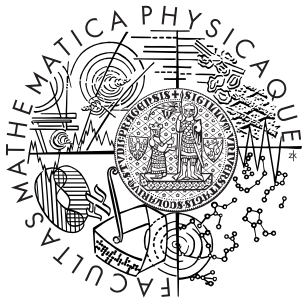
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

BAKALÁRSKA PRÁCA



František Kačmarik

SNMP Trap generátor

Ústav formální a aplikované lingvistiky

Vedúci bakalárskej práce: Mgr. David Kolovratník
Študijný program: Informatika, správa počítačových systémů

2010

Chcel by som sa poďakovať Mgr. Dávidovi Kolovratníkovi za cenné pripomienky, trpezlivosť a pomoc pri zostavovaní práce.

Prehlasujem, že som svoju bakalársku prácu napísal samostatne a výhradne s použitím citovaných prameňov. Súhlasím so zapožičiavaním práce a s jej zverejňovaním.

V Prahe dňa 3. augusta 2010 František Kačmarik

Obsah

Názov práce: SNMP Trap generátor
Autor: František Kačmarik
Katedra (ústav): Ústav formální a aplikované lingvistiky
Vedúci bakalárskej práce: Mgr. David Kolovratník
E-mail vedúceho: *david@kolovratnik.net*

Abstrakt: Bakalárska práca nadväzuje na ročníkový projekt s rovnakým názvom SNMP Trap generátor. Jeho cieľom je vytvorenie programu schopného zachytiť a prehrať udalosti hlásené protokolom SNMP a tiež užívateľského rozhrania pre správu a tvorbu takýchto udalostí.

Bakalárska práca obsahuje zhodnotenie dosiahnutia cieľov projektu (vhodnosť reprezentácie udalostí, možnosti ich prehrávania), prípadné návrhy na možné vylepšenia a tiež praktické skúsenosti s použitím programu k ladeniu softvéru obsluhujúceho vzniknuté udalosti.

Popisuje scenáre pre simuláciu neštandardných situácií nastávajúcich na monitorovanom zariadení. To umožňuje ľahko testovať správanie sa obslužného softvéru.

Taktiež prináša popis princípov a najdôležitejších častí použitých či nadväzujúcich aplikačných protokolov.

Kľúčové slová: SNMP, trap, generátor

Title: SNMP Trap replayer
Author: František Kačmarik
Department: Institute of Formal and Applied Linguistics
Supervisor: Mgr. David Kolovratník
Supervisor's e-mail address: *david@kolovratnik.net*

Abstract: The bachelor's thesis is a continuation of an Academic Year Project named SNMP Trap replayer. Its goal has been to develop software that is able to catch and replay notifications thru SNMP protocol, a user interface for administering and creating these notifications.

The bachelor's thesis contains an evaluation of achieved goals (such as event representation, ability to replay them), opportunities for improvement as well as practical applications of this program in regard of testing management software for handling incoming events.

It includes test cases for testing of non-standard situations that occur on a monitored hardware making it easier to test management software.

The bachelor's thesis also includes a specification of fundamental principles and the most important parts of used and follow-up application protocols.

Keywords: SNMP, trap, generator, replayer

Úvod

Inšpiráciou ku vzniku SNMP Trap generátora bol podnet z praxe, kde sme sa stretli s potrebou neustáleho opakovania testov v spojitosti so skutočným zariadením, ktoré však častokrát nebolo k dispozícii.

SNMP Trap generátor bol vyvinutý ako pomôcka pre testovanie a vývoj aplikácii monitorujúcich stav serveru, prípadne aj iných zariadení s využitím protokolu SNMP (Simple Network Management Protocol). Je to jednoduchá náhrada skutočného servera, ktorá je schopná reprodukovať správy, ktoré sú za normálnych okolností posielané serverom pri rôznych, hlavne chybových situáciách. Tiež je schopná na prichádzajúce požiadavky odpovedať.

Najväčšou výhodou SNMP Trap generátora je zníženie potreby prístupu k reálnemu zariadeniu a obmedzenie jeho využívania na čo najkratší možný čas. Pri testovaní i vývoji monitorovacieho software je potrebné zdieľať ten istý hardware medzi programátorom i testerom, tiež i pri súbežnom testovaní či vývoji viacerých aplikácii, čo však prináša veľmi vysoké požiadavky na množstvo potrebných zariadení, ich umiestnenie, náklady na prevádzku, atď. Niekedy je server zdieľaný medzi viacerými týmami, napríklad kvôli jeho vysokej cene, čo značne znižuje čas, cez ktorý je možné server využívať.

Práve v týchto prípadoch má SNMP Trap generátor svoje uplatnenie, keďže dokáže obmedziť potrebu prístupu k skutočnému serveru a zjednodušiť, zľahčiť a teda aj urýchliť celý vývojový proces. Vývojár v konečnom dôsledku vôbec nemá potrebu využívať reálny hardware, stačí iba nakonfigurovať SNMP Trap generátor a celý vývoj môže prebiehať „lokálne“ na ľubovoľnom počítači.

SNMP Trap generátor je možné využívať ako „command-line“ aplikáciu, teda využívať jeho funkcionalitu priamo z príkazového riadku, alebo ako serverovú aplikáciu prístupnú cez webový prehliadač. Pre jeho fungovanie nie je potrebná žiadna dodatočná inštalácia, potrebný je iba nainštalovaný Python v požadovanej verzii. Webové rozhranie ponúka o trochu viac možností ako command-line rozhranie. Hlavnou výhodou je možnosť ľahkej a jednoduchej editácie uložených dát a taktiež vytváranie vlastných testovacích dát. Táto možnosť je určená pre používanie vývojárom i testerom a umožňuje navodiť situácie, ktoré by bolo v skutočnom prostredí obtiažne vytvoriť. Command-line funkcionalita slúži hlavne na urýchlenie a automatizáciu testovania.

SNMP Trap generátor je pomenovaný podľa časti protokolu SNMP, teda presnejšie podľa typu jedného zo SNMP PDU (Protocol Data Unit), ktorá sa nazýva Trap-PDU, odtiaľ pomenovanie SNMP Trap generátor. Je to označenie pre asynchrónnu notifikáciu smerom od monitorovaného zariadenia, zvyčajne sa toto zariadenie nazýva agent, smerom ku management softwaru, ktorý túto správu vyhodnotí a podľa jej vážnosti o nej informuje. Management software je obyčajne označovaný ako manager alebo NMS (Network

Management System). Je to spôsob, akým o sebe agent dáva vedieť, že sa dostal do nejakého neočakávaného stavu, že vznikol nejaký problém. Správa vo svojom tele obsahuje popis problému, ktorý daný trap vyvolal.

Hlavný prínos SNMP Trap generátora spočíva práve v tom, že dokáže tieto posielané trapy odchytávať a následne reprodukovať. Dokáže však pracovať aj s ostatnými časťami SNMP protokolu, ako SetRequest či GetRequest. Dokáže vystupovať i v role agenta, v tomto režime vie odpovedať na SetRequest, GetRequest či SNMP walk rovnakým spôsobom ako skutočný server. Podporuje protokol SNMP verzie 1 a tiež štandardizovanú verziu 2c. Druhá verzia SNMP protokolu sa oproti jednotnej prvej verzii rozštiepila na viaceré vývojové vetvy, ako napríklad SNMPv2u, SNMPv2p, SNMPv2c. O týchto rozdielnych verziách detailnejšie hovorí kapitola 1.3. Najnovšou verziou protokolu SNMP je verzia 3, ktorá znova priniesla zjednotenie vývojových vetiev SNMP protokolu. Túto verziu SNMP Trap generátor zatiaľ nepodporuje hlavne kvôli zvýšenej úrovni bezpečnosti. O dôvodoch nevyužívania SNMPv3 hovorí kapitola 1.4.4. Protokol SNMPv3 zatiaľ nepodporujú všetky zariadenia a ani v praxi nie je často používaný. SNMPv1 spolu so SNMPv2c patria medzi najviac rozšírené verzie, ktoré sa reálne používajú a preto SNMP Trap generátor implementuje práve tieto verzie SNMP protokolu.

V tejto práci ďalej popíšeme samotný SNMP protokol i protokoly na neho bezprostredne nadväzujúce, detailnejšie predstavíme SNMP Trap generátor, vysvetlíme princíp jeho fungovania, naznačíme možné scenáre použitia a tiež skúsenosti s jeho použitím pri testovaní a ladení software, ktorý je určený na správu zariadení.

Kapitola 1

SNMP

Simple Network Management Protocol (SNMP), vo voľnom preklade jednoduchý protokol na správu siete, je štandardizovaný protokol aplikačnej vrstvy, využíva UDP (User Datagram Protocol), ktorý je časťou rodiny protokolov IP. Využíva sa hlavne na správu rôznych sieťovými zariadení ako napríklad routre, servery, tlačiarne, atď. Slúži na získavanie stavových informácií o zariadeniach, nastavovanie parametrov, atribútov, monitorovanie záťaže systému, času nepretržitého behu systému, zber rôznych iných dát zo sieťových rozhraní, stav teplotných senzorov, otáčok chladiacich ventilátorov či hodnôt napätí, atď.

1.1 Komponenty SNMP

Základnými komponentami pri spravovaní zariadení pomocou protokolu SNMP sú: spravované zariadenie (*managed device*), *agent* a nejaký systém na správu (*Network Management System (NMS)*).

- *Managed device*: je samotné zariadenie, o ktorom je potrebné získavať informácie. Je spojené s agentom a komunikuje s management systémom výhradne skrz agenta.
- *Agent*: môže byť implementovaný v špeciálnej časti zariadenia určenej na jeho správu alebo byť súčasťou operačného systému. Je to program bežiaci na spravovanom zariadení, ktorý monitoruje stav, získava a uchováva informácie o zariadení. Tieto dáta sprístupňuje skrz SNMP protokol Network Management systému (NMS). Realizuje všetku komunikáciu, čaká na podnety od NMS a odpovedá mu na ne, obvykle neiniciuje kontakt s NMS. Iba v prípade vzniku neštandardnej situácie sám zasiela NMS chybové hlásenie. Agent môže mať viacero podriadených agentov, od ktorých môže získavať informácie. Títo podriadení agenti môžu získavať informácie z rôznych častí zariadenia alebo aj z rôznych zariadení.
- *Network Management System*: software, ktorý monitoruje, vyhodnocuje správanie sa monitorovaných zariadení a informuje o prípadných vzniknutých neštandardných situáciách.

