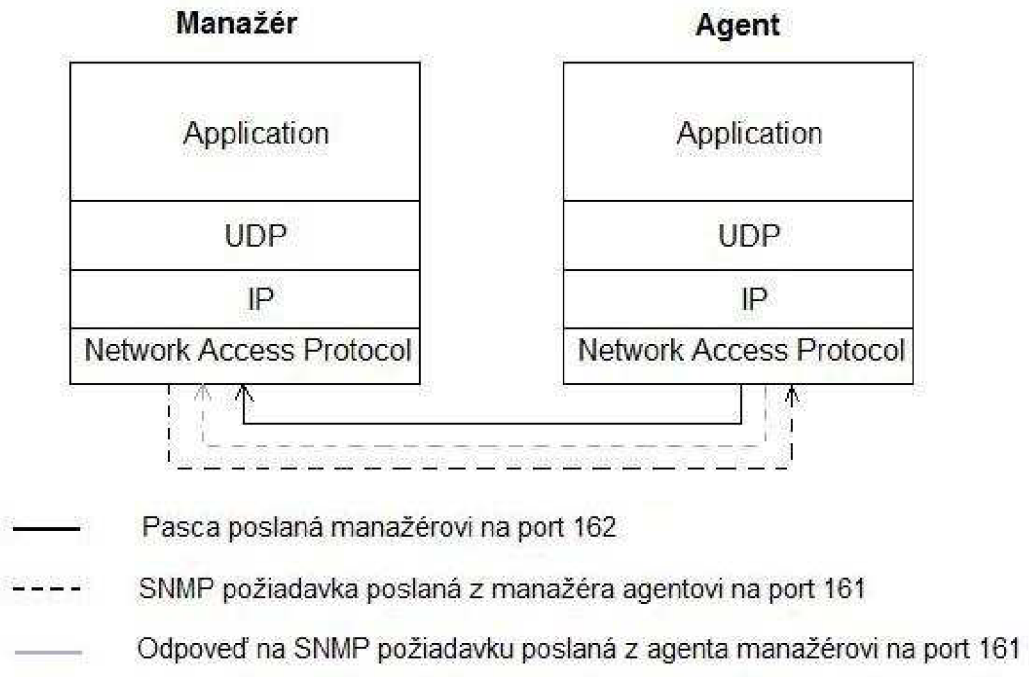
- SNMPv1

- SNMPv2
- SNMPv3

SNMP sa prenáša protokolom UDP (User Dátagram protokol). Táto voľba bola zvolená kvôli menšej réžii oproti TCP, v prípade zahltenia siete má niekoľko po sebe poslaných UDP Dátagramov väčšiu pravdepodobnosť doraziť do cieľa. Keďže UDP je nespojový, nespoľahlivý protokol, Dátagramy sa môžu po ceste stratiť a žiadne potvrdzovanie na úrovni protokolu neexistuje. Úlohou SNMP aplikácií je zabezpečiť detekciu straty. Aplikácie väčšinou definujú časový limit, do ktorého vypršania musí doraziť odpoveď, ak nedorazí tak sa pošle požiadavka znovu. Pri pascách je situácia horšia, keď agent pošle pascu a tá nedorazí, manažér nemôže vedieť, že bola vôbec poslaná [1].

Každý SNMP paket obsahuje číslo verzie, komunitu, čo je textový reťazec, ktorý sa používa ako primitívna metóda autorizácie, je možné ho nastaviť iba pre čítanie alebo aj pre zápis. V protokole SNMPv1 a SNMPv2 sa komunity prenášajú v čitateľnej podobe, takže ak ich zachytí útočník je schopný ich zneužiť. Vo verzií snmpv3 sa zaviedol bezpečný spôsob autorizácie a prenosu dát. Tretím údajom v hlavičke SNMP je typ odosielaného požiadavku a označuje sa ako PDU (Protocol Dáta Unit). Môže mať nasledúce hodnoty:

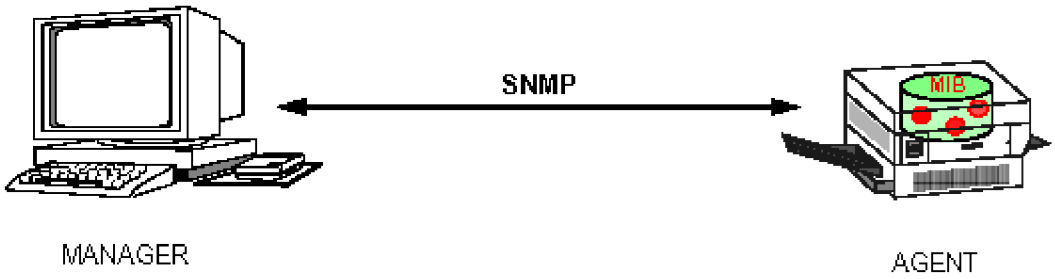
- get-request - zistenie hodnoty premennej
- get-next-request - zistenie nasledujúcej premennej a jej hodnoty
- get-response - odpoveď od zariadenia
- set-request - nastavenie hodnoty premennej
- trap - informácia zaslaná zariadením



Obr. 2.2: TCP/IP komunikačný model a SNMP (prevzaté z [1])

2.1.2 MIB (Management Information base)

Obsahujú spravované objekty (premenné), ktoré reprezentujú zdroje systému. Tieto zdroje môžu byť monitorované alebo zmenené vzdialeným manažérom za účelom kontroly správania systému. MIB môžeme chápať ako istý typ Dátabázy, ktorá je umiestnená vo vzdialenom systéme a manažérovi je dostupná pomocou protokolu SNMP (vid obr. 2.3 prevzatý z [8]) [3].



Obr. 2.3: Ukážka MIB vo vnútri spravovaného systému

Pri MIB rozlišujeme 2 veci:

- **MIB definícia** je predpis toho ako má MIB vyzerať, aké premenné má obsahovať. Definícia by mala byť známa tvorcovi MIB, kvôli implementácií premenných, ale tak isto aj manažérovi aby vedel aké objekty má dostupné k zmenám a k čítaniu.
- **MIB inštancia** predstavuje skutočné premenné, ktoré obsahujú hodnoty. Je dostupná vo spravovanom systéme, nie v riadiacom. Ak dochádza k zmenám premenných malo by to byť vždy na strane agenta. Manažér si môže spraviť lokálnu kópiu premenných, ale musí počítať s tým, že kópia môže byť zastaralá.

Jednou z dôležitých vlastností MIB je jej **modularita**. Je zlým zvykom definovať všetky objekty spravovaného systému v jedenej MIB definícií. Je lepšie definovať niekoľko MIB modulov. Výhoda modulárnej štruktúry spočíva vtom, že rôzny ľudia môžu vytvárať rôzne časti MIB podľa ich skúseností. Rôzne druhy systémov môžu podporovať rôzne typy MIB modulov. Napríklad modul pre správu e-mailového servera nemá veľký zmysel pre bridge (most) [3].

2.1.3 SMI (Structure of Management Information)

V tejto časti by som chcel stručne charakterizovať načo slúži SMI a uviesť pár príkladov. Základný zmysel SMI je uľahčiť definíciu nových MIB. Definuje syntax premenných a umožňuje vytvorenie nástrojov, ktoré dokážu pracovať s MIB jednoduchým spôsobom.

Informácia o spravovanom systéme môže byť reprezentovaná:

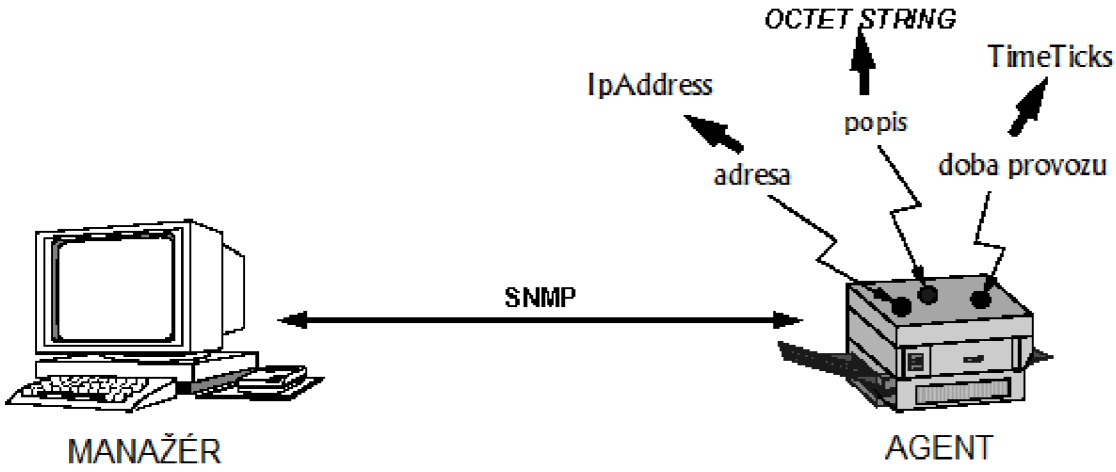
- skalármi (napr. aktuálny čas, počet paketov, atd.)

- datové typy: celé čísla, znakové reťazce, ...
- tabuľkami - dvojrozmerné pole skalárov
  - tvorené štruktúrou skalárov

Príklad SMI skalárnych dátových typov (vid obr. 2.4 prevzatý z [9]):

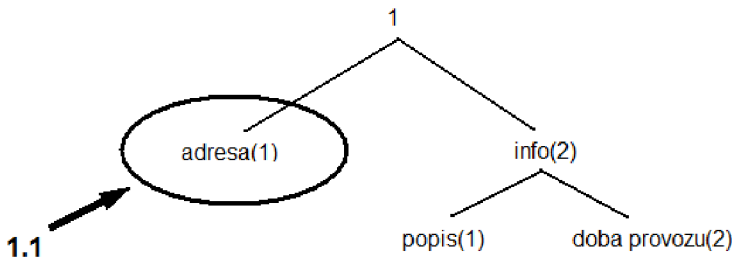
- Gauge32 - čítač, ktorý sa pohybuje v intervale <0, maxINT>, môže stúpať aj klesať
- Counter32 - klasické počítadlo, ktoré sa zvyšuje po maximálnu hodnotu a po prekročení začína počítať od 0
- Timeticks - čas v 1/100 sekundy
- IpAddress - ip adresa zariadenia
- OCTET STRING - znakový reťazec

SNMP protokol môže vymieňať medzi systémami iba skaláry. Nie je možné získať jednou operáciou celú tabuľku, tá sa prenáša postupne ako zoznam skalárnych hodnôt.



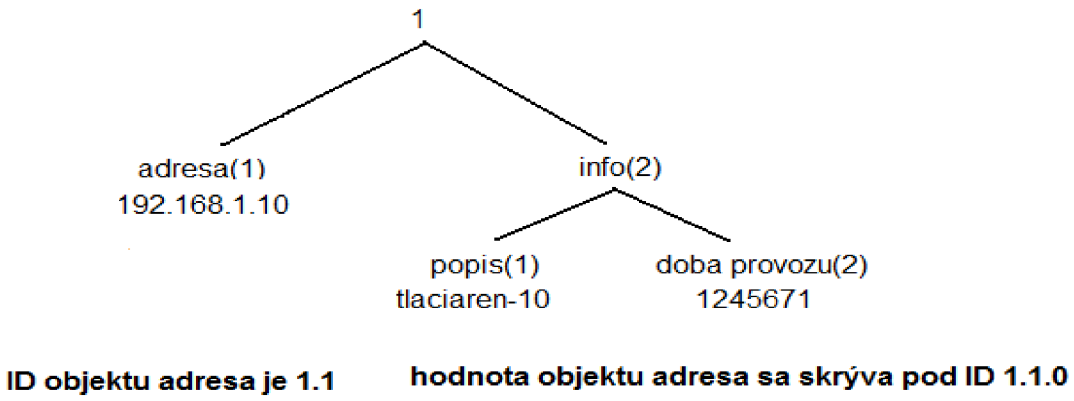
Obr. 2.4: Príklad skalárnych objektov a ich dátových typov

Aby mohol manažér jednoznačne identifikovať premenné ako adresa, popis, dobu provozu vo spravovanom systéme bol zavedený tzv. menný strom (vid obr. 2.5). Listy menného stromu reprezentujú spravované objekty.



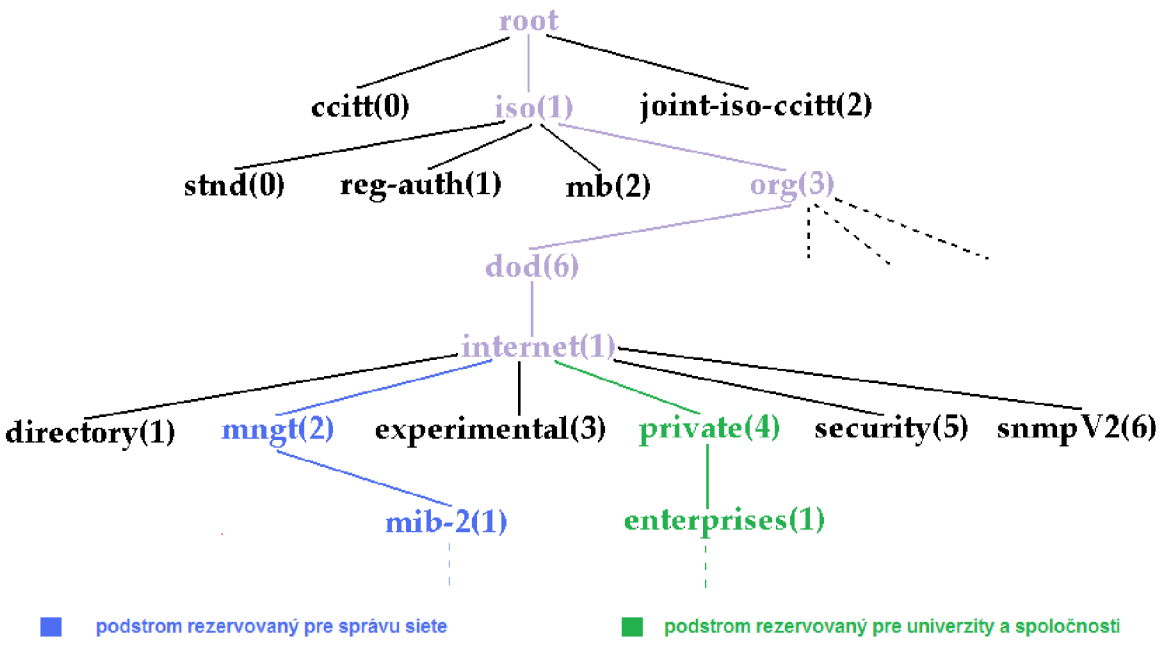
Obr. 2.5: Príklad menného stromu

Dôležité je vysvetliť aký je rozdiel medzi **objektom** a **inštanciou objektu**. Objekt považujeme za definíciu, skutočnú hodnotu daného objektu má jeho inštancia. V prípade skalárnych objektov nemá veľký zmysel rozlišovať objekt a inštanciu, avšak u tabuľkách objekt definuje ako budú vyzerať jej riadky. Keďže tabuľka môže mať niekoľko riadkov, môže existovať niekoľko inštancií daného objektu. Na rozlíšenie objektu od inštancie je potrebné u skalárnych objektov pridať za jeho názov ".0" [2].



Obr. 2.6: rozdiel medzi objektom a jeho inštanciou

V predchádzajúcich častiach sme si ukázali ako jednoznačne identifikovať jednotlivé objekty vo spravovanom systéme. Predstavme si, že systém implementuje niekoľko MIB modulov, od rôznych výrobcov, problémom je ako zabezpečiť jednoznačné označenie objektov v jednotlivých moduloch? Riešením je globálny menný strom (vid obr. 2.7), v ktorom si každý výrobca daný MIB modul zaregistruje a tým je zaručená jeho unikátnosť [3].



Obr. 2.7: Globálny menný strom

2.2 Definícia metrík

V nasledujúcej časti by som chcel vysvetliť čo to metrika je, aké sieťové výkonnostné metriky sa v praxi používajú, ako sa definujú, získavajú a aké majú jednotky.

Metrika je entita, ktorá nám umožňuje popísať výkonnosť, spoľahlivosť a operačný stav siete alebo sieťových uzlov. Metrika by nemala byť závislá od metódy, ktorá bola použitá na jej meranie a namerané výsledky musia používať štandardné jednotky (napr. bity za sekundu). Typickými príkladmi metrík je one-way-delay (oneskorenie v jednom smere), packet loss (strata paketov), atď.

Metriky môžeme rozdeliť do 2 kategórií: výkonnostné a zmiešané metriky. V práci som sa zameral na výkonnostné metriky a pokúsim sa ich rozdeliť a stručne charakterizovať.

Výkonnostné metriky môžeme rozdeliť do nasledujúcich skupín [4]:

- **Availability (dostupnosť)** je skupina, ktorou môžeme ohodnotiť robustnosť siete. Množstvo času počas, ktorého sieť funguje bez problémov a nie sú v nej chyby, ktoré znemožňujú dostupnosť služieb.
- **Loss and errors (strata a chyby)** je skupina metrík, ktoré vyjadrujú podiel paketov, ktoré sa stratili, prípadne dorazili s chybami do cieľa.
- **Delay (oneskorenie)** vyjadruje oneskorenie paketov prenášaných v sieti.
- **Bandwidth (šírka pásma)** patrí do skupiny metrík, ktoré hodnotia množstvo dát, ktoré môže užívateľ preniesť cez sieť v danej časovej jednotke.

2.2.1 Availability (dostupnosť)

Metrika dostupnosť, sa v IP sieťach zvyčajne vzťahuje k percentuálnemu množstvu času počas, ktorého bola linka alebo sieťový uzol plne funkčné (zvyčajne v stave “up”). Dostupnosť je možné vzťahovať aj na ďalšie zariadenia alebo sieťové služby.

2.2.1.1 Dostupnosť smerovača alebo linky

Definícia: Percento času v definovanom časovom intervale počas, ktorého je linka alebo smerovač v stave “up”. Pre linku to znamená schopnosť prenášať informácie v jednom alebo v oboch smeroch, pre smerovač dostupnosť jeho loopback rozhrania.

Spôsoby získavania: Existujú dva rozdielne modely na získanie dostupnosti: pull a push. V pull modeli periodické dotazy smerované na sieťové zariadenia pripojené k linke zbierajú informácie o jej stave. Manažér vytvára SNMP správy a posiela ich v pravidelných intervaloch sieťovým zariadeniam pripojených k linke. Špecifické MIB premenné definujú stav linky, na základe tejto informácie sieťové zariadenie generuje SNMP odpoveď manažérovi.

V push modeli sieťové zariadenia asynchrónne generujú trapy manažérom, keď dôjde k zmene stavu linky.

Jednotky: Zvyčajne sa jedná o percentuálne množstvo času počas, počas ktorého fungoval sieťový uzol alebo linka bez chýb. Napríklad sieťová linka bola funkčná na 99,9% počas doby jedného roka. Namerané hodnoty nemajú jednotky [4].

2.2.2 Loss and errors (stratovosť a chyby)

Popisuje koľko informácií poslaných zo zdrojového uzlu nebolo doručených alebo prijatých s chybami cieľovým uzlom v porovnaní s celkovým počtom informácií. Väčšinou je počítaná v percentách chybných bitov alebo zahodených paketov.

Definícia: Pomer medzi počtom paketov považovaných za stratené v cieľovom uzly a celkovým počtom paketov poslaných zdrojovým uzlom. Podľa [6] je paket poslaný zdrojovým uzlom smerom k cieľovému považovaný za stratený ak nedosiahne cieľový uzol v určitom časovom intervale. Ak meriame stratu v jednom smere, od zdrojového uzlu k cieľovému jedná sa o one-way packet loss (strata paketov v jednom smere), ak berieme do úvahy stratu, ktorá vzniká aj smerom naspäť jedná sa o two-way packet loss (strata v oboch smeroch).

Hlavné dôvody straty paketov sú nasledovné:

- Chyby na vrstve 1 a 2 spôsobia, že CRC kontrola na vrstve 3 zlyhá. V takomto prípade je prijatý paket zahodený.
- Preplnené fronty na routroch, v dôsledku záťaže na linkách.
- Chyby v konfigurácií na smerovačoch môžu spôsobiť zlé smerovanie paketov do slučiek. V takých prípadoch sú pakety zahodené, keď im vyprší TTL.

Spôsoby získavania:

- **One-way packet loss** - Strata paketov v jednom smere pozdĺž linky alebo cesty môže byť odmeraná aktívnymi alebo pasívnymi metódami. Pri aktívnom meraní sú do siete vkladané umelé pakety odosielateľom a zbierané príjemcom na konci cesty. Meraním straty paketov na umelom provoze môžeme odhadnúť stratu za reálneho provozu. Pasívne merania môžu byť uskutočnené hromadením dát extrahovaných z hlavičiek paketov medzi dvomi bodmi, sú oveľa presnejšie keďže merajú stratu paketov v reálnom provoze, ale je ťažšie ich implementovať v širšom rozsahu.
- **Two-way packet loss –** Strata paketov v oboch smeroch môže byť získaná iba aktívnymi meraniami. Nástroje postavené na protokole ICMP ako ping generujú umelý provoz k cieľovému uzlu, ktorý po prijatí ICMP správy generuje ICMP odpoveď.

Jednotky: Strata je reprezentovaná ako pomer počtu stratených paketov k celkovému počtu paketov generovaných v danom časovom intervale a preto sa nepoužívajú žiadne jednotky. Napríklad strata paketov medzi dvomi uzlami je vyjadrená v % počas x-minútového intervalu [4].

2.2.3 Delay (oneskorenie)

Vyjadruje čas, za ktorý sa určité množstvo dát (napr. paket danej veľkosti) prenesie zo zdrojového do cieľového uzlu. Chcel by som predstaviť dve najpoužívanejšie inštancie tejto metriky, ktorými je One-Way Delay (OWD) a Round Trip Time (RTT).

2.2.3.1 One Way Delay (OWD)

Definícia: Jedná sa o čas medzi výskytom prvého bitu paketu na zdrojovom rozhraní a výskytom posledného bitu paketu na cieľovom rozhraní [6].

Spôsoby získavania: Môžu byť použité aktívne alebo pasívne metódy. Pri aktívnych metódach sú monitorované pakety označené časovou známkou pri vstupe do siete zdrojovým uzlom. Prijímací uzol porovná časové známky každého monitorovaného paketu s časom kedy bol prijatý a vypočíta oneskorenie. Pri pasívnych meraniach, monitorované uzly zbierajú informácie z hlavičiek z reálneho provozu, ktorý cez ne prechádza. Nazbierané informácie z hlavičiek paketov sú označené časovou známkou v sledovaných bodoch. Dáta z rozdielnych sledovaných uzlov sú následne sumarizované kvôli identifikácií OWD paketov, ktoré cez ne prechádzali.

Jednotky: Jednotkou oneskorenia je sekunda, v praxi sa väčšinou používajú milisekundy. Dosahovaná presnosť pri meraní oneskorenia je ovplyvnená viacerými parametrami, pričom synchronizácia v čase medzi sledovanými bodmi je jedna z najťažšie dosiahnuteľných. Na dosiahnutie dostatočnej presnosti musia byť hodiny v sledovaných uzloch synchronizované vhodnou metódou [4].

2.2.3.2 Round Trip Time (RTT)

Definícia: RTT je považované za periódu medzi časom kým je paket s požiadavkou poslaný zdrojovým uzlom a časom kým je prijatá odpoveď z cieľového uzlu (vid [6]). Paket s odpoveďou musí byť poslaný okamžite ako bol prijatý paket s požiadavkou.