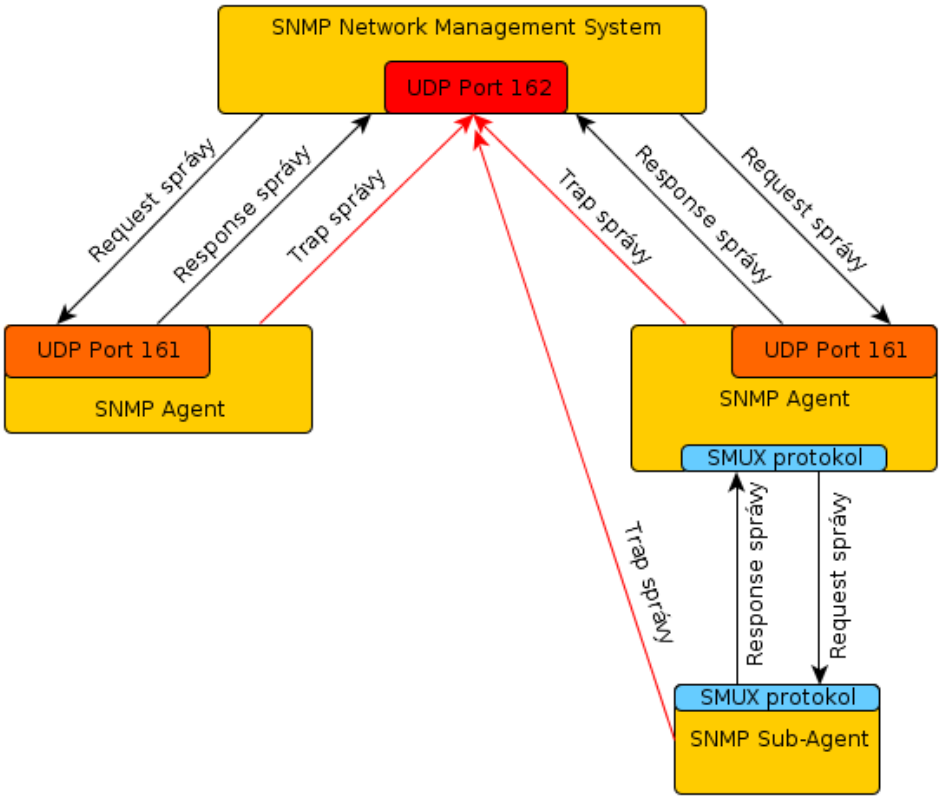
1.2 Komunikácia cez SNMP

Monitoring siete prebieha tak, že NMS v nejakých pravidelných časových intervaloch zbiera informácie z monitorovaných zariadení a vyhodnocuje ich. NMS pošle požiadavok, zvyčajne formou GetRequest správy, na ktorý agent odpovedá pomocou GetResponse správy.

Štandardne agent sám neiniciuje komunikáciu s NMS. V prípade vzniku neštandardnej situácie dokáže agent okamžite upozorniť NMS formou zaslania správy, tzv. SNMP trap. Agent však musí byť správne nakonfigurovaný, teda vedieť kam má poslať správu, trap.

Štandardne a najčastejšie komunikácia medzi agentom a NMS prebieha pomocou UDP na porte 161, ktorý štandardne používa agent. SNMP trap je posielaný na port 162, kde štandardne NMS očakáva trapy. Využitie UDP znamená efektívnosť a jednoduchosť zasielania správ, zároveň však negarantuje prijatie správy príjemcom. Zodpovednosť za doručenie nesie zariadenie, ktoré správu posiela. NMS zvyčajne používa časovač na meranie času uplynutého od odoslania poslednej správy. Ak nedorazí odpoveď, správa je poslaná opäť. Ak odpoveď ani po viacnásobnom odoslaní a uplynutí stanoveného časového limitu nedorazí, aj to je dôvod predpokladať, že sa monitorované zariadenie dostalo do neštandardného stavu alebo je z nejakého dôvodu nedostupné. Podobný problém je so SNMP trap správami. Keďže na ne agent po odoslaní neočakáva odpoveď, môžu sa veľmi jednoducho stratiť a neinformovať NMS o vzniknutej situácii. Preto je dôležité v nastavení NMS použiť rozumný časový interval na zisťovanie stavu zariadení, nespoliehať sa iba na trapy.

Agent dokáže komunikovať so svojimi podriadenými agentmi. Na túto komunikáciu využíva SNMP multiplexing protokol (SMUX). Definuje komunikáciu medzi SNMP agentom a inými podriadenými agentmi, procesmi.



Obrázok 1.1: Komunikácia protokolom SNMP.

1.3 História a vývoj SNMP

SNMP protokol sa od svojho zrodu vyvinul až do podoby štandardu pre monitorovanie a správu zariadení a taktiež sa stal jedným zo štandardných nástrojov na zisťovanie sieťovej topológie.

Začal sa vyvíjať v osemdesiatych rokoch dvadsiateho storočia, kedy vznikla potreba vytvoriť nejaký jednotný štandard pre správu sieťových zariadení. V tomto období existovali tri štandardy: High-level Entity Management System (HEMS) / High-level Entity Management Protocol (HEMP), definovaný v RFC 1021 až 1024, Simple Gateway Monitoring Protocol (SGMP), definovaný RFC 1028 a Common Management Information Protocol (CMIP), ktorý bol súčasťou Open Systems Interconnection (OSI)[1] modelu. V roku 1988 Internet Engineering Task Force (IETF) vydala RFC 1052 pod názvom IAB Recommendations for the Development of Internet Network Management Standards. Nebol to štandard, skôr snaha o vytvorenie jednotného prístupu a nového štandardu. Výsledkom tejto snahy bol vznik štandardu nazývaného Simple Network Management Protocol (SNMP), ktorý vychádzal zo SGMP.

Novovzniknutý protokol zo SGMP prevzal napríklad základné operácie, ktoré sú doteraz súčasťou SNMP. Jeho vývoj pripadol SNMP Working Group. SNMP definovalo spôsob, akým by sa dali dostatočne univerzálne definovať informácie o spravovanom zariadení, pracovať s nimi vo forme objektov a tieto objekty prenášať medzi spravovaným zariadením a zariadením slúžiacim na jeho správu. Zároveň samotný protokol nedefinuje spravovateľné objekty, zaoberá sa iba ich prenosom. Objekty sú definované pomocou Structure of Management Information (SMI) a Management Information Base (MIB), ktoré sú bližšie popísané v kapitole 1.5. SNMP protokol preto pozostáva iba z niekoľkých operácií a je relatívne jednoduchý na implementáciu.

V roku 1988 bol vydaný SNMP version 1, skrátene SNMPv1, definovaný RFC 1067 (neskôr upravený v RFC 1098 a RFC 1157). Dočkal sa veľkého úspechu a širokého použitia. Doteraz patrí medzi jednu z najrozšírenejších verzií SNMP. Jeho nevýhodou i výhodou zároveň je nízka úroveň ochrany, zabezpečená iba reťazcom, ktorý je možné ľahko odchytávať.

SNMP verzia 2 priniesla dva nové typy správ, snažila sa vylepšiť bezpečnosť, nebol však všeobecne prijatý jednotný prístup, preto vznikli viaceré vetvy SNMPv2.

SNMPv2p (Party-based) sa snažila o vylepšenie protokolu ale zároveň chcela ponechať zabezpečenie formou stringov, nikdy sa nestala štandardom.

SNMPv2c sa veľmi podobá SNMPv2p, vychádzala však zo štandardu definovaného v RFC 1901, ktorý má iba experimentálny status. Implementuje SNMPv2, ale využíva ten istý bezpečnostný model ako SNMPv1. Aj napriek tomu však dosiahla úspech v komerčnej sfére a prakticky sa stala štandardom.

Ďalšou vetvou je SNMPv2u (User-based), v tejto verzii bola bezpečnosť založená na použití overovania užívateľov namiesto overovania reťazcov. Stále je považovaná za experimentálnu vetvu.

Komerčnú vetvu reprezentuje SNMPv2*. Tá kombinovala SNMPv2p a SNMPv2u. Nikdy sa však nedočkala masívnejšieho používania a nebola ani štandardizovaná.

Spomedzi týchto vetiev sa SNMP Trap Generátor bude zaoberať iba štandardizovaným a pomerne dosť rozšíreným SNMPv2c protokolom (RFC 1902 až 1908).

V roku 1996 až 1998 bola vyvinutá SNMPv3, bola jednotnou odpoveďou na roztrieštenú verziu 2. Priniesla hlavne jednotnosť a doplnkové rozšírenia SNMP protokolu.

Formalizovala prístup k bezpečnosti, priniesla autentifikáciu (MD5 a SHA) a šifrovanie (DES). Formát SNMPv3 je popísaný v RFC 3412.

SNMPv1, SNMPv2 a SNMPv3 nie sú vzájomne kompatibilné.

1.4 Bezpečnosť a SNMP

1.4.1 Bezpečnosť v SNMPv1

SNMPv1 je známa pomerne slabým zabezpečením. Pri jej vzniku nebol kladený prílišný dôraz na bezpečnosť, no aj napriek tomu je stále veľmi rozšírenou verziou, ktorá sa používa v praxi. SNMPv1 je chránená iba pomocou „community string“, čo je reťazec znakov, heslo. Toto heslo sa dá jednoduchým spôsobom odchytiť. V správe sa nachádza ako obyčajný text. Autentifikácia prebieha iba na základe porovnania tohoto hesla s nastaveniami SNMP agenta. Samotná správa nie je nijako šifrovaná. Nebezpečenstvo je najväčšie, ak sa takéto správy pohybujú po nebezpečných verejných sieťach. Uvedieme niekoľko príkladov bezpečnostných rizík:

- Útočník vystupuje v roli iného zariadenia. Vykonáva činnosť, ktorú by sa normálnych okolností mohlo vykonávať iba zariadenie na to určené. Napríklad sa môže vydávať za NMS a môže vykonávať všetko čo mohol vykonávať skutočný NMS. Takýto typ útoku je anglicky označovaný „spoofing“.