Spôsoby získavania: RTT je väčšinou merané s využitím nástroja ping, ktorý je široko dostupný na sieťových elementoch. Ping používa ICMP, firewally musia povoliť ICMP pakety. Vo väčšine nástrojov, ktoré vykonávajú RTT merania je generovaných niekoľko paketov, pre každý je spočítaný RTT. Následne je dostupné minimum, priemer, maximum alebo štandardná odchýlka.

Jednotky: RTT jednotkou je sekunda a platnou hodnotou je kladné číslo alebo nekonečno. Meranie RTT je založené na jedných hodinách, nie je potrebná synchronizácia v čase [4].

2.2.4 Bandwidth (šírka pásma)

Vo všeobecnosti šírka pásma odpovedá množstvu dát, ktoré je možné poslať cez sieť v danom časovom intervale, meraných v bitoch za sekundu (bps). Šírka pásma by mala byť vždy vzťahovaná k ISO/OSI vrstve, keďže každá vrstva je zabalená spolu s hlavičkou nižšej vrstvy a toto redukuje dostupnú šírku pásma pre vyššie vrstvy.

2.2.4.1 Capacity of a link (kapacita linky)

Definícia: Je definovaná ako maximálne množstvo dát (bitov) za jednotku času, ktoré môžeme cez linku preniesť ak na nej nie je žiadny provoz.

Spôsoby získavania: Kapacity liniek je možné väčšinou získať z pripojených routrov k jednotlivým linkám pomocou SNMP dotazov. Kapacita po ceste je definovaná ako minimálna kapacita linky, ktorá tvorí cestu.

Jednotky: Jednotkou kapacity sú bity za sekundu (bps). V rýchlych sieťach je kapacita väčšinou prezentovaná v giga alebo mega bitoch za sekundu.

2.2.4.2 Link Utilisation (využitie linky)

Definícia: Jedná sa o množstvo kapacity využitej IP paketmi (hlavičky + užitočné Dáta) počas daného časového rámca.

Spôsoby získavania: Využitie linky sa väčšinou získava z počítadiel provozu, ktoré aktualizujú routre. Tieto informácie sú získavané SNMP dotazmi na konkrétne MIB premenné a výpočet je vykonaný za určitý časový interval (typicky, priemer medzi 2 za sebou idúcimi čítaniami).

Jednotky: Využitie linky je reprezentované ako pomer kapacity okupovanej provozom k celkovej kapacite linky. Časové okno meraní je väčšinou veľké (typicky 5-15 minút) aby sa zjemnili efekty spôsobené krátkym nárazovým provozom a zmenšením záťaže na routroch, ktoré musia generovať odpoveď na každý SNMP dotaz.

2.2.4.3 Available bandwidth of a link (dostupná šírka pásma linky)

Definícia: Je definovaná ako maximálne množstvo dát, ktoré je možné preniesť za jednotku času, pri danom využití linky.

Spôsoby získavania: Väčšinou je spočítaná odčítaním využitia linky od celkovej kapacity.

Jednotky: Dostupná šírka pásma môže byť vyjadrená v percentách kapacity linky, ktorá nie je využitá.

2.3 Spracovanie metrík

V nasledujúcej časti by som chcel vysvetliť prečo je dôležité metriky definované v predchádzajúcej časti ďalej spracovávať a aké metódy je možné použiť.

Zhromažďovanie základných metrík nestačí nato aby sme pochopili správanie siete. Sieťové merania musia byť ďalej spracované aby boli užitočné pre ďalšie úlohy týkajúce sa správy siete. Jednou z prvých možností spracovania je proces skladania jednotlivých meraní do väčších celkov.

Jedným z dôvodov môže byť rozšíriteľnosť. Vzhľadom k veľkému počtu sieťových elementov je nemožné získavať Dáta zo všetkých. Preto je vhodné vybrať najdôležitejšie uzly, z ktorých budú prebiehať merania a z nich sa snažiť odvodiť metriky, ktoré charakterizujú linky medzi zvyšnými uzlami.

Ďalším dôvodom pre skladanie metrík je ich redukcia. Ak máme doménu, v ktorej je merané oneskorenie na podmnožine elementov a chceme vedieť či existuje vo všeobecnosti problém s oneskorením bolo by dobré keby sme dokázali vyhodnotiť jednu hodnotu, ktorá nám popíše oneskorenie. Táto hodnota môže byť spočítaná z jednotlivých čiastkových meraní na linkách a vážená množstvom dát prenesených cez jednotlivé linky.

Analýza z dlhodobého hľadiska je ďalším dôvodom prečo je užitočné merania skladať. Môže byť vyhodnotená jedna hodnota za hodinu, deň, mesiace zo základných meraní, ktoré prebiehajú každých päť minút.

2.3.1 Základná terminológia

V tejto sekcií si predstavíme terminológiu používanú v ďalších podkapitolách.

Podľa [5], jediné pozorovanie metriky sa nazýva *singleton metric* (unikátna metrika) jedná sa o atomickú hodnotu. Skupina unikátnych metrík sa nazýva *sample metric* (vzorka metriky) a hodnota, ktorú spočítame štatisticky na vzorke danej metriky sa nazýva *statistic metric* (štatistická metrika).

Pri spracovaní metrík existujú určité praktiky, pri ktorých vzniknutá hodnota nespadá ani do jednej zo spomínaných kategórií. Podľa [5] tieto hodnoty nazývame *derived metrics* (odvodené metriky).

Odvodené metriky môžu byť získané aplikovaním štatistických operácií na elementy vzorky, napr. priemerom, v takomto prípade sa jedná o štatistickú metriku. Avšak odvodené metriky môžu vzniknúť odlišnými spôsobmi a nemajú žiadny praktický význam, pričom pomáhajú pri ďalších operáciách. V takomto prípade nazývame tieto okamžité výsledky *help metrics* (pomocné metriky).

Ďalšou vlastnosť, ktorá sa vzťahuje k metrikám je fyzický a logický rozsah. Príkladom fyzického rozsahu metriky sú:

- Pozorovaný bod, merania sú získané z jedného bodu, rozhranie na smerovači.
- Merania sú získané medzi dvomi rozdielnymi bodmi, pozdĺž cesty.

Príkladom logického rozsahu metriky sú:

- Vlastnosti IP paketu, ktorý je použitý pri meraní, jeho veľkosť, transportný protokol, atd.

K väčšine nameraných hodnôt sa vzťahujú ďalšie informácie (metaData), zvyčajne týkajúce sa času. Prítomnosť počiatočného času ku ktorému vzťahujeme danú metriku alebo časové okno definované začiatkom a koncom. Dáta rovnakej metriky môžu byť časovo normalizované vzhľadom k danému oknu alebo nie. Príkladom je počet paketov poslaných cez dané rozhranie v danom časovom intervale môžeme podať užívateľovi ako absolútnu hodnotu alebo priemernú podelením dĺžky intervalu.

Spôsoby skladania metrík môžeme rozdeliť do 3 kategórií [4]:

1. Aggregation in time (Agregácia v čase)
2. Aggregation in space (Agregácia v priestore)
3. Concatenation in space (Spájanie v priestore)

V bakalárskej práci som sa zameral na spôsob ako agregovať metriky v čase, čo bližšie popíšem v kapitole 2.3.2. Agregácia v priestore alebo spájanie v priestore je možnosť ako systém rozšíriť.

2.3.2 Agregácia metrík v čase

Môžeme ju definovať ako skladanie metrík rovnakého typu a rozsahu získaných v rôznych časových okamžikoch alebo časových oknách. V nasledujúcich podkapitolách si pokúsime odvodiť vzťahy pomocou, ktorých sa dokážeme dopracovať od vzorky metrík k ΔT metrikám. ΔT metrika je hodnota, ktorá môže zastúpiť všetky hodnoty vzorky, ktoré spadajú do intervalu ΔT. V nasledujúcich podkapitolách si ukážeme ako môže byť zvolený interval ΔT (podkapitoly 2.3.2.1 a 2.3.2.2), ukážeme si príklady funkcií, ktoré môžu byť aplikované na vzorky v intervale ΔT (podkapitola 2.3.2.3) [4].

2.3.2.1 Pevné časové okno

Uvažujme, že máme k dispozícií vzorku S zloženú z dvoch parametrov <čas, metrika> s hodnotami $<T_i, M_i>$, získanými v časovom okne (T_0, T_f). Ak rozdelíme časové okno (T_0, T_f) na N po

sebe idúcich časových intervalov ($T_{a,k-1}$; $T_{a,k}$). Hodnoty <T_i, M_i>, ktoré padnú do tohto intervalu, vytvoria menšiu vzorku S_k z pôvodnej vzorky S.

Pre jednoduchosť uvažujme prípad, keď rozdelíme pôvodnú vzorku S na po sebe idúce rovnako veľké časové intervaly s trvaním ΔT, tieto intervaly nazývame aj pevné časové okná. Z toho vyplýva:

$$T_{a,k} - T_{a,k-1} = \Delta T, \text{ pre } k = 1, \ldots, N \quad (2.1)$$

$$T_{a,k} = T_0 + \Delta T * k \quad (2.2)$$

Trvanie ΔT vyjadruje rozlíšenie agregácie. Ak sú pevné časové okná zarovnané na rozlíšenie, nazývame ich „*slotted time windows*“.

Teraz môžeme definovať agregovanú hodnotu V_k v časovom intervale ($T_{a,k-1}$; $T_{a,k}$) ako:

$$V_k = F(M_i), \text{ pre každé } <T_i, M_i> \text{ pre ktoré } T_{a,k-1} <= T_i < T_{a,k} \quad (2.3)$$

Kde F je kompozičná funkcia aplikovaná na vzorku S_k, M_i je množina unikátnych hodnôt v intervale ($T_{a,k-1}$; $T_{a,k}$). Takto pre každý interval, ktorý vznikol rozdelením pôvodného okna, vzniká nová metrika <časový interval, agregovaná hodnota>, ktorú nazývame “*ΔT metrika*“ a formálne označujeme ako <($T_{a,k-1}$; $T_{a,k}$), V_k>.

2.3.2.2 Voľné časové okno

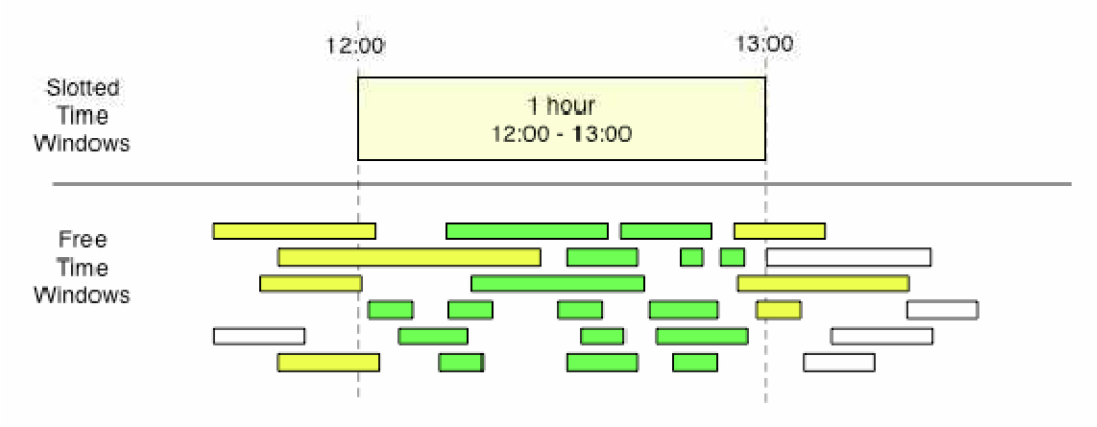
V týchto prípadoch časové intervaly, na ktoré rozdelíme časové okno (T_o, T_f) nie sú rovnako veľké, časové intervaly sa nazývajú voľné časové okná. Príkladom meraní kde sú atomické hodnoty vzťahované k voľným časovým oknám sú merania, pri ktorých si značíme počiatok a majú určité trvanie. Ak sa počiatočný čas a jeho trvanie nezmestia do zarovnaného časového okna, nemôžeme s unikátnou hodnotou ďalej pracovať.