- Zmena informácií v prenášanej správe. Útočník môže odchytiť správu, pozmeniť ju a poslať ju pozmenenú znova. Útočník môže odchytiť správu a pozmeniť v nej celú PDU, pričom zachová nezmenené ostatné časti správy, vrátane tých ktoré sa týkajú bezpečnosti. Niekedy je dokonca možné spustiť vlastný kód na spravovanom zariadení.

- Útočník dokáže odchytiť viaceré po sebe idúce správy a následne ich pozmeniť, zameniť ich poradie alebo znova ich odoslať v jemu vyhovujúcom čase. Pri tomto type útoku sa využíva to, že SNMP je orientovaný na použitie s nespoľahlivými protokolmi, ktoré nezaručujú poradie prijatia správ. Útočník môže napríklad pomocou takýchto odchytených správ vypnúť zariadenie alebo zmeniť jeho nastavenie. Takéto správanie môže viesť k útoku typu Denial of Service (DoS), kedy efektívne zabráni používaniu zariadenia.

- Nepríjemnou je tiež možnosť úniku citlivých informácií. Niektoré SNMP správy môžu obsahovať informácie o sieti samotnej a tiež o zariadeniach v nej. Napríklad IP adresy, ktoré sa dajú následne využiť pri ďalších útokoch.

- Community string je ako heslo častokrát zvolené nevhodne. Administrátori často dostatočne dobre nenakonfigurujú SNMP agentov alebo použijú prednastavené community string. Prednastavené heslá sú dobre známe a typické pre každého výrobcu zariadení. Najčastejšie sa vyskytujúce sú public, private, admin, default, atď.

Kvôli nízkej úrovni bezpečnosti SNMPv1 sa administrátori chránia zvyčajne zakázaním prijímania SetRequest správ. Agenti teda slúžia iba na monitorovanie zariadení.

1.4.2 Bezpečnosť v SNMPv2c

SNMP verzie 2 sa snažil poskytnúť vyššiu bezpečnosť ako predchádzajúca verzia SNMP protokolu. Vývoj však nebol jednotný a SNMP protokol verzie 2c sa spomedzi ostatných vývojových vetiev ako jediný stal štandardom. Nepriniesol však žiadnu zmenu čo sa týka bezpečnosti, využíval opäť autentifikáciu pomocou community string, presne ako v staršom SNMP protokole verzie 1. Protokoly, ktoré sa snažili implementovať nový bezpečnostný model, ako napríklad SNMPv2u a SNMPv2*, sa nedočkali masívnejšieho použitia, ale novovzniknuté bezpečnostné modely stali sa základom pre SNMPv3.

1.4.3 Bezpečnosť v SNMPv3

Protokol SNMP verzie 3 už zabezpečuje bezpečnú výmenu informácií medzi agentom a NMS. Nadväzuje na SNMPv2u a SNMPv2*. Na overovanie totožnosti užívateľa používa „group-based“ model, ktorý umožňuje rôznym užívateľom pristupovať k SNMP agentovi s rôznymi prístupovými právami. Taktiež dáta prenášané v správe sú šifrované a môžu byť zasielané aj po nezabezpečených sieťach. PDU, ktorá sa nachádza v tejto verzii je SNMPv2-PDU, ktorá je rozšírená o časti týkajúce sa šifrovania a bezpečnosti.

SNMPv3 používa User-Based Security Model (USM). Je to bezpečnostný model, ktorý pracuje s dvoma šifrovacími kľúčmi. Jeden sa využíva na šifrovanie a druhý na autentifikáciu. Tieto kľúče nie sú uložené v MIB, nie sú teda ani prístupné skrz SNMP.

Na autentifikáciu sa v SNMPv3 využíva buď hašovacia funkcia MD5 alebo SHA-1, ktorých vstupom je správa, ktorá sa bude odosielať a autentifikačný kľúč. Ich výstup je skrátený na pevne stanovenú dĺžku a nazýva sa Message Authentication Code (MAC). Tento MAC je priložený do správy. Pri prijatí správy príjemca spočíta zo správy MAC a porovná ho s MAC uloženým v prijatej správe. Ak sa líšia, správu zahodí. Takýto postup zaručuje, že správa nebola zmenená počas jej prenosu a zároveň že nebola poslaná útočníkom, pretože útočník nemá k dispozícii tajný autentifikačný kľúč na vytvorenie MAC.

SNMPv3 je taktiež odolný voči útokom, kedy útočník prehráva predom odchytené správy, napríklad odchytené správy na vypnutie zariadenia. Implementuje mechanizmus, ktorý vyžaduje aby správa bola prijatá v určitom čase po jej odoslaní. Zariadenie si udržuje odhad o čase ostatných zariadení, s ktorými komunikuje. Tieto odhady sú vkladané do každej odchádzajúcej správy. Príjemca potom na základe tohoto času určí, či správu prijme alebo zahodí.

Šifrovanie obsahu správy je zabezpečené pomocou Data Encryption Standard (DES). Použitie symetrického šifrovania však vytvára malý problém, a to v správe tajných kľúčov. Ak majú všetky zariadenia rozdielne kľúče, NMS musí obsahovať všetky tajné kľúče, čo môže časom spôsobovať neprehľadnosť a problém pri správe. Pri použití spoločného kľúča má zasa útočník pri získaní tohoto kľúča prístup k celej sieti. Tento problém sa snaží riešiť technika nazývaná „Key Localization“. Bližší popis práce s týmito kľúčmi je popísaný v RFC 2274.

SNMPv3 môže využívať aj iný spôsob overovania prístupu k informáciám v MIB, a to View-Based Access Control (VACM), ktorý je popísaný v RFC 2575. Ten sa líši od USM modelu, ktorý definuje prístup pre každého užívateľa zvlášť. VACM používa začlenenie užívateľov do skupín a prideľuje rôzne prístupové práva pre rôzne skupiny.

1.4.4 SNMP Trap generátor a verzie SNMP

SNMP Trap generátor pracuje so SNMP protokolom vo verzii 1 a 2c. Pri simulovaní skutočného hardware využíva nedostatky v bezpečnosti SNMP protokolu verzie 1 i 2c popísane v kapitolách 1.4.1 a 1.4.2. Iba vďaka nim je schopný vydávať sa za reálne zariadenie, správať sa rovnako ako simulovaný hardware, odchytávať a neskôr prehrávať a taktiež meniť odchytené správy. Preto ho je možné využiť pri testovaní a vývoji NMS.

SNMP Trap generátor nepracuje so SNMP protokolom verzie 3 práve kvôli jeho zvýšenej úrovni bezpečnosti, ktorej sa detailnejšie venovala kapitola 1.4.3. Keďže je bezpečný voči úprave správ, vydávaní sa za iné zariadenie i voči opätovnému preposielaniu odchytených správ, SNMP Trap generátor nedokáže simulovať zariadenie komunikujúce pomocou tohoto protokolu tak ako tomu bolo v prípade predchádzajúcich verzií SNMP protokolu. Dokázal by síce komunikovať pomocou tejto verzie protokolu, ale už by verne nesimuloval nejaký skutočný hardware, čo je jedným z hlavných cieľov SNMP Trap generátora.

1.5 MIB, SMI, ASN.1

SNMP nie je protokol, o ktorom by sa dalo povedať, že pozostáva priamo zo sady príkazov, napríklad na zápis, čítanie, nastavenie nejakého parametru, vykonanie nejakej akcie. V takýchto typoch protokolov sú presne stanovené príkazy pre získanie jednotlivých informácií, napríklad na zistenie, ako dlho už dané zariadenie pracuje bude existovať zvláštny príkaz, na vypnutie zariadenia iný príkaz. Takáto sada príkazov môže byť značne rozsiahla v závislosti na možnostiach zariadenia a zároveň i na množstve zariadení, ktoré môžu protokol využívať. Protokol je tým pádom veľmi úzko naviazaný na hardware, pri vzniku nového zariadenia musí byť prispôsobený protokol a tiež pri zmene takéhoto protokolu je ovplyvnené množstvo hardware.

SNMP protokol sa dá označiť za „information-oriented“ protokol. Pracuje s objektami, teda premennými, do ktorých je možné zapisovať alebo z nich získavať informácie. V tomto prípade by sa napríklad na zistenie času, ako dlho už zariadenie pracuje použil obsah premennej obsahujúcej túto informáciu. SNMP protokolu teda stačí iba niekoľko málo základných operácií, ako napríklad načítaj obsah premennej alebo zapíš do premennej. Množstvo informácií o zariadeniach je možné rozširovať bez nutnosti akéhokoľvek zásahu do protokolu.

Objekty sa rôzne líšia v závislosti na zariadení. Objekty používané SNMP protokolom sú špecifikované v Management Information Base (MIB). V SNMPv1 bol definovaný jediný štandard, ktorý popisoval celý MIB. To však už neplatí. Existujú prídavné moduly, rozšírenia, ktoré definujú sadu objektov pre dané zariadenie. Zvyčajne sú poskytované výrobcom spolu so zariadením.

Univerzálnosť MIB objektov zabezpečuje štandard Structure of Management Information (SMI). Tento štandard definuje spôsob, akým je celý MIB modul konštruovaný a ako sú popisované jednotlivé objekty. Vytvára tiež hierarchickú štruktúru podobajúcu sa stromu na pomenovanie objektov, aby mohli byť jednoducho identifikovateľné a adresovateľné. V SMI, sú MIB objekty presne špecifikované pomocou jazyka slúžiaceho na definovanie dát, ich štruktúry, kódovania a dekódovania, International Organization for Standardization (ISO)[2] štandardu Abstract Syntax Notation 1 (ASN.1).