Nezarovnané vzorky vedú ku klamným štatistickým hodnotám (vid obr. 2.8 prevzatý z [4]). Preto je dôležité aby sa všetky merania, ktoré sa nachádzajú vo voľných časových oknách konvertovali do pevných okien vždy keď je to možné. Existujú tri spôsoby ako uskutočniť nápravu:

1. Presunieme merania do časového okna, v ktorom sa nachádza počiatočný čas. Čo spôsobí posun o niekoľko časových okien do skoršieho času.
2. Presunieme merania do časového okna, v ktorom sa nachádza koncový čas. Čo spôsobí posun o niekoľko časových okien do neskoršieho času.

3. Rozložíme množstvo hodnôt metrík do niekoľkých časových okien, vážených dĺžkou intervalu, ktorý spadá do daného okna.

Tretí prístup sa javí ako najlepší ale je zložitý na implementáciu. V prípadoch kde iba malé percento hodnôt metrík spadá do viacerých slotov je posun oveľa jednoduchší.



Obr. 2.8 Príklad pevných a voľných časových okien

Rozlišujeme dva typy ΔT metrík: ΔT štatistická metrika (podkapitola 2.3.2.3) a ΔT pomocná metrika (podkapitola 2.3.2.4).

2.3.2.3 ΔT štatistická metrika

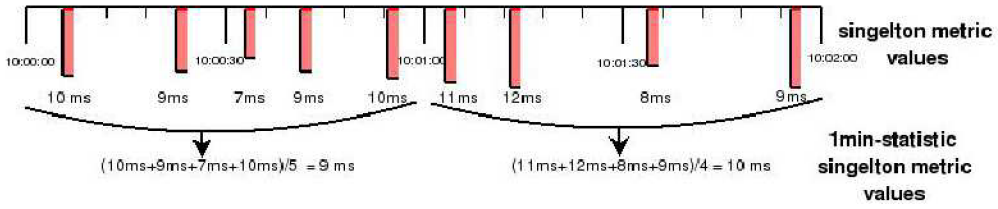
Ak je na vzorku v pevnom časovom okne použitá štatistická funkcia výsledkom je štatistická metrika. Najpoužívanejšie funkcie sú:

1. Aritmetický priemer

$$\bar{x} = \frac{1}{n} \cdot \sum_{i=1}^{n} x_i \tag{2.4}$$

- kde $x_1..x_n$ je postupnosť reálnych čísel

Ak máme k dispozícii vzorku oneskorení v jednotlivých časoch, zvolíme časový interval ΔT pre ktorý spočítame priemer dostávame ΔT priemer oneskorenia. Všetky spočítané priemerné oneskorenia v intervale ΔT tvoria priemernú ΔT vzorku oneskorenia (vid obrázok 2.9 prevzatý z [4]).



Obr. 2.9 Príklad agregácie v čase pomocou priemeru

2. Medián

   Je hodnota od ktorej je rovnaký počet hodnôt menších a väčších ako je hodnota mediánu (pri nepárnom počte), pri párnom počte je to priemer 2 hodnôt v strede. V mnohých prípadoch je medián oveľa užitočnejší ako priemer pretože pri priemere dokáže niekoľko extrémnych hodnôt radikálne ovplyvniť výsledok.

3. Minimum a maximum
4. X-percentil

   Percentil je hodnota, ktorá je väčšia ako X % v danom štatistickom súbore. Keďže minimum a maximum sa môže voliteľne meniť je percentil v mnohých prípadoch užitočnejší.

5. Smerodajná odchýlka

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})^2} \tag{2.5}$$

   Je definovaná ako odmocnina z priemeru druhých mocnín odchýlok jednotlivých hodnôt štatistického súboru od aritmetického priemeru a určuje nám mieru toho ako široko sú hodnoty v súbore rozložené.

2.3.2.4 ΔT pomocná metrika

Kompozičná funkcia nemusí byť vždy štatistickou funkciou. Môžeme použiť inú matematickú funkciu, ktorú aplikujeme na hodnoty vzorky v danom časovom intervale, čo môže viesť k vytvoreniu inej ΔT metriky, ktorú nazývame ΔT pomocná metrika. Príkladom môže byť uloženie sumy štvorcov, ktorá môže byť neskôr použitá na výpočet štandardnej odchýlky a pôvodné hodnoty nie sú potrebné.

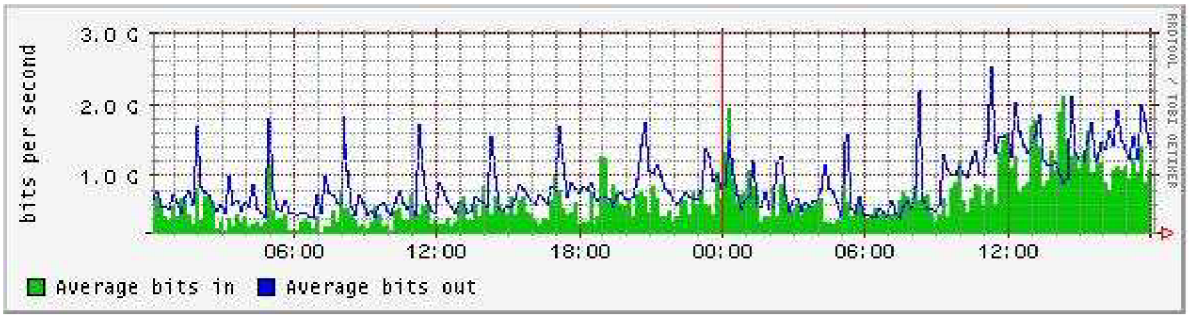
2.4 SNMP aplikácie

V tejto kapitole by som chcel predstaviť niektoré nástroje, ktoré sa v dnešnej dobe používajú pri správe siete a poukázať na ich hlavné výhody.

2.4.1 MRTG (Multi router traffic grapher)

Jedná sa o voľne dostupný a plne konfigurovateľný nástroj pre trendovú analýzu. Je prekvapivo malý, odľahčený balík pretože neimplementuje zbytočne zložité užívateľské rozhranie. Generuje grafy vo forme GIF, PNG obrázkov, ktoré znázorňujú denné (vid obr. 2.10), týždenné, mesačné a ročné zaťaženie siete a vkladá ich do webových stránok. Takto je možné výstup z MRTG prezerať webovým prehliadačom alebo zviditeľniť výsledky pre celú sieť umiestneným stránok na webový server.

Jedná sa o veľmi užitočný nástroj pri diagnostike problémov, pretože jednak znázorňuje aktuálny stav siete, ale je možné ho opticky porovnať s predošlým stavom. MRTG je najlepší pri zobrazovaní grafov využitia jednotlivých rozhraní na routri, ale môže byť nakonfigurovaný na vizualizáciu rôznych vecí (napr. využitie pamäte, priemerný výkon, využitie disku na serveroch). MRTG získava Dáta zo smerovačov a ďalšieho sieťového hardwaru pomocou protokolu SNMP. Implicitne posiela svoje dotazy každých 5 minút a ukladá ich do špeciálneho formátu, ktorý zabraňuje neustálemu zväčšovaniu súborov, ale Dáta vhodne sumarizuje [1, 2].



Obr. 2.10: príklad dennej záťaže na jednom rozhraní routra [12]

Pri riešení sieťových problémov je veľmi výhodné dokázať sa spätne pozrieť na históriu provozu. V praxi sa na základe okamžitých hodnôt prenosu za sekundu dá len ťažko zistiť, či sa sieť chová normálne. Ak máme k dispozícii linku s prenosovou kapacitou 100Mbit/s a aktuálne sa po nej prenáša 85Mbit/s. Jedná sa o normálne zaťaženie alebo preťaženie spôsobené útokom? Tým, že máme k dispozícií históriu provozu siete, môžeme hľadať náhle zmeny, ktoré môžu byť dôvodom problémov v chovaní siete. Útok typu DOS je charakteristický náhlym trvalým zvýšením prenosu, čím vyčerpáva dostupnú prenosovú kapacitu.

2.4.2 Nagios a Sysmon

Administrátori sa často stretávajú s problémom, že niečo prestane fungovať - vypadne sieťová linka, prestanú fungovať určité služby na serveroch. O podobných výpadkoch sa administrátor väčšinou automaticky nedozvie. Smerovač, v ktorom je softwarová chyba nemá možnosť ako administrátora upozorniť, že nefunguje.

A práve v takýchto prípadoch prichádzajú tzv. dohladové alebo dotazovacie programy, medzi ktoré patrí napr. Nagios a SysMon. Tieto programy bežia na serveri, z ktorého sa v pravidelných intervaloch dotazujú napr. na dostupnosť služieb, prípadne kontrolujú sieťovú konektivitu. Programy môžu pingovať jednotlivé zariadenia, generovať SNMP dotazy alebo vytvárať iné testy na dostupnosť služieb. Ak im cieľové zariadenie neodpovie môžu ohlásiť dané zlyhanie administrátorovi.

Dohladový systém môže mať veľký vplyv na výkon siete alebo sledovaných zariadení. Krátke intervaly pre generovanie dotazov môže spôsobiť zahltenie siete, prípadne spomaliť smerovače spracovávaním veľkého počtu SNMP transakcií [2].

2.4.3 Net-SNMP

Jedná sa o veľmi užitočný balíček nástrojov. Obsahuje napríklad nasledujúce nástroje, ktoré je možné použiť z príkazovej riadky:

- snmpget, snmpgetnext - získavajú hodnoty zo spravovaného zariadenia použitím jednej požiadavky
- snmptable - dokáže získať zo vzdialenej MIB celú tabuľku a prehladne ju zobraziť
- snmpset - dokáže meniť konfiguračné premenné na spravovanom zariadeníň
- snmptranslate – umožňuje nám preklad medzi textovou a numerickou formou MIB identifikátorov a zobrazuje obsah MIB a jej štruktúru

V balíčku sa nachádza aj rozšíriteľný agent, ktoré je schopný odpovedať na SNMP dotazy týkajúce sa správy siete (snmpd). Má vstavanú podporu pre široký rozsah MIB modulov a môže byť ďalej rozšíriteľný za použitia dynamických modulov, externých skriptov a príkazov. Net-SNMP je dostupný pre unixové operačné systémy aj pre windows, pričom funkcionalita je závislá od daného systému [11].

3 Návrh riešenia

Systém by mal spĺňať požiadavky zadávateľa, preto by bolo dobré ich zhrnúť a analyzovať. Systém má spĺňať nasledujúce požiadavky:

- Pomocou protokolu SNMP komunikuje so sieťovými uzlami.
- Zbiera atribúty skupiny uzlov na spoločnom prenosovom médiu a snaží sa z nich odvodiť nové charakteristiky.
- Systém má sledovať stav sieťových spojení a vhodnou formou ich prezentovať.

Výsledný systém som sa rozhodol implementovať formou hybridného agenta, ktorý má nasledujúce funkcie.

- Pomocou protokolu SNMP získava v pravidelných časových intervaloch z navzájom prepojených uzlov atribúty.
- Zo získaných atribútov počíta metriky.
- Z metrík sa počítajú štatistiky.
- Metriky a ich štatistiky nám slúžia na ohodnotenie linky medzi sieťovými uzlami.
- Aktuálne metriky aj štatistiky sa priebežne aktualizujú v MIB a sú dostupné ostatným nástrojom, ktorý ich môžu získavať cez protokol SNMP.
- Agent by mal byť multiplatformný a implementovaný ako služba alebo démon, ktorý beží na pozadí.