SNMP protokol prenáša informácie o stave zariadení, MIB definuje aké informácie sa dajú prenášať a SMI popisuje ako sú definované jednotlivé objekty MIB. SNMP protokol

taktiež nedefinuje žiadne informácie, aké by mal spravovaný systém ponúknuť alebo aké by si mal NMS vyžadovať. SMI existuje v dvoch verziách: SMIv1 definovaná RFC 1155, slúžiaca pre SNMPv1 a pre novšie verzie SNMP vznikla SMIv2, pôvodne definovaná v RFC 1442 a neskôr upravená RFC 2578.

1.5.1 MIB objekty

SMIv1 a SMIv2 si sú veľmi podobné v definícii objektov, SMIv2 poskytuje možnosť asociovať viac informácií s objektom. Základná štruktúra objektu obsahuje: *Object Name*, *Syntax*, *Access*, *Status*, *Definition*, *Optional Characteristics*.

- *Object name*: obsahuje identifikátor objektu. Pomenovanie objektov je bližšie vysvetlené v kapitole 1.5.2

- *Syntax*: definuje dátový typ objektu, teda jeho štruktúru. Môžu to byť buď regulárne dátové typy definované ASN.1, alebo tabuľka, zoznam z viacerých základných dátových typov. V SMIv2 boli názvy niektorých typov rozšírené o číslicu 32, aby z nich bola jasná ich bitová veľkosť. Dátové typy sú zobrazené v nasledujúcej tabuľke 1.1.

- *Access*: v SMIv2 nazývaný *Max-Access*, je položka špecifikujúca použitie a prístup k objektu. V SMIv1 môže táto položka nadobúdať štyri hodnoty: read-only, read-write, write-only a not-accessible. V SMIv2 je rozšírená na päť hodnôt: read-create, read-write, read-only, accessible-for-notify a not-accessible.

- *Status*: popisuje stav definície. V SMIv1 môže nadobúdať tri hodnoty: mandatory, optional a obsolete, v SMIv2 už iba dve: current a deprecated.

- *Definition*: v SMIv2 nazývaný *Description*, obsahuje textový popis objektu.

- *Optional Characteristics*: obsahuje rozširujúce informácie týkajúce sa objektu, ako napríklad referencie k doplnkovým dokumentom alebo iným relevantným informáciám týkajúcich sa objektu alebo index, ak sa jedná o komplexnejší objekt pozostávajúci z viacerých objektov.

Ukážka skutočnej definície SMIv2 objektu zo SNMPv2-MIB popisujúce objekt, ktorý obsahuje čas od posledného spustenia entity komunikujúcej SNMP protokolom:

```
sysUpTime OBJECT-TYPE
    SYNTAX      TimeTicks
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
            "The time (in hundredths of a second) since the
            network management portion of the system was last
            re-initialized."
    ::= { system 3 }
```

Ukážka definície SMIv2 objektu.

Nasledujúca tabuľka popisuje dátové typy používané v MIB:

Dátový typ	*SMIv1*	*SMIv2*	*Popis*
Integer / Integer32	x	x	32 bitové celé číslo so znamienkom, v rozsahu od -2,147,483,648 do +2,147,483,647, môže byť tiež použitý aj ako výčtový typ, kde jednotlivé čísla reprezentujú iné konštanty.
Octet String	x	x	Textový alebo binárny reťazec premenlivej dĺžky.
Null	x	-	Typ pre označenie prázdnej hodnoty.
Bits	-	x	Vymenované pomenované bity. Používa sa na to, aby sa skupina bitov dala použiť ako dátový typ.
Unsigned	-	x	32 bitové bezznamienkové celé číslo, od 0 do 4,294,967,295.
Network Address / IpAddress	x	x	IP adresa, kódovaná ako štvorbajtový reťazec oktetov.
Counter / Counter32	x	x	32 bitové bezznamienkové celé číslo, od 0 do 4,294,967,295.
Gauge / Gauge32	x	x	32 bitové bezznamienkové celé číslo, hodnoty od 0 do 4,294,967,295. Zvyčajne má asociované minimum a maximum indikujúce jeho rozsah.
TimeTicks	x	x	32 bitové bezznamienkové celé číslo indikujúce počet stotín sekúnd od nejakého momentu.
Opaque	x	x	Dáta používajúce vlastnú ASN.1 syntax, ktoré sú zasielané medzi zariadeniami bez toho, aby boli interpretované. K dátam sa stavia ako ku čiernej skrinke, nezaujíma ho ich obsah.
Counter64	-	x	Podobne ako Counter32, akurát má dĺžku 64 bitov. Môže nadobúdať hodnoty od 0 do 18,446,744,073,709,551,615.

Tabuľka 1.1: Dátové typy využívané MIB objektami.

1.5.2 Hierarchická štruktúra a popis MIB objektov

Každý objekt má svoje meno a identifikátor. Meno je textový názov objektu. Slúži na lepšiu predstavu o objekte, jeho účele. S veľkým množstvom objektov je potrebné využívať štruktúru, ktorá by bola dostatočne flexibilná, dalo sa v nej ľahko vyhľadávať a zároveň by bola ľahko rozšíriteľná. Štruktúra MIB je veľmi podobná Domain Name System (DNS). Obsahuje objekty v stromovej štruktúre, vo vrstvách od najobecnejších k špecifickým. Každý objekt má svoje presné umiestnenie. Mená objektov vznikajú prechádzaním tohoto stromu od koreňa, pričom jednotlivé názvy, ktorými sa na jednotlivých hladinách prechádza sa spájajú bodkami a vytvárajú jednoznačný identifikátor objektu. Táto štruktúra je pripravená na rozširovanie, umožňuje pridávať celé podstromy (MIB moduly) pre nové zariadenia. O tieto moduly sa starajú výrobcovia zariadení. O zachovanie a správu celej tejto MIB hierarchie sa starajú organizácie International Organization for Standardization (ISO) a International Telecommunication Union (ITU).

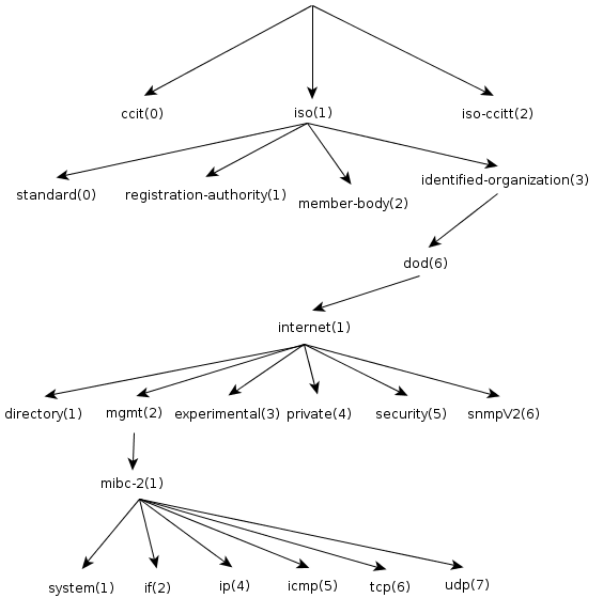
Jednotlivé mená objektov sú okrem mena označené aj číselným identifikátorom. Na identifikáciu sa používajú práve tieto číslicové identifikátory, ktoré rovnako ako mená tvoria stromovú štruktúru, oddeľujú sa bodkou a sú s menom pevne zviazané. Neposkytujú

síce taký jasný popis objektu, sú však oveľa kratšie ako textové popisy. V predchádzajúcej kapitole 1.5.1 je v ukážke definície SMIv2 objektu na poslednom riadku nasledujúci text:

```
::= { system 3 }
```

Pre tento objekt bude *sysUpTime* jeho pomenovanie a *3* jeho identifikátor. Znamená to, že to bude tretí objekt v podstrome objektu *system*, ktorý má identifikátor *1.3.6.1.2.1.1*, čo je vidieť aj na nasledujúcej ukážke MIB stromu. *System* je názov pre skupinu identifikátorov. Objekty sa v hierarchickej štruktúre snažia vytvárať spoločné skupiny a združovať sa v závislosti na type alebo určení objektov, ktoré reprezentujú. Skupiny majú aj svoje kódy, kód skupiny *system* je *sys*, ten je aj zahrnutý v názve identifikátora objektu. Celý identifikátor *SysUpTime* teda bude *1.3.6.1.2.1.1.3*. Takto rekurzívne je definovaný celý MIB.

Ukážka MIB stromu spolu s textovými popismi i číselnými identifikátormi:



Obrázok 1.2: Grafické znázornenie MIB stromu.

1.6 Formát SNMP správ

SNMPv3 správa sa dosť výrazne odlišuje od predchádzajúcich dvoch verzií protokolu. Formát SNMPv3 správy podrobnejšie rozoberie kapitola 1.6.3.

V rozvetvenej druhej verzii sa mierne líši formát správ v závislosti na úrovni bezpečnosti, aká je pre danú vetvu SNMP protokolu špecifická. SNMP Trap generátor pracuje s rozšírenou a štandardizovanou SNMPv2c verziou, preto podrobne popíšeme práve túto vetvu SNMP protokolu v kapitole 1.6.2. Verzie SNMPv1 a SNMPv2c si sú podobné v celkovej štruktúre správy. Každá SNMP správa sa skladá z hlavičky, tzv. *Message Header* a tela správy, *Message Body*, ktorú tvorí *Protocol Data Unit (PDU)*. Hlavička obsahuje informácie o tom, ako spracovávať telo správy, teda o akú verziu SNMP protokolu sa jedná a tiež obsahuje informácie týkajúce sa bezpečnosti. Telo správy obsahuje prenášané dáta. Popis PDU pre SNMPv1 obsahuje kapitola 1.6.1 a o detailoch SNMPv2c PDU hovorí kapitola 1.6.2.

Všeobecný formát správ SNMPv1 a SNMPv2c je zobrazený v nasledujúcej tabuľke:

Názov	*Typ*	*Popis*
Version	`Integer`	Označuje verziu SNMP protokolu správy, SNMPv1 je označená číslom 0.
Community	`Octet String`	Identifikuje reťazec, ktorý slúži na autentifikáciu správy.
PDU	-	Obsahuje telo správy.