V zadaní bakalárskej práce bola vyslovená požiadavka k vytvoreniu systému, ktorý by sledoval stav sieťových spojení a umožňoval by prezentovať výsledky vhodnou formou. Keďže agent neumožňuje vytvárať priamo grafický výstup, metriky a štatistiky jednotlivých spojení sú uložené v jeho MIB databáze je potrebné využiť existujúce nástroje na ich prezentáciu (napr. MRTG).

3.1 Návrh rozhrania

Keďže jedinou úlohou agenta je získavať z navzájom prepojených sieťových elementov hodnoty, z ktorých následne počíta metriky, štatistiky a aktualizuje ich vo svojej MIB databáze, jedinou formou ako komunikovať s agentom je pomocou protokolu SNMP (sú podporované všetky verzie).

3.2 Voľba implementačných prostriedkov

Existuje veľké množstvo rôznych SNMP aplikačných programových rozhraní, ktoré umožňujú vytváranie aplikácií pre správu siete. Väčšina týchto rozhraní poskytuje veľké množstvo funkcií, čo vyžaduje od programátora dobrú znalosť fungovania SNMP. Veľa rozhraní je závislá od platformy a tým pádom je napísaný kód neprenositeľný.

Keďže som počas štúdia nadobudol veľké skúsenosti v objektovo orientovanom jazyku C++, snažil som sa pre výsledný systém nájsť rozhranie implementované práve v tomto jazyku. Objektovo orientovaný prístup k programovaniu sieťových SNMP aplikácií poskytuje veľa výhod ako ľahké použitie, bezpečnosť, prenositeľnosť a rozšíriteľnosť. Výsledného agenta som sa rozhodol implementovať s využitím aplikačného rozhrania SNMP++, Agent++.

3.2.1 SNMP++

SNMP++ je množina C++ tried, ktoré poskytujú SNMP služby programátorom sieťových aplikácií zaoberajúcich sa správou. SNMP++ nie je ďalšia vrstva alebo obálka nad existujúcimi rozhraniami, je postavené na základných štandardných knižniciach a preto je plne prenositeľné na rôzne operačné systémy (napr. linux, windows) [13].

Niektoré zo základných vlastností, ktoré nám SNMP++ poskytuje:

- jednoduché SNMP rozhranie
- možnosť programovať na nižších SNMP vrstvách
- podpora protokolu SNMPv1, SNMPv2, SNMPv3
- jednoduchá rozšíriteľnosť dosiahnutá dedičnosťou, prípadne preťažením základných tried

3.2.2 Agent++

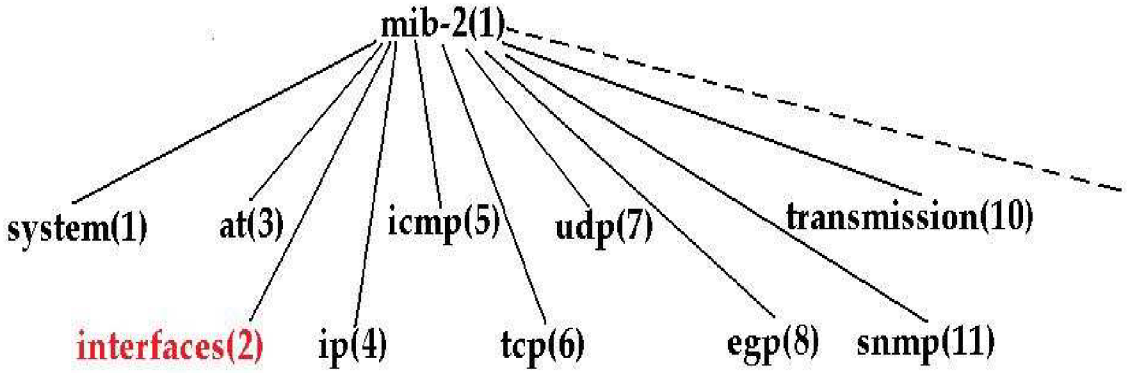
Agent++ rozširuje základné koncepty SNMP++ tým, že umožňuje tvorbu SNMP agentov. Jedná sa o množinu tried, ktorá zabezpečuje kompletnú funkčnosť SNMP protokolu a priraďovaciu tabuľku pre vývoj SNMP agentov. V rozhraní sa nachádzajú základné triedy pre správu MIB tabuliek, objektový prístup skrýva mnoho vnútorných detailov ako správa prichádzajúcich SNMP požiadaviek, mapovanie na metódy pre spracovanie, posielanie odpovedí a pascí [14].

4 Implementácia

4.1 Implementované metriky

V tejto kapitole by som chcel predstaviť MIB-2 poukázať na atribúty, na ktoré som sa zameral. Následne budú predstavené metriky, ktoré som z daných atribútov odvodil a štatistické funkcie, ktoré som sa pre jednotlivé metriky rozhodol uplatniť.

MIB-2 je štandardná MIB, nachádza sa v nej zhruba 170 premenných, pričom väčšina z nich je dostupná iba na čítanie, počet premenných, ktoré je možné meniť je obmedzený. MIB-2 tvorí niekoľko skupín, ktoré sú definované v oddelených moduloch (vid obr. 4.1).



Obr. 4.1: MIB-2 a jej podskupiny

Výhodou MIB-2 je to, že musí byť implementovaná každým zariadením preto som sa zameral práve na skupiny nachádzajúce sa v danej MIB. Jedinou výnimkou sú premenné, ktoré pre dané zariadenie nemajú žiadny význam. Napríklad bridge (most), ktorý nemá TCP protokol nemusí implementovať premenné na jeho správu. V MIB-2 je implementovaný malý počet objektov, ktorých hodnoty môžeme zmeniť a tým ovplyvniť správanie systému. Tvorcovia sa snažili o odstránenie redundancie, nenachádzajú sa tu premenné, ktoré sa dajú odvodiť od ostatných.

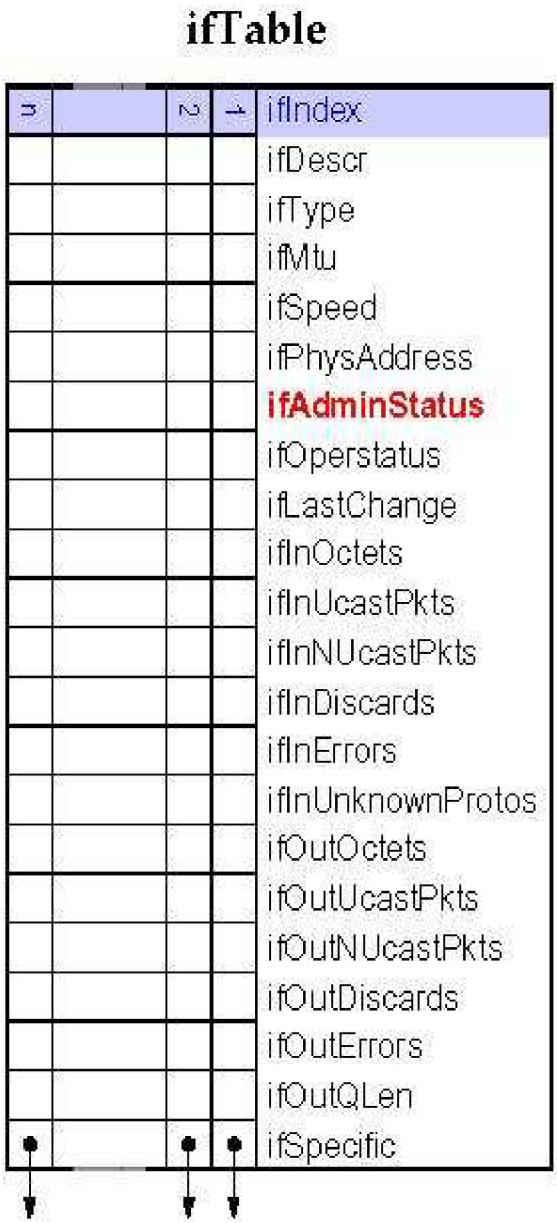
V štandardnej MIB-2 sa mi nepodarilo nájsť žiadne premenné, z ktorých by som dokázal zistiť vyťaženie procesora, pamäte, a pod. Zameral som sa hlavne na skupinu rozhraní (interfaces, vid obr. 4.1), ktorá charakterizuje jednotlivé rozhrania daného sieťového uzlu. V skupine rozhraní sa nachádza tabuľka rozhraní, ktorá obsahuje základné premenné, ktoré je možné použiť pri definovaní metrík (vid obr. 4.2 prevzatý z [10]).

Z daných atribútov som sa rozhodol spočítať nasledujúce metriky:

- dostupnosť
- šírka pásma
- stratovosť

V nasledujúcich podkapitolách by som chcel vysvetliť ako sú dané metriky definované, aké používajú jednotky a ktoré štatistické funkcie som sa rozhodol použiť.

ifSpeed - rýchlosť, ktorou operuje dané rozhranie
ifOperstatus - stav rozhrania (up/down)
ifInOctets - počet bytov prijatých rozhraním
ifInUcastPkts - počet prijatých unicast paketov
ifInNUcastPkts - počet prijatých non-unicast paketov
ifInDiscards - počet zahodených paketov (väčšinou dôsledkom nedostatku zdrojov, preplnená fronta)
ifInErrors - počet prijatých paketov s chybou
ifOutOctets - počet odoslaných bytov
ifOutUcastPkts - počet odoslabných unicast paketov
ifOutNCastPkts - počet odoslaných non-unicast paketov
ifOutDiscards - počet zahodených paketov pri odosielaní
ifOutErrors - počet paketov s chybou pri odosielaní



Obr. 4.2: Tabuľka rozhraní a ich základných atribútov

4.1.1 Dostupnosť

V agentovi sú pre dostupnosť vyhradené 3 premenné, ktoré určujú aktuálny stav linky, počet zlyhaní a dostupnosť. Pod pojmom dostupnosť budeme v agentovi rozumieť percento času v danom pevne stavenom intervale počas, ktorého bola linka, ktorá spája dva sieťové uzly schopná prenášať informácie (v stave "up").

Z premennej *ifOperstatus* som bol schopný zistiť v akom stave sa nachádza dané rozhranie, či je v stave „up" alebo „down". Linka, ktorá spája dve rozhrania je vo funkčnom stave, keď sú oba jej koncové uzly v stave „up". Ak sa aspoň jedno rozhranie nachádza v stave "down" považujem celú linku za nefunkčnú. Ak sa mi nepodarilo spojiť sa so sieťovým uzlom, nebol zistený stav rozhrania je linka v neznámom stave.

Dostupnosť nám nevyjadruje frekvenciu v akej dochádzalo k zmene stavu linky nato nám poslúži dodatočná informácia o počtu zmien, ku ktorým na linke došlo. Je možné odvodiť ďalšie metriky: Mean Up Time (priemerný čas, počas ktorého bola linka vo funkčnom stave), Mean Down Time (priemerný čas počas, ktorého bolo spojenie v stave „down"), Mean Time Between Failures (priemerný čas medzi poruchami) vid obr. 4.3.