Tabuľka 1.2: Všeobecný formát správ SNMPv1 a SNMPv2c.

1.6.1 Formát SNMPv1 PDU

SNMPv1 správa je krátka, čo je spôsobené jej jednoduchosťou a tiež nízkou úrovňou bezpečnosti. Všetky typy správ vo verzii SNMPv1, okrem Trap-PDU majú rovnaký formát. Jednotlivé typy správ sa rozlišujú podľa prvej položky: PDU Type. Podľa typu správy niektoré položky nadobúdajú alebo strácajú zmysel, napríklad Error Status a Error Index má dobrý zmysel iba pri odpovedi, nie pri posielaní požiadavku.

Tabuľka popisuje formát PDU SNMP protokolu verzie 1:

Názov	*Typ*	*Popis*
PDU Type	`Integer`	0 - GetRequest-PDU 1 - GetNextRequest-PDU 2 - GetResponse-PDU 3 - SetRequest-PDU
Request ID	`Integer`	Číslo, ktoré sa používa k párovaniu požiadaviek s odpoveďami. Vygeneruje ho zariadenie, ktoré posiela správu, odpovedajúce zariadenie na ňu odpovedá správou s rovnakým identifikačným číslom.
Error Status	`Integer`	Má zmysel iba pri GetResponsePDU, hovorí o výsledku požiadavku, tj. či nastala nejaká chyba a príslušné číslo chyby ju bližšie špecifikuje. 0 - noError: Nenastala žiadna chyba, používa sa v požiadavkoch. 1 - tooBig: Správa GetResponsePDU je príliš veľká. 2 - noSuchName: Požadovaný objekt nebol nájdený. 3 - badValue: Hodnota v požiadavku sa nezhoduje so štruktúrou objektu prijímateľa. 4 - readOnly: Pokus o zmenu objektu, ktorý slúži iba na čítanie. 5 - genErr: Nastal bližšie nešpecifikovaný druh chyby.
Error Index	`Integer`	Bližšie špecifikuje chybu, ak je Error Status nenulový.
Variable Bindings	-	Obsahuje súbor párov OID - hodnota.

Tabuľka 1.3: Formát SNMPv1 PDU.

SNMPv1 trap má špecifický formát, ktorým sa odlišuje od ostatných správ v tejto verzii SNMP protokolu. Takúto správu identifikuje položka PDU Type, ktorá má hodnotu 4.

Nasledujúca tabuľka popisuje formát SNMPv1 Trap-PDU:

Názov	*Typ*	*Popis*
PDU Type	`Integer`	4 - Trap-PDU
Enterprise	`Object Identifier`	Object identifier, ktorý bližšie určuje typ zariadenia, ktoré vygenerovalo trap.
Agent Address	`Network Address`	IP adresa SNMP agenta, ktorý trap vygeneroval.
Generic Trap	`Integer`	Špecifikuje niektorý z preddefinovaných generických typov trapov.
Specific Trap	`Integer`	Špecifikuje trap v závislosti na implementácii.
Time Stamp	`TimeTicks`	Čas, ktorý uplynul od poslednej reinicializácie SNMP agenta.
Variable Bindings	-	Obsahuje súbor párov OID - hodnota.

Tabuľka 1.4: Formát SNMPv1 Trap-PDU.

1.6.2 Formát SNMPv2c PDU

SNMPv2c priniesla dva nové typy správ oproti predchádzajúcej verzii, a to *GetBulkRequest* a *InformRequest*.

- *GetBulkRequest*: slúži na získavanie väčšieho množstva informácii, objektov súčasne v jednom požiadavku, jednej správe. Použitie takejto správy má znižovať zaťaženie siete. Je to taktiež možná alternatíva k iteratívnemu používaniu správy GetNextRequest.
- *InformRequest*: má slúžiť ako potvrdzovacia správa ku asynchrónnym udalostiam, akou je napríklad trap alebo tiež slúži na preposielanie informácií medzi viacerými NMS.

V tejto verzii protokolu sa narozdiel od SNMPv1 neodlišuje Trapv2-PDU, ale GetBulkRequest-PDU. SNMP Trap generátor túto správu nepoužíva, namiesto nej iteratívne používa GetNextRequest správu, preto podrobný formát GetBulkRequest-PDU neuvádzame.

Nasledujúca tabuľka popisuje formát SNMPv2c PDU:

Názov	*Typ*	*Popis*
PDU Type	`Integer`	0 - GetRequest-PDU 1 - GetNextRequest-PDU 2 - GetResponse-PDU 3 - SetRequest-PDU 4 - Trap-PDU: v SNMPv1, tu sa už nepoužíva 5 - GetBulkRequest-PDU: má vlasný formát 6 - InformRequest-PDU 7 - Trapv2-PDU 8 - Report-PDU
Request ID	`Integer`	Číslo, ktoré sa používa k párovaniu požiadavkov s odpoveďami. Vygeneruje ho zariadenie, ktoré posiela správu, odpovedajúce zariadenie na ňu odpovedá správou s rovnakým identifikačným číslom.
Error Status	`Integer`	Hovorí o úspešnosti výsledku požiadavku. Hodnoty 0 až 6 sú kompatibilné s SNMPv1. Kódy chybových stavov sú vo zvláštnej tabuľke 1.6.
Error Index	`Integer`	Ak je Error Status nenulový, bližšie špecifikuje objekt, ktorý vygeneroval chybu.
Variable Bindings	-	Obsahuje súbor párov OID - hodnota.

Tabuľka 1.5: Formát SNMPv2c PDU.

Tabuľka 1.6: Kódy chybových stavov položky Error Status v SNMPv2c.

Hodnota	*Kód*	*Popis*
0	*noError*	Nenastala žiadna chyba, používa sa v požiadavkoch.
1	*tooBig*	Veľkosť odpovede by bola príliš veľká.
2	*noSuchName*	Požadovaný objekt nebol nájdený.
3	*badValue*	Hodnota v požiadavku sa nezhoduje so štruktúrou objektu prijímateľa.
4	*readOnly*	Pokus o zmenu objektu, ktorý slúži iba na čítanie.
5	*genErr*	Nastal bližšie nešpecifikovaný druh chyby.
6	*noAccess*	Prístup k objektu bol odmietnutý z bezpečnostných dôvodov.
7	*wrongType*	Typ objektu špecifikovaný vo Variable Bindings je pre daný objekt nesprávny.
8	*wrongLength*	Dĺžka špecifikovaná pre objekt vo Variable Bindings je nesprávna.
9	*wrongEncoding*	Kódovanie špecifikované pre objekt vo Variable Bindings je nesprávne.
10	*wrongValue*	Hodnota špecifikovaná pre objekt vo Variable Bindings je nesprávna.
11	*noCreation*	Špecifikovaná premenná neexistuje a nemôže byť vytvorená.
12	*inconsistentValue*	Hodnota špecifikovaná vo Variable Bindings danému objektu vyhovuje, avšak mu nemôže byť priradená.

pokračovanie na ďalšej strane

Hodnota	*Kód*	*Popis*
13	*resourceUnavailable*	Pre nastavenie premennej je potrebný prostriedok, ktorý nie je k dispozícii.
14	*commitFailed*	Pokus o nastavenie premennej zlyhal.
15	*undoFailed*	Pokus o nastavenie premennej ako súčasti väčšej skupiny zlyhal, a pokus o odstránenie zmenených nastavení z ostatných premenných v skupine bol taktiež neúspešný.
16	*authorizationError*	Nastal problém pri autorizácii.
17	*notWritable*	Do premennej sa nedá zapisovať alebo nemôže byť vytvorená.
18	*inconsistentName*	Názov špecifikovaný vo Variable Bindings neexistuje.

1.6.3 Formát SNMPv3 správ

SNMPv3 priniesla hlavne rozšírenie bezpečnosti. Tomu sa prispôsobil aj formát SNMPv3 správy. Bola rozšírená o položky špecifikujúce a súvisiace s použitým bezpečnostným modelom. PDU v SNMPv3 zostalo takmer nezmenené, používa formát PDU z predchádzajúcej SNMPv2c verzie. PDU však bolo rozšírené o ďalšie položky, a to *Context Engine ID*, ktorý identifikuje akej aplikácii má byť PDU zaslaná na spracovanie a *Context Name*, čo je identifikátor objektu špecifikujúci kontext asociovaný s PDU.

Správa SNMPv3 obsahuje nasledujúce položky:

Názov	*Syntax*	*Popis*
Msg Version	`Integer`	Popisuje verziu SNMP protokolu. Pre SNMPv3 má hodnotu 3.
Msg ID	`Integer`	Číslo, ktoré sa používa k párovaniu požiadavkov s odpoveďami. Vygeneruje ho zariadenie, ktoré posiela správu, odpovedajúce zariadenie na ňu odpovedá správou s rovnakým identifikačným číslom.
Msg Max Size	`Integer`	Maximálna veľkosť správy, akú je schopný odosielateľ prijať, jej minimálna veľkosť je 484.
Msg Flags	`Octet String`	Slúži pre riadenie spracovania správy. Štruktúra je bližšie určená v RFC 3412.
Msg Security Model	`Integer`	Popisuje typ použitého bezpečnostného modelu.
Msg Security Parameters	-	Bližšie špecifikuje potrebné parametre bezpečnostného modelu, sú definované jednotlivými bezpečnostnými modelmi.
Scoped PDU	-	Obsahuje PDU spolu s ďalšími špecifickými parametrami. Bližšie špecifikuje RFC 3411.

Tabuľka 1.7: Formát SNMPv3 správ.

1.7 IPMI

Moderné servery podporujú štandard IPMI[3] (Intelligent Platform Management Interface). Je to definované rozhranie pre správu a monitorovanie počítačových systémov. Tento štandard je vedený firmou Intel[4], ktorý podporujú všetci renomovaný výrobcovia hardware. Pomocou tohoto štandardu je možné získavať trapy, teda priviesť zariadenie do stavu, ktorý spôsobí odoslanie notifikácie, teda trapu.