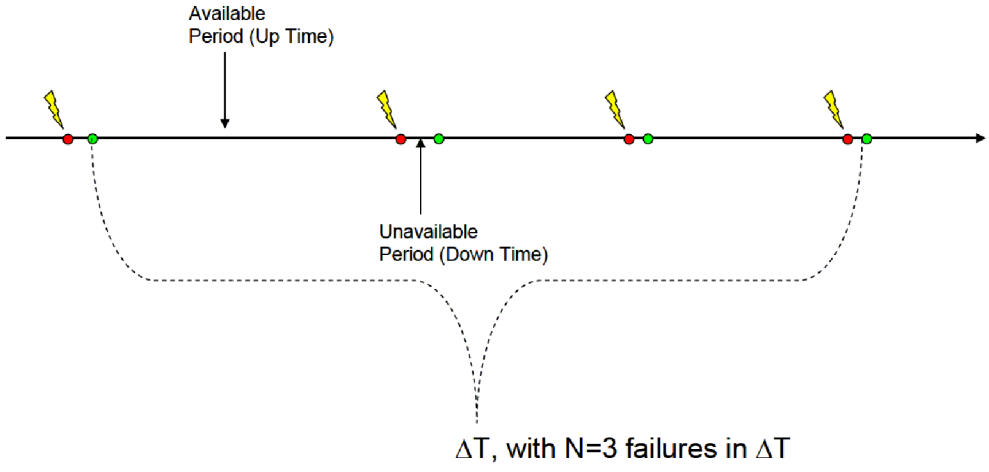
S použitím nasledujúcich rovníc je možné vyjadriť dané metriky:

$$MUT = \frac{A \times \Delta T}{N} \tag{4.1}$$

$$MDT = \frac{(1-A) \times \Delta T}{N} \tag{4.2}$$

$$MTBF = \frac{\Delta T}{N} \tag{4.3}$$

Pričom A je dostupnosť, N počet zlyhaní linky počas intervalu ΔT.



Obr. 4.3: Príklad odvodených metrík

Zo štatistických funkcií, ktoré boli predstavené v kapitole 2.3.2.3 som sa rozhodol použiť priemer, pre odhad maximálnej dostupnosti je použitý percentil 97,5%, na získanie minima zase percentil 2,5%.

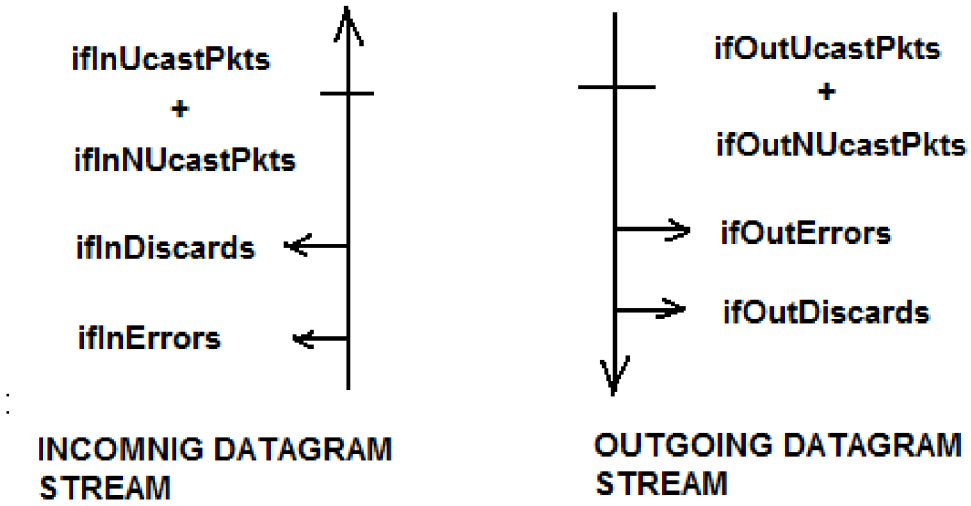
4.1.2 Šírka pásma

Zameral som sa na využitie linky a dostupnú šírku pásma. Z premennej *ifSpeed* som dokázal zistiť rýchlosť na akej operujú dané rozhrania, následne premenná *ifInOctets* nám určuje množstvo bytov, ktoré boli prijaté daným rozhraním. Pomerom množstva prijatých bytov k celkovému množstvu dát, ktoré môžu byť prenesené za časový interval dostaneme využitie linky, ktoré je v databáze vyjadrené ako percentuálne využitie linky <0%, 100%>. Odčítaním percentuálneho využitia linky od 100% získame dostupnú šírku pásma.

Z dlhodobého hľadiska je zaujímavé sledovať strednú hodnotu a percentil, preto som sa rozhodol pre dané metriky použiť dané funkcie. V prípade využitia linky je rozumné sledovať maximálne hodnoty (percentil 97,5%), v prípade dostupnej šírky pásma zase minimálnu hodnotu (percentil 2,5%) aby sme vedeli odhadnúť aká rýchlosť bude po danej linke dostupná.

4.1.3 Stratovosť

Na obrázku 4.4 môžeme vidieť ako prebieha počítanie paketov v SNMP agentoch, pri prijímaní a odosielaní po sieti.

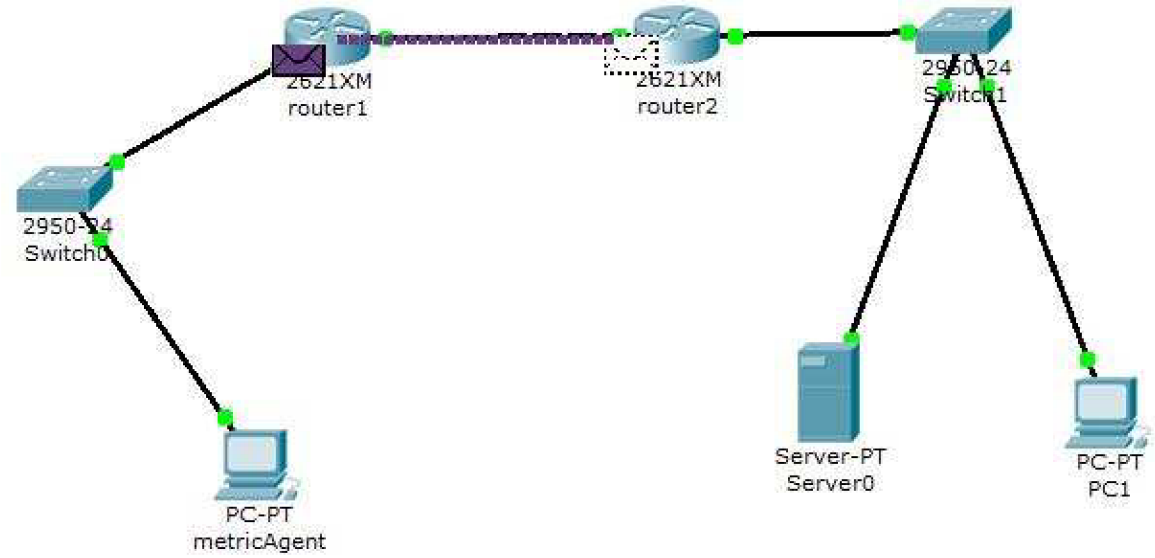


Obr. 4.4: Príklad počítadiel pre prichádzajúce a odchádzajúce pakety

Stratené pakety sú počítané nasledujúcou rovnicou:

lostPackets = abs((ifOutUcastPkts + ifOutNUcastPkts - ifOutErrors - ifOutDiscards) - (ifInUcastPkts + ifInNUcastPkts)) (4.1)

Strata je vyjadrená ako množstvo paketov, ktoré nedorazili do cieľového uzlu, dorazili s chybou, prípadne boli zahodené v dôsledku nedostatku zdrojov. Dáta o stratených paketoch nie sú časovo normalizované, udávajú sa ako absolútna hodnota za časový interval, ak chceme získať priemernú hodnotu treba podeliť dĺžkou intervalu.



Obr. 4.5: Ukážka získavania dát z jednotlivých sieťových uzlov

Predstavme si situáciu na obr. 4.5. Máme 2 počítačové siete, ktoré sú navzájom prepojené pomocou smerovačov. Ak chceme získať počty stratených paketov medzi routrom1 a routrom2, ku ktorým došlo za určitý časový interval je potreba, aby agent posielal routrom v pravidelných intervaloch požiadavky.

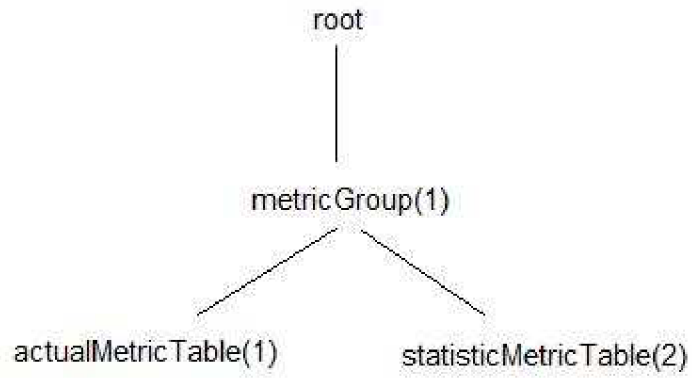
Problém, ktorý v danej situácií vzniká je ten, že požiadavky môžu doraziť do cieľových uzlov v rôznych časoch, čoho dôsledkom môžu byť nezodpovedajúce si hodnoty počítadiel. Práve pre tento dôvod je v rovnici (4.1) použitá absolútna hodnota, keďže strata môže byť tým pádom aj záporná.
Pri veľkých prenosových rýchlostiach, ktoré sa často medzi smerovačmi dosahujú môže aj malý časový rozdiel medzi dorazením požiadavky do zdrojového a cieľového smerovača znamenať veľkú stratovosť. Preto je strata paketov sporná a netreba ju chápať doslovne. V plne funkčných ethernetových sieťach je prakticky 0% strata, ktorá by bola spôsobená samotným prenosom po linke. Strata pri ethernete býva spôsobená zahadzovaním paketov v dôsledku nedostatku zdrojov (napr. pamäť), pričom v bezdrôtových sieťach môže byť strata spôsobená práve pri prenose.

Samotná aktuálna strata počítaná podľa vzorca (4.1) pre nás nemá veľkú vypovedajúcu hodnotu, je potrebné sledovať zmenu straty paketov v čase. Nato nám slúžia štatistické funkcie ako priemer, stredná hodnota, smerodajná odchýlka, ktoré som sa rozhodol použiť.

Smerodajnú odchýlku a priemernú stratu paketov môžeme využiť pri charakterizovaní straty na danom spojení. Ak sa aktuálna hodnota straty paketov pohybuje v intervale < priemer - σ , priemer + σ > znamená, že počet stratených paketov spadá do intervalu prípustných hodnôt, takže k strate prakticky nedošlo. Ak sa aktuálna hodnota výraznejšie líši od priemernej hodnoty znamená to, že sa objavil extrém, ktorý môže predstavovať zmenu v rýchlosti prenosu dát, avšak pri dlhodobejšom zbieraní dát, ktoré sa prenášajú zhruba rovnakou rýchlosťou môže tento extrém predstavovať stratu.

4.2 Implementácia MIB

V agentovi som sa pokúsil vytvoriť MIB pre ukladanie metrík a ich štatistík, z ktorej ich môžu externí manažéri pomocou protokolu SNMP získavať a využívať na vlastné účely. Na obrázku 4.6 môžeme vidieť umiestnenie jednotlivých častí MIB v mennom strome.