Hlavnou časťou zariadenia podporujúceho IPMI je BMC (Baseboard Management Controller), niekedy tiež nazývaný SP (Service Processor). Je to mikrokontrolér umiestnený na serverovej matičnej doske, ktorý sprístupňuje a monitoruje hardware. Operuje nezávisle na operačnom systéme inštalovanom na serveri. Všetky senzory teplôt, napätia, otáčok ventilátorov, svetelných diód sú napojené na tento mikrokontrolér. Dokáže vypínať i zapínať inštalovaný operačný systém, poskytovať vzdialený prístup či presmerovať grafický výstup, má vlastné sieťové rozhranie. Výhodou je možnosť „out-of-band“ monitoringu a tiež nezaťažovanie operačného systému informáciami o stave samotného servera.

Out-of-band monitoring je anglické pomenovanie prístupu k monitorovaniu servera, kedy je možné server spravovať zo vzdialeného počítača aj vtedy, keď je vypnutý jeho operačný systém, má na to zvyčajne zvláštne rozhranie a zvyčajne naň tiež dokáže presmerovať konzoly. Po prvotnom nastavení BMC sa prakticky celá správa serveru môže odohrávať bez nutnosti fyzického kontaktu so serverom, či už sa jedná o inštaláciu operačného systému, správu systému alebo reštart. Keďže má vlastné sieťové rozhranie, ktoré zvyčajne býva oddelené v inej sieti ako rozhranie operačného systému, prístupu k serveru nezabráni ani problém so sieťou. BMC je vždy zapnutý, dokonca i pri vypnutom operačnom systéme. Operačný systém môže získať prístup k informáciám o senzoroch len cez BMC. Časťou BMC je aj SEL (System Event Log), pamäť obsahujúca záznamy o dôležitých neštandardných udalostiach hlásených senzormi, ktoré vznikli na danom zariadení. Do nej môže pristupovať operačný systém a využívať takto uložené informácie pre svojho vlastného SNMP agenta. Takýto monitoring je schopný vykonávať operačný systém Windows i Linux, na tento účel existuje viacero démonov.

Na monitorovanie cez IPMI je možné využiť utilitu ipmitool[5]. Tá má okrem iného možnosť nastavovať parametre senzorov, siete či vypínať a zapínať napájanie. Vie zobraziť stavy všetkých senzorov prístupných BMC. Informácie o senzoroch sú uložené v SDR (Sensor Data Repository). Správnym použitím tejto utility je možné navodiť kritický stav, ktorý je totožný so stavom, akoby kritická situácia nastala na niektorom zo senzorov, čo bude viesť k adekvátnej odpovedi BMC, napríklad odoslanie trapu alebo podľa závažnosti udalosti môže viesť aj k vypnutiu operačného systému. Pri monitorovaní cez operačný systém, teda použití agenta bežiaceho na operačnom systéme, agent pristupuje do SEL alebo do systémového logu, ktorý je so SEL prepojený a takto získava informácie o stave zariadenia. V takomto prípade je možné použiť ipmitool, zapísať ním neštandardnú situáciu do SEL a testovať reakciu agenta.

Kapitola 2

SNMP Trap generátor

SNMP Trap generátor dokáže simulovať správanie sa reálneho hardware v neštandardných situáciách, a to odchytávaním a opätovným prehrávaním SNMP správ, ktoré zariadenie v kritickej situácii generuje. Zachytené správy uloží, dáva možnosť ich upravovať a odosielať. Vstavaným editorom je možné vytvoriť úplne nové, vlastné správy. Z viacerých rôznych súborov odchytených trapov dokáže SNMP Trap generátor vytvoriť trapstorm. Jedná sa o funkcionalitu, kedy sa odošle veľké množstvo trapov súčasne, teda simuluje kritickú situáciu, ktorá by nastala naraz na viacerých zariadeniach. Napríklad pri nefunkčnosti chladiacich zariadení v serverovej miestnosti, kedy sa začnú všetky stroje prehrievať súčasne a všetky súčasne začnú posielať trapy. Tým je možné testovať, ako sa správa NMS pod takouto záťažou. Vytvoriť takúto situáciu v skutočných podmienkach by si vyžadovalo prístup k množstvu zariadení súčasne, čo môže byť častokrát problém a tiež by bolo časovo náročné všetky tieto zariadenia správne nakonfigurovať.

SNMP Trap generátor podporuje aj ostatné časti SNMP protokolu, ako GetRequest, SetRequest či GetNextRequest. Využíva ich pri získavaní informácií o zariadení alebo pri nastavovaní zariadení. Umožňuje GetRequest požiadavkom zisťovať stav zariadenia, SetRequest správou meniť stav zariadenia a pomocou GetNextRequest správ vykonať SNMP walk, teda prejsť celý MIB zariadenia a získané informácie využiť ako vstupné informácie pre vstavaného SNMP agenta. Agent sa dokáže vydávať za simulované zariadenie, odpovedať na požiadavky NMS presne tak akoby na nich odpovedal skutočný hardware, ktorý simuluje.

SNMP Trap generátor umožňuje vyvíjať i testovať NMS bez nutnosti prístupu k reálnemu hardware, čo značne zefektívňuje a zlacňuje celý vývojový proces.

2.1 Prehľad podobných existujúcich aplikácií

Existuje veľký počet nástrojov, ktoré dokážu odosielať a prijímať správy SNMP protokolu, či už sú to jednoducho použiteľné command-line aplikácie ako napríklad balíček funkcií Net-SNMP[6], alebo zložitejšie simulátory agentov, generátory trapov využívajúce záznam systémových udalostí, démoni schopní odchytávať trapy, atď. Zvyčajne je potrebné na ich prepojenie a konfiguráciu k dosiahnutiu požadovaného výsledku, tj. pokrytie funkcionality SNMP Trap generátora, vynaložiť nemalé úsilie a čas. Je však málo komplexnejších programov, ktoré by sa zaoberali vernou simuláciou reálneho stroja. Napríklad SNMP agenti sú prispôsobený na reálne použitie, na získavanie aktuálnych dát priamo z hardware, nie na prezentáciu dát z nejakého konfiguračného súboru. Ani pri dôkladnom hľadaní sme sa nestretli so žiadnym voľne šíriteľným programom, ktorý by bez nutnosti

zložitejšej konfigurácie pokrýval celú funkcionalitu SNMP Trap generátora.

Na testovanie SNMP NMS existujú proprietárne programy. SNMP Trap generátor sa svojou funkcionalitou podobá napríklad MIMIC Simulator Suite[7], ktorej časťou je MIMIC SNMP Agent Simulator. Ten dokáže simulovať rôzne druhy hardware, až 20 000 SNMP agentov na jednom stroji, vie vytvoriť trapstorm, MIMIC Recorder dokáže importovať dáta zo skutočného hardware. Je to aplikácia, ktorá sa využíva i pri testovaní takého nástroja na správu akým je HP Openview[8]. Má väčšie možnosti ako SNMP Trap generátor, nevýhodou je ale cena a náročnejšie ovládanie. SNMP Trap generátor je jednoduchšou alternatívou k takémuto druhu software, je to riešenie šité presne na mieru, dostatočne pokrýva potrebnú funkcionalitu, je jednoduchý na inštaláciu, používanie a otvorený na ďalšie rozširovanie a prispôsobovanie podmienkam, do ktorých bude nasadený.

2.2 Implementácie SNMP

Existuje viacero implementácií SNMP ako napríklad Net-SNMP, Netsnmpj, jSNMP, SnmpB, PySNMP, Ruby SNMP, Net::SNMP, BSNMP. Líšia sa použitým jazykom i funkcionalitou. Asi najznámejšia a veľmi často používaná implementácia je už spomínaný balíček funkcií Net-SNMP: open-source implementácia, spravovaná komunitou. Je súčasťou mnohých linuxových či unixových distribúcií. Poskytuje jednoduché command-line nástroje pre všetky typy SNMP požiadavkov. Taktiež je schopný pracovať aj so SNMPv3 verziou, odchytávať trapy, vstavaný snmpd démon môže vystupovať ako agent, ktorý ale informácie získava zo systému na ktorom beží, nemá za úlohu simulovať iné zariadenie. Agent SNMP Trap generátora prezentuje iba informácie získané zo zariadenia, ktoré má simulovať bez závislosti na stroji, na akom beží.

SNMP Trap generátor využíva na prácu so SNMP protokolom PySNMP[9] framework. Je to implementácia SNMP protokolu v jazyku Python. Nie je to jediná možnosť, ako jazyk Python môže pracovať s protokolom SNMP, pri prvých pokusoch sa však ukázala byť dostatočne spoľahlivá na používanie a poskytovala všetku potrebnú funkcionalitu. Poskytuje funkcie pre vytváranie, kódovanie a dekódovanie správ, odosielanie a prijímanie správ SNMP protokolu. Pre potreby SNMP Trap generátora však bolo potrebné PySNMP pozmeniť, poopraviť drobnú chybu a prispôsobiť ho požiadavkám, aké si jeho funkcionalita kladie. Voľba PySNMP vychádzala zo situácie a prostredia, kam SNMP Trap generátor zapadá a pre aké účely bol vytvorený.

2.2.1 Chybička v PySNMP

Pri testovaní SNMP Trap generátora bola objavená chyba v definícii dátového typu Opaque. Je to dátový typ definovaný ASN.1. Jeho popis v RFC 1155[10] je nasledovný:

This application-wide type supports the capability to pass arbitrary ASN.1 syntax. A value is encoded using the ASN.1 basic rules into a string of octets. This, in turn, is encoded as an OCTET STRING, in effect "double-wrapping"the original ASN.1 value.