Obr. 4.6: Umiestnenie MIB v mennom strome

MIB je tvorená jednou skupinou, v ktorej sa nachádzajú 2 tabuľky:

- actualMetricTable(vid obr. 4.7) - Jedná sa o tabuľku aktuálnych metrík, ktoré boli spočítané za posledný časový interval (napr. posledných 5 minút).
- statisticMetricTable(vid obr. 4.8) - Jedná sa o tabuľku štatistík, v ktorej sa jednotlivé metriky agregujú v čase pomocou funkcií (napr. priemer, medián, percentil, atd.).

actualMetricTable(1)

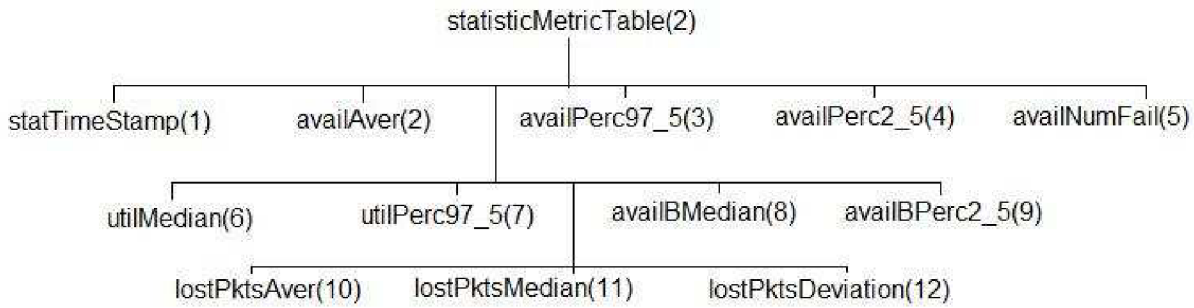
timeStamp(1)	linkStatus(2)	linkAvail(3)	linkFail(4)	linkUtil(5)	linkABand(6)	linkLostPkts(7)	srcIpAddr	destIpAddr
10:30	1	60	1	30	70	110	192.168.1.1	192.168.1.2
10:30	1	100	0	60	40	0	192.168.1.3	192.168.1.4

Obr. 4.7: Tabuľka aktuálnych metrík

Tabuľka aktuálnych metrík obsahuje okrem metrík, ktoré sme spomenuli v kapitole 3.2. premennú timeStamp (časová známka), ktorá nám slúži na určenie počiatočného času, ku ktorému vzťahujeme danú metriku. Nachádzajú sa tu ešte premenné srcIpAddr (zdrojová IP adresa) a destIpAddr (cieľová IP adresa), ktoré slúžia ako indexy do tabuľky a jednoznačne identifikujú smer od zdrojového do cieľového rozhrania, tak chápeme aj samotným metrikám.

Jednoznačná identifikácia metriky má tvar:
root.metricGroup(1).actualMetricTable(1).metricOID.srcIpAddr.destIpAddr

Ak by sme napríklad chceli získať metriku, ktorá charakterizuje počet stratených paketov medzi rozhraním s IP adresou 192.168.1.1 a 192.168.1.2, jednoznačný číselný identifikátor by mal tvar: root.1.1.7.192.168.1.1.192.168.1.2.



Obr. 4.8: Objekty štatistickej tabuľky

Jednoznačná identifikácia štatistiky má tvar:
root.metrigGroup(1).statisticMetricTable(2).statMetricOID.srcIpAddr.destIpAddr

Ak chceme získať priemernú stratu paketov medzi rozhraním s IP adresou 192.168.1.1 a 192.168.1.2, jednoznačný identifikátor by mal tvar:

root.1.2.10.192.168.1.1.192.168.1.2.

4.3 Testovanie

Testovanie patrí medzi najdôležitejšie etapy vývoja a netreba ho podceňovať. Dostatočný čas venovaný testovaniu môže odhaliť kritické chyby v programe, ktoré boli spôsobené pri implementácií. Výsledný systém treba podrobiť okrajovým podmienkam a sledovať ako na ne bude reagovať.

Pri agentovi sa testovala hlavne jeho komunikácia s ostatnými vzdialenými agentami, umiestnenými v sieťových uzloch. Pri testovaní boli navzájom prepojené dva stolné počítače, na ktorých bol nainštalovaný univerzálny snmp agent simulujúci skutočných agentov. Agent získaval z daných počítačov atribúty pomocou, ktorých počítal metriky charakterizujúce dané "p2p" spojenie. Takto bola testovaná klientská stránka agenta, pri ktorej boli objavené a následne opravené chyby pri pretekaní jednotlivých počítadiel.

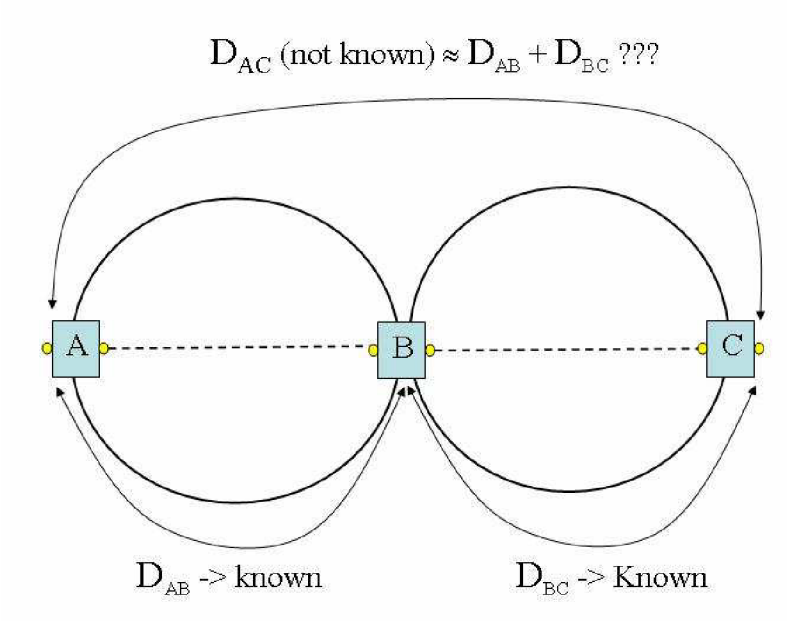
Ďalšou testovanou časťou bola serverová časť agenta. Tá bola testovaná pomocou snmp nástrojov, ktoré posielali požiadavky na dané MIB objekty. Tu neboli objavené žiadne závažné chyby.

4.4 Možné rozšírenia

Prvým možným rozšírením by bolo pridanie možnosti generovania grafického výstupu vo forme grafu, momentálne je táto možnosť dostupná len za pomoci nástroja MRTG, ktorý dokáže generovať grafy vo formáte png z jednotlivých sledovaných MIB objektov.

Ďalším z možných rozšírení by mohla byť definícia "informov". Pre niektoré typy metrík by bola stanovená hraničná hodnota, pri ktorej prekročení by boli informované riadiace stanice.

Agent agreguje jednotlivé metriky v čase, zaujímavým rozšírením by bola možnosť spájať metriky v priestore. Skladanie metrík v priestore môžeme definovať ako kompozíciu metrík rovnakého typu ale odlišného neprekrývajúceho fyzického rozsahu. Výsledná metrika reprezentuje hodnotu, ktorá by bola dosiahnutá len priamym meraním cez sekvenciu jednotlivých úsekov (vid obr. 4.9 prevzatý z [4]). Dobrým uplatnením spájania metrík v priestore je možnosť získania dostupnej šírky pásma po ceste.



Obr. 4.9: Spájanie metrík v priestore

5 Zhodnotenie

Pri vypracovaní projektu som mal možnosť získať zaujímavé vedomosti o problematike správy siete, s ktorou som sa počas štúdia stretol len minimálne. Získal som väčší prehľad o nástrojoch pre správu siete, ktoré sa v dnešnej dobe používajú.

Náplňou systému, ktorý som vyvíjal bolo sledovanie stavu sieťových zariadení prostredníctvom protokolu SNMP, ktorý zabezpečuje samotnú komunikáciu. Z jednotlivých prepojených uzlov boli vybrané atribúty, z ktorých sa následne spočítali metriky, ktoré sa agregovali v čase a slúžili na charakterizovanie spojenia medzi uzlami. Systém bol implementovaný ako služba bežiaca na pozadí.

Systém je možné použiť na charakterizovanie stavu medzi jednotlivými sieťovými uzlami, pričom vzhľadom na problémy, ktoré boli spomenuté v kapitole 4.1.3 nie je možné získať plnohodnotný konzistentný snímok stavu siete.

Ďalší vývoj samotného programu by sa mal v prvom rade týkať vecí, ktoré boli spomenuté v kapitole možných rozšírení (kapitola 4.4). Pričom na vyhodnotenie určitých typov metrík je prístup protokolu SNMP nevhodný a preto by sa skôr zišlo použitie rôznorodých nástrojov, ktoré sa v dnešnej dobe používajú pri podobných systémoch veľmi často.

Literatúra

[1] Douglas R. Mauro & Kevin J. Schmidt: Essential SNMP, O'Reilly & Associates, 2001.

[2] James M. Kretchmar, Libor Dostálek: Administrace a diagnostika sítí pomocí OpenSource utilit a nástroju, Computer Press, Brno, 2004, ISBN 80-251-0345-5.

[3] David T. Perkins, Evan McGinnis: Understanding SNMP MIBs, Prentice Hall, 1997.

[4] Geant2 - Deliverable DJ1.2.3 Network Metric Report, 14.2.2006 - http://www.geant2.net/upload/pdf/GN2-05-265v4-Deliverable_DJ1-2-3_Network_Metric_Report.pdf.

[5] V. Paxson, RFC 2330 - Framework for IP Performance Metrics, [online], Dokument dostupný na URL: http://www.ietf.org/rfc/rfc2330.txt, Posledný prístup 10.4. 2009.

[6] G. Almes, RFC 2680 - A one-way Packet Loss Metric for IPPM, [online], Dokument dostupný na URL: http://www.ietf.org/rfc/rfc2680.txt, Posledný prístup 5.3. 2009.

[7] Introduction to SNMP, Aiko Pras, [online], Prednáška dostupná na URL: http://www.simpleweb.org/tutorials/video/04-intro-snmp.zip, Posledný prístup 23.2. 2009.

[8] Management Information Bases, Aiko Pras, [online], Prednáška dostupná na URL: http://www.simpleweb.org/tutorials/video/07-mibintro.zip, Posledný prístup 23.2. 2009.

[9] Structure of Management Information (SMI), Aiko Pras, [online], Prednáška dostupná na URL: http://www.simpleweb.org/tutorials/video/05-smi.zip, Posledný prístup 23.2. 2009.

[10] MIB-2, Aiko Pras, [online], Prednáška dostupná na URL: http://www.simpleweb.org/tutorials/video/08-mib2.zip, Posledný prístup 23.2. 2009.

[11] The Net-SNMP package, [online], Dokument dostupný na URL: http://www.net-snmp.org/wiki/index.php/Tutorials, Posledný prístup 1.5. 2009.

[12] MRTG: The Multi Router Traffic Grapher, [online], Dokument dostupný na URL: http://oss.oetiker.ch/mrtg/doc/index.en.html, Posledný prístup 21.3. 2009.

[13] SNMP++ - C++ based Application Programmers Interface for the simple Network Management Protocol, [online], Dokument dostupný na URL: http://man.chinaunix.net/develop/snmp_pp/index.html, Posledný prístup 20.4. 2009.

[14] Agent++ - An object Oriented Application Programmers Interface for Development of SNMP Agents Using C++ and SNMP++, [online], Dokument dostupný na URL: http://www.agentpp.com/doc2.x/agent++.html, Posledný prístup 15.3. 2009.

Príloha A Obsah elektronického média

Na elektronickom médiu sa nachádzajú adresáre:

/Dokumentacia

- Obsahuje dokumenty bakalárskeho projektu vo formáte PDF a DOC

/boost_1_38_0

- Obsahuje súbo boost_1_38_0.zip, ktorý je potrebné rozbaliť do koreňového adresára, zdrojové súbory sa musia nachádza v adresári /boost_1_38_0

/agent++

- Obsahuje knižnicu pre programovanie agentov

/libdes

- Obsahuje knižnicu pre šifrovanie

/snmp++

- Obsahuje knižnicu pre programovanie manažérov

/tinyxml

- Obsahuje knižnicu pre parsovanie XML dokumentov

/metricAgent

- Obsahuje zdrojové kódy a skripty pre preklad výsledného agenta

V adresári sa nachádzajú súbory:

metricAgent.sln

- slúži na preklad agenta vo Visual Studiu 2008

build.sh

- preloží agenta pod Linuxom

Nachádzajú sa tu podadresáre:

/metricAgent

- obsahuje zdrojové kódy agenta

/release

- obsahuje skompilovanú release verziu agenta

/debug

- obsahuje skompilovanú debug verziu agenta

V oboch adresároch /release aj /debug sa navyše nachádza konfiguračný súbor config.xml.