Použitie a funkcia dátového typu Opaque je naznačená v tabuľke 1.1. Podľa RFC sa prenášané dáta kódujú do reťazcov oktetov. Definícia triedy Opaque v PySNMP bola správne potomkom dátového typu OctetString, avšak nesprávne nastavovala jeden zo svojich implicitných parametrov. V SNMP Trap generátore je PySNMP framework uložený v adresári *source_ code/netfunctions/pysnmp*. Zdrojový súbor, v ktorom sa definícia

nachádza je v adresári PySNMP frameworku *pysnmp/proto* pod názvom *rfc1155.py*. Pôvodná definícia tejto triedy bola:

```
class Opaque(univ.OctetString):
    tagSet = univ.Integer.tagSet.tagImplicitly(
        tag.Tag(tag.tagClassApplication, tag.tagFormatSimple, 0x04)
        )
```

Správna definícia triedy Opaque by mala vyzerať nasledovne:

```
class Opaque(univ.OctetString):
    tagSet = univ.OctetString.tagSet.tagImplicitly(
        tag.Tag(tag.tagClassApplication, tag.tagFormatSimple, 0x04)
        )
```

Na tento problém sme upozornili vývojárov PySNMP. Veľmi promptne zareagovali, zaregistrovali a uznali túto chybu a tiež prijali našu navrhovanú opravu. Táto úprava bude zahrnutá do najbližšej verzie PySNMP.

2.3 Implemenácia SNMP Trap generátora

SNMP Trap generátor je implementovaný v jazyku Python, pre jeho verziu 2.6.4. Táto voľba vychádzala z cieľového prostredia, kde je SNMP Trap generátor nasadený. Spája v sebe Django[11], čo je framework na tvorbu webových aplikácií a PySNMP, čo je framework na prácu so SNMP. PySNMP bol zvolený, pretože je celý implementovaný v jazyku Python a teda nepotrebuje žiadne ďalšie závislosti. Na kódovanie a dekódovanie správ využíva PyASN1[12], ktorý je tiež súčasťou SNMP Trap generátora. Oba tieto nástroje majú BSD licenciu. Pod touto licenciou je taktiež framework Django, napísaný opäť v jazyku Python.

Pre Django a webové užívateľské rozhranie sme sa rozhodli aj kvôli možnému budúcemu jednoduchému rozšíreniu pre použitie viacerými používateľmi súčasne. Django má v sebe vstavaný webový server, tento je použitý pre vytvorenie užívateľského rozhrania. Dokáže sa tiež postarať o autentifikáciu a prístup užívateľov, generovať formuláre a overovať ich jednotlivé položky na základe špecifikovaných modelov, nahrávať súbory z užívateľovho počítača na stroj, kde beží SNMP Trap generátor, generovať HTML odpovede na základe šablón. Slúži ako grafické rozhranie nad SNMP Trap generátorom. Zároveň je možné jednoducho a ľahko meniť výzor užívateľského rozhrania prípadne ho rozšíriť o ďalšiu funkcionalitu. Taktiež výhodou je aj pohodlné a intuitívne ovládanie z prostredia obľúbeného webového prehliadača.

Pri používaní z webového rozhrania sa pri štarte vytvorí globálna štruktúra všetkých užívateľov, nazýva sa *all_logged_users* a jednotlivý užívatelia si do nej zaregistrujú svoje vlastné inštancie triedy *User_variables* pri prihlásení. Definícia užívateľskej triedy sa nachádza v *source_code/all_logged_users.py*. V prípade žiadosti užívateľa o vykonania akcie je zvyčajne vytvorené nové vlákno, v ktorom sa požadovaná akcia začne vykonávať. Ovládanie týchto novovzniknutých vláken prebieha cez premenné, ktoré sú uložené pre každého užívateľa zvlášť v jeho vlastnom objekte. Tie sa nastavujú podľa akcií užívateľa vo webovom rozhraní. Takto funguje prijímanie i odosielanie požiadavkov, trapstorm aj SNMP agent.

SNMP Trap generátor je navrhnutý tak, aby sa dal používať buď ako command-line aplikácia, napríklad pri automatickom testovaní, teda bez jeho webového rozhrania, alebo ako webová aplikácia, ktorá je prístupná nielen lokálne ale aj na vzdialenom počítači. Pri spustení SNMP Trap generátoru priamo z príkazového riadku zavolaním *./trapgen.py* z adresára so SNMP Trap generátorom spolu s príslušnými parametrami sú tieto parametre spracovávané štandardnou knižnicou jazyka Python nazývanou *optparse* a podľa nich sú volané jednotlivé funkcie zodpovedné za funkcionalitu. Rovnaká sada funkcií sa používa aj pri volaní z webového rozhrania a tieto funkcie zvyčajne očakávajú ako jeden z parametrov objekt užívateľa, kam ukladajú odchytené dáta a tiež podľa hodnôt v tomto objekte riadia svoju činnosť. Pri spustení bez webového rozhrania je vygenerovaný jeden preddefinovaný užívateľ, s ktorého štruktúrou funkcie pracujú.

Webové editory ako editor súborov a trapov, vytvorenie trapov ale taktiež funkcionalita agenta sú plne v správe SNMP Trap generátora, na ich zobrazovaní sa podieľa Django.

Pri implementácii funkcionality, hlavne práce s protokolom SNMP je použitý PySNMP framework, ale tiež aj PyASN1. PyASN1 slúži na kódovanie a dekódovanie správ obsahujúcich informácie, je využívané PySNMP na tvorbu a spracovávanie správ. PySNMP obsahuje definované dátové typy, ktoré využíva SNMP Trap generátor pri vytváraní správ. Obsahuje funkcie potrebné pre prácu so sieťou, odoslanie požiadavku a taktiež príjem správ, ktoré po prijatí poskytne SNMP Trap generátoru. Ten ich potom podľa potreby uchováva a prezentuje.

Odchytávanie aj posielanie trapov je vždy vykonávané v oddelenom vlákne. Táto funkcionalita je implementovaná v súbore *source_code/netfunctions/receive_send_trap.py*. Obsahuje triedy *Server_Thread*, ktorá vytvára nove vlákno, v ktorom sa spustí funkcia *run_server* z triedy *Trap_listenning_server*. Tá sa pripojí k portu a začne očakávať prichádzajúce správy. Pri príchode správy je zavolaná funcia *cbFun*, ktorá prichádzajúcu správu spracováva. Dôležité je správne ukončenie odchytávania. V prípade webového rozhrania na tento účel slúži odkaz *Stop* a v prípade command-line spustenia SNMP Trap generátor očakáva signál č. 15. Je to potrebné z toho dôvodu, že SNMP Trap generátor nemôže sám od seba vedieť, či ešte stále má čakať na prichádzajúce správy alebo už môže svoju činnosť ukončiť. Pre užívateľa používajúceho SNMP Trap generátor to môže byť nepohodlné, svoje výhody to však prináša pri automatizovanom testovaní, kedy je možné napríklad skriptom spustiť odchytávanie, spustiť generovanie trapov na zariadení a po ukončení tohoto generovania poslať signál na ukončenie SNMP Trap generátora.

Posielanie trapov je implementované v tom istom súbore ako prijímanie trapov, a vytvorením vlákna triedou *Sending_Thread* a spustením funkcie s názvom *run_send* triedy *Send_Trap*. Zdrojový XML súbor je spracovaný funkciou *source_code/web_interface/read_xml_functions/read_xml*, tá používa knižnicu *xml.dom.minidom*, ktorá dokáže spracovávať XML dokumenty, následne vytvorí nový trap a ten je odoslaný.

Trapstorm prebieha ako viaceré paralelné odosielania súčasne. Sú vytvorené viaceré vlákna, z ktorých každé odosiela jeden XML súbor. Trapstorm je implementovaný vo webovom rozhraní v *source_code/web_intefrace/views.py* a pre command-line rozhranie v súbore *./trapgen.py*.

SNMP walk odošle ako prvý SNMP GetNextRequest požiadavok na začiatok MIB stromu, na ktorú mu zariadenie odpovie nasledujúcou položkou MIB pomocou správy GetResponse. Tá nesie OID spolu s hodnotu nasledujúceho objeku MIB. Táto informácia je zobrazená, prípadne aj uložená a SNMP Trap generátor následne odošle ďalší

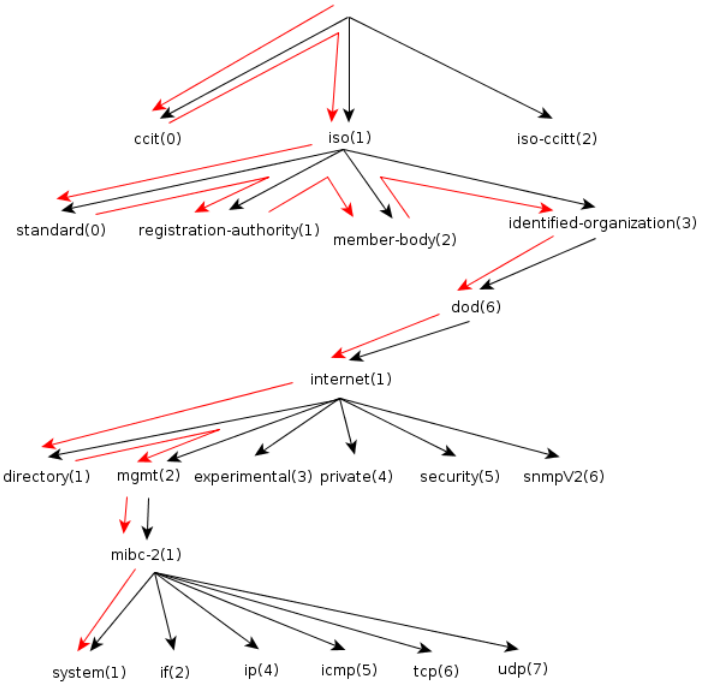
GetNextRequest s novozískaným OID. Takto prejde celý MIB. Získané informácie slúžia ako konfiguračný súbor pre vstavaného agenta. SNMP walk je implementovaný v *source_ code/netfunctions/snmp_ walk.py*. Tu je implementované aj odpočítavanie maximálneho času, ktorý SNMP Trap generátor čaká na príchod odpovede funkciou *cbTimerFun*.

SNMP get je implementovaný v súbore *source_ code/netfunctions/snmp_ get.py* pripojením sa na nejaký port, vytvorením a odoslaním správy so špecifikovaným OID, na ktorý sa užívateľ pýta, prijatím odpovede a jej spracovaním funkciou *cbRecvFun_ v1* alebo *cbRecvFun_ v2c*, v závislosti na verzii protokolu. Na odpoveď sa čaká iba určitý čas, o jeho odpočítavanie sa stará funkcia *cbTimerFun*.