Príloha B Manuál

V nasledujúcej časti by som chcel vysvetliť základné možnosti konfigurácie, ktoré snmp agent ponúka. Po preložení pomocou skriptu build.sh alebo pomocou metricAgent.sln (visual studia) sa vygeneruje spustiteľný súbor s názvom *metricAgent,* ktorý je umiestnený v adresáre /metricAgent/release/ alebo /metricAgent/debug/. Jedinou možnosťou ako ovplyvniť správanie agenta je pomocou súboru config.xml, ktorý sa nachádza v spomínaných adresároch. Agentovi je možné predať názov tohto súboru cez príkazovú riadku pomocou parametra -f.

Príklad:

./metricAgent -f /home/janko/config.xml

V nasledujúcich podkapitolách by som chcel vysvetliť význam jednotlivých častí tohto konfiguračného súboru.

B.1 Základné nastavenia

Lokálne nastavenia agenta možno zmeniť pomocou atribútov, ktoré sú ohraničené tagom <localSettings>. Ich význam je nasledovný:

- requestPort - určuje port, na ktorom bude agent čakať na prichádzajúce požiadavky od manažérov
- trapInformPort – port, na ktorý mu môžu zasielať jednotlivé sieťové uzly informácie o zmene stavu na rozhraniach (momentálne sú podporované 2 typy pascí: linkUp, linkDown)
- persistentStoragePath – slúži na nastavenie adresára kam budú ukladané pomocné súbory
- mibRoot – vyjadruje OID, na ktorom bude umiestnená skupina metrík (metricGroup)

Príklad:

<localSettings requestPort="161" trapInformPort="162" persistentStoragePath="/home" mibRoot ="1.3.6.1.4.1.4976.6.2.1" />

B.2 Definícia sieťových uzlov a ich rozhraní

Nato aby sme mohli získavať atribúty zo sledovaných uzlov je potrebné pre ne nastaviť potrebný prístup. Tieto nastavenia zahŕňa tag <networkNodes>, v ktorom sa postupne nastavuje prístup k jednotlivým sieťovým uzlom a v rámci nich sa definujú rozhrania, ktoré chceme sledovať.
V jednotlivých <networkNode> tagoch sa musí nachádzať atribút:

- address – jednoznačne identifikuje sieťový uzol pomocou IP adresy alebo doménového mena a súčasne SNMP port, na ktorý je možné zasielať požiadavky. Adresa môže byť zadaná v tvare <doménové meno>:<port>, <ip adresa>:<port>, ak nie je zadaný berie sa do úvahy štandardný SNMP port 161.

V rámci tagu <networkNode> sa tu nachádzajú nasledujúce vnorené tagy:

- <snmp> - Určuje verziu protokolu pomocou, ktorého bude prebiehať komunikácia medzi manažérom a agentom. Verzia protokolu na nastavuje pomocou parametru:
    - protocolVersion - môže nadobúdať nasledujúcich hodnôt {snmpv1, snmpv2, snmpv3}

  Pod tagom <snmp> sa môžu vyskytovať nasledujúce tagy:

  - <access> - Obsahuje údaje potrebné pre prístup, počet parametrov závisí od voľby protokolu. Pri verzií protokolu SNMPv1, SNMPv2 je potrebné vyplniť atribút:
      - readCommunity - slúži ako primitívna forma ochrany, obyčajný textový reťazec, ktorý sa posiela po sieti v čitateľnej podobe

    Pri voľbe protokolu SNMPv3 sa v tagu access vyskytujú nasledujúce atribúty:
      - securityModel { “USM“, “v1“, “v2“}
      - securityLevel - určuje stupeň ochrany { “authPriv“ - autorizácia a šifrovanie, “authNoPriv” - iba autorizácia, “noAuthNoPriv“ - bez zabezpečenia }
      - contextName
      - contextEngine

    Pri voľbe bezpečnostného modelu v1 alebo v2 je potrebné vyplniť atribút securityName. Ak je zvolený mód USM je nevyhnutné pod <snmp> tag vložiť tag s názvom
  - <usm> - obsahuje nasledujúce bezpečnostné atribúty:
      - securityName - užívateľské meno
      - authPassword - heslo, ktoré sa použije pri autorizácií
      - privPassword - heslo, ktoré sa použije pri šifrovaní

- authProtocol - autorizačný protokol {HMACSHA, HMACMD5}
- privProtocol - šifrovací protokol {DES}

- <interfaces> - Ak má cieľový uzol niekoľko rozhraní, ktoré chceme sledovať, každý tag <interface> bude obsahovať atribút address, ktorý podľa ip adresy identifikuje dané rozhranie.

Príklad:

```
<networkNodes>
    <networkNode address="192.168.1.1" >
            <snmp protocolVersion="snmpv1" >
                    <access readCommunity="public" />
            </snmp>
    </networkNode>

    <networkNode address="192.168.1.2:161">
            <snmp protocolVersion="snmpv3" >
                    <access securityModel="USM" securityLevel="authPriv" contextName=""
                            contextEngineID="" />
                    <usm securityName="my_user" authPassword="my_password"
                            privPassword="my_password"  authProtocol="HMACMD5"
                            privProtocol="DES" />
            </snmp>
    </networkNode>

    <networkNode address="192.168.1.3" >
            <snmp protocolVersion="snmpv1" >
                    <access readCommunity="public" />
            </snmp>

            <interfaces>
                    <interface address="192.168.1.4" />
                    <interface address="192.168.1.5" />
            </interfaces>
    </networkNode>
</networkNodes>
```

B. 3 Definícia vzájomných prepojení uzlov

Keďže metriky sú definované medzi jednotlivými sieťovými uzlami je potreba nadefinovať ich vzájomné prepojenie. Nato slúži tag:

- <connections> ktorý obsahuje nasledujúce atribúty:
    - timeWindowAligned – určuje či bude časové okno, počas ktorého sa získavajú atribúty zarovnané {0, 1}
    - refreshTime - perióda v sekundách, za ktorú bude prebiehať posielanie požiadaviek o Dáta jednotlivým sieťovým uzlom a počítanie aktuálnych metrík
    - statisticTime - čas v sekundách počas, ktorého sa aktualizujú z daných metrík štatistiky

Pod tagom <connections> sa nachádza zoznam tagov typu:

- <connection> ktorý obsahuje nasledujúce atribúty:
    - direction - určuje smer, v ktorom chceme počítať metriku {0 - od zdroja k cieľu, 1 - od cieľa k zdroju, 2 - v oboch smeroch }
    - srcNodeAddr - ip adresa / doménové meno zdrojového uzlu
    - srcNodeIfAddr - ip adresa zdrojového rozhrania, ak má daný uzol len jedno rozhranie nemusí byť uvedená
    - destNodeAddr - cieľová ip adresa / doménové meno uzlu
    - destNodeIfAddr - cieľová ip adresa rozhrania, ak má cieľový uzol jedno rozhranie nemusí byť uvedená

Príklad:

```
<connections timeWindowAligned="1" refreshTime="300" statisticTime="3600">
        <connection direction="0" srcNodeAddr="192.168.1.1" destNodeAddr="192.168.1.3"
                    destNodeIfAddr="192.168.1.4" />
        <connection direction="0" srcNodeAddr="192.168.1.3" srcNodeIfAddr="192.168.1.5"
                    destNodeAddr="192.168.1.2 />
</connections>
```

B. 4 Definícia prístupových práv do MIB

Keďže agent má hlavne poskytovať metriky a ich štatistiky ostatným manažérom alebo nástrojom je potrebné pre nich nadefinovať prístupové práva. Definícia užívateľov sa nachádza pod tagom <usm> pričom každý užívateľ je reprezentovaný jednou položkou <user> s danými atribútmi:

- userName - meno užívateľa
- authProtocol - metóda, ktorá sa použije pri autorizácií {HMACMD5, HMACSHA}
- privProtocol - metóda pre šifrovanie momentálne len {DES}
- authPassword - heslo, ktoré sa použije pri autorizácií
- privPassword - heslo, ktoré sa použije pri šifrovaní

Príklad:

```
<usm>
    <user userName="MD5" authProtocol="HMACMD5" privProtocol=""
            authPassword="MD5UserAuthPassword" privPassword="" />
    <user userName="SHA" authProtocol="HMACSHA" privProtocol=""
            authPassword="SHAUserAuthPassword" privPassword="" />
    <user userName="MD5DES" authProtocol="HMACMD5" privProtocol="DES"
            authPassword="MD5DESUserAuthPassword"
            privPassword="MD5DESUserPrivPassword" />
    <user userName="SHADES" authProtocol="HMACSHA" privProtocol="DES"
            authPassword="SHADESUserAuthPassword"
            privPassword="SHADESUserPrivPassword" />
</usm>
```

Na obmedzenie prístupu jednotlivých užívateľov je možné využiť prístupu VACM (View-based access control model - prístupový model založený na pohľadoch). Jednotlivých užívateľov je možné priradiť do skupín, ktorým sa priradia pohľady na Dátabázu. Pohľady určujú, ktoré časti MIB budú pre užívateľov viditeľné. Celý VACM model je zahrnutý pod tagom <vacm> s nasledujúcimi elementami:

- <contexts> - definuje jednotlivé kontexty
- <views> - obsahuje definíciu pohľadov v tagoch <view> s nasledujúcimi atribútmi:
    - viewName - meno pohľadu
    - oid - identifikátor časti MIB stromu
    - viewIncluded - určuje či má byť časť MIB stromu zahrnutá v pohľade alebo vylúčená {1, 2}

- <groups> - definícia jednotlivých skupín v tagoch <group> s atribútmi:
  - groupName - názov skupiny
  - context - názov kontextu
  - securityModel - model zabezpečenia {v1, v2, USM}
  - securityLevel - level ochrany {“authPriv”, “AuthNoPriv”, “NoAuthNoPriv”}
  - contextMatch - zhoda s kontextom {“match_exact”, “match_prefix”}
  - readView – pohľad na čítanie
  - writeView – pohľad na zápis
  - notifyView – pohľad pri upozorneniach
- <securitytogroup> - umožňuje priradiť skupiny jednotlivým užívateľom, ak je špecifikovaný model bezpečnosti {v1, v2} nie je potrebná zhoda s existujúcim užívateľom definovaným cez <usm>. Obsahuje zoznam <entry> elementov s danými atribútmi:
  - securityModel - model bezpečnosti {USM, v1, v2}
  - securityName - v prípade USM sa jedná o názov užívateľa, ak je zvolený v1 alebo v2 model jedná sa o komunitný reťazec
  - groupName – názov skupiny, do ktorej pridáme daného užívateľa

Príklad:

```
<vacm>
    <contexts>
        <context>""</context>
        <context>other</context>
    </contexts>

    <views>
        <view viewName="v1ReadView" oid="1.3" mask="" viewIncluded="1"
            storageType="nonVolatile" />
        <view viewName="v1WriteView" oid="1.3" mask="" viewIncluded="1"
            storageType="nonVolatile" />
        <view viewName="v1NotifyView" oid="1.3" mask="" viewIncluded="1"
            storageType="nonVolatile" />
    </views>

    <groups>
        <group groupName="v1v2group" context="" securityModel="v2"
            securityLevel="noAuthNoPriv"
```

```
                contextMatch="match_exact" readView="v1ReadView"
                writeView="v1WriteView" notifyView="v1NotifyView"
                storageType="nonVolatile" />

            <group groupName="v1v2group" context="" securityModel="v1"
                securityLevel="noAuthNoPriv"
                contextMatch="match_exact" readView="v1ReadView"
                writeView="v1WriteView" notifyView="v1NotifyView"
                storageType="nonVolatile" />
        </groups>

        <securitytogroup>
            <entry securityModel="v1" securityName="public" groupName="v1v2group"
                storageType="volatile" />
            <entry securityModel="v2" securityName="public" groupName="v1v2group"
                storageType="volatile" />
        </securitytogroup>
</vacm>
```

Príloha C Elektronický nosič