SNMP agent sa spúšťa v novom vlákne iba v prípade spustenia webového rozhrania. Po spustení spracuje konfiguračný súbor, vytvorí si z neho vlastnú databázu MIB objektov a nastaví si práva na zápis a čítanie MIB objektov podľa zadaných „community string“ reťazcov. Následne začne prijímať požiadavky na špecifikovanom porte. Po prijatí je správa identifikovaná a v závislosti na jej type je vykonaná primeraná akcia. Agent je schopný reagovať na GetRequest, GetNextRequest a SetRequest správy.

Ako odpoveď na GetRequest správu zašle agent GetResponse správu s požadovaným OID spolu s jeho obsahom. Na GetNextRequest odpovie GetResponse správou s nasledujúcim OID v strome a jeho hodnotou. Tento typ správy využíva SNMP walk. MIB strom sa prechádza rekurzívne od koreňa k listom, na jednotlivých hladinách zľava doprava, teda od identifikátorov s nižším číslom na danej hladine k vyšším číslam identifikátorov.

Nasledujúci obrázok názorne popisuje prechod MIB stromom, prechod je zobrazený červenými šípkami:



Obrázok 2.1: Prechod MIB stromom pri používaní GetNextRequest správ.

SetRequest správa je prijatá a spracovaná agentom. Podľa OID a hodnoty, ktorú správa nesie si agent zmení hodnotu svojho vlastného OID na požadovanú hodnotu a ako odpoveď zašle GetResponse správu s touto novou zmenenou hodnotou. Správy so zlým autentifikačným reťazcom, nedostatočnými právami na vykonanie požadovanej akcie alebo zlej verzie protokolu sú zamietnuté. Agent SNMP Trap generátora musí pracovať s konfiguračným súborom s rovnakou verziou protokolu, v akej sú aj prichádzajúce požiadavky. Ak teda pracuje s konfiguračným súborom pre verziu SNMPv1, nemôže odpovedať na požiadavok SNMPv2c, pretože sa tieto verzie líšia v niektorých dátových typoch.

SNMP Trap generátor obsahuje okrem klasickej nápovedy, ktorá sa z príkazového riadku spúšťa parametrom *-h* aj detailnejšiu nápovedu vo webovom rozhraní. Tá popisuje všetky položky, ktoré môže užívateľ pri práci so SNMP Trap generátorm vypĺňať. Túto nápovedu tvoria obyčajné HTML stránky.

2.4 NMS

Existuje množstvo monitorovacieho software, medzi najznámejší open-source software patrí Nagios[13]. Ten sa nezaoberá iba monitorovaním hardware, ale aj služieb na ňom bežiacich. Funkcionalita sa rozširuje pridávaním rozširujúcich modulov. Existujú viaceré moduly na podporu SNMP protokolu a tiež návody, ako správne nastaviť SNMP monitoring. Spomedzi komerčných riešení vhodných na monitorovanie stojí za zmienku HP OpenView[8], IBM Tivoli[14], CA Unicenter[15] či Microsoft System Center Operations Manager (SCOM)[16]. Všetko sú to komplexné aplikácie s pokročilou funkcionalitou na správu serverov od renomovaných značiek. Výhodou komerčného software je lepšia podpora a znalosť hardware. Nevýhodou je jeho vyššia náročnosť a cena. Výber monitorovacieho systému závisí aj na tom, či monitorovanie bude prebiehať za pomoci agenta bežiaceho v operačnom systéme servera alebo využitím servisného procesora. Môže to ovplyvniť požiadavky kladené na NMS.

Servisný procesor nezaťažuje operačný systém, vyžaduje si však starostlivosť v podobe pravidelnej aktualizácie, zvyčajne je pripojený do oddelenej siete určenej na správu zariadení. Táto aktualizácia sa pri rôznorodosti hardware ťažko vykonáva automaticky a zároveň každé zariadenie si vyžaduje svoj vlastný aktualizačný súbor, nedá sa aplikovať iba jedna aktualizácia na všetky rozdielne stroje súčasne. Zároveň je potrebné sledovať vydávanie aktualizácií všetkými výrobcami hardware, ktorý je potrebné aktualizovať.

Riešenie s použitím agenta sa zasa aktualizuje pohodlnejšie, napríklad nejakým jednoduchým príkazom z NMS, pretože agent môže mať rozšírenú funkcionalitu a je schopný komunikácie s NMS nielen pomocou SNMP. NMS teda môže distribuovať aktualizácie jednotlivým agentom a automaticky ich aktualizovať, všetci agenti budú aktualizovaný na tú istú verziu a táto aktualizácia je pre všetkých rovnaká.

SNMP Trap generátor dokáže napodobiť iba štandardného SNMP agenta bez ďalšej pridanej funkcionality, dal by sa skôr prirovnať k agentovi, ktorý je implementovaný servisným procesorom. Jeho úlohou je testovať iba funkcionalitu spojenú so SNMP protokolom a reprodukovať skutočné správanie sa ozajstného agenta v rôznych situáciách. Ak by si niektorý z NMS vyžadoval aj iný typ komunikácie, SNMP Trap generátor nebude vhodným nástrojom na testovanie takéhoto NMS.

2.5 Získavanie trapov

Prvým krokom je nastavenie samotného zariadenia, ktoré má SNMP trapy generovať. Potrebné je nastaviť minimálne adresu a port, kam sa majú trapy posielať a tiež zvoliť verziu SNMP protokolu. Veľká väčšina zariadení má nejaké rozhranie, prevažne webové, ktoré umožňuje pohodlné nastavenie týchto parametrov. Vo vyspelejších zariadeniach existujú rôzne filtre a možnosti konfigurácie, ktoré umožňujú bližšie špecifikovať kedy a aký trap má byť vygenerovaný a odoslaný. Napríklad nastavenie hraničnej hodnoty teploty na teplotnom senzore, pri ktorej prekročení bude generovaný trap.

Zariadenia generujú trapy nielen pri krízových situáciách. Zvyčajne výrobcovia v dokumentáciách ku zariadeniam uvádzajú spôsob, akým je jednoducho možné overiť si túto funkcionalitu. Napríklad záložné zdroje je možné overiť krátkodobým odpojením z elektrickej siete, sieťové prvky môžu posielať trapy pri odpojení alebo zapojení sieťového rozhrania, server pri zastavení niektorého z ventilátorov, vytiahnutí jedného z redundantných zdrojov, teda vo všeobecnosti vytiahnutím nejakej Field Replaceable Unit (FRU), vytiahnutí servera zo serverovej skrine, zapnutí servera, otvorení krytov servera, zapnutím lokalizačnej diódy, atď.

Jednoduchý spôsob, ako prinútiť server odosielať všetky trapy je možné pomocou nástroja ipmitool a štandardu IPMI, popisovaného v kapitole 1.7. Ipmitool sa dá využiť napríklad zistenie informácií o všetkých senzoroch z SDR, nastavovanie medzí a hodnôt senzorov alebo zapisovanie do SEL, čím sa navodí kritická situácia bez rizika poškodenia zariadenia.

2.6 Reprezentácia SNMP trap správ

Odchytené SNMP trap správy sú ukladané vo formáte XML (Extensible Markup Language). Táto voľba sa ukázala ako veľmi šťastná, pretože umožňuje prehľadnú editáciu nielen vstavaným editorom SNMP Trap generátora, ale i jednoduchú úpravu ľubovoľným iným textovým editorom, nakoľko štruktúra XML tagov jasne popisuje jednotlivé prvky a nedáva možnosť pomýliť si položky SNMP trap správy ani pri rôznych verziách SNMP protokolu. Dovoľuje mať spoločne uložené SNMP trap správy vo verzii SNMPv1 i SNMPv2c. Taktiež vizuálne kopíruje štruktúru správ. Okrem toho umožňuje spracovanie súboru pomocou knižníc určených pre prácu s XML dokumentami, ktoré umožňujú z načítaných dát vytvárať objekty a tie sa neskôr dajú použiť pri vytváraní SNMP trap správ.

SNMP Trap generátor vytvára webové rozhranie, kde užívateľ po výbere súboru a v ňom konkrétneho trapu môže meniť jeho obsah vo forme webového formulára, pričom toto rozhranie zároveň kontroluje správnosť jednotlivých položiek a nedovolí užívateľovi uložiť zmeny, ktoré nie sú v súlade so špecifikáciou SNMP protokolu. Umožňuje tiež editáciu súboru ako celku.

Trapy oboch verzií SNMP v súbore sú uložené v spoločnom tagu <all_traps>, podobne aj viacero položiek <varbind> v jednej SNMP trap správe je obsiahnutých v spoločnom tagu <varbinds>. Varbind vznikol ako skratka zo slov „Variable Binding“, označuje dvojicu, ktorú tvorí OID a jeho hodnota spolu s typom. Hodnota OID môže byť iba takého dátového typu, akého typu môže byť objekt MIB, čo je kontrolované pri prijatí správy SNMP agentom, iba zo znalosti OID sa jeho typ určiť nedá. Prípustné typy objektov v MIB sú zobrazené v tabuľke 1.1.

Nasledujúca Document Type Definition (DTD) definícia dokumentu popisuje formát XML súboru obsahujúceho trapy:

```
<?xml version="1.0"?>
<!DOCTYPE all_traps [

<!ELEMENT all_traps (trap*)>
<!ELEMENT trap (version, community, pdu) >
<!ELEMENT pdu (pdu-type,(enterprise,
                         agent_address,
                         generic_trap,
                         specific_trap,
                         uptime) | (request-id,
                         error-status,
                         error-index), varbinds) >
<!ELEMENT varbinds (varbind*)>
<!ELEMENT varbind (varbind-oid, varbind-value)>
<!ELEMENT varbind-value (varbind-value-type, varbind-value-content)>

<!ELEMENT version (#CDATA)>
<!ELEMENT community (#CDATA)>
<!ELEMENT pdu-type (#CDATA)>

<!ELEMENT enterprise (#CDATA)>
<!ELEMENT agent_address (#CDATA)>
<!ELEMENT generic_trap (#CDATA)>
<!ELEMENT specific_trap (#CDATA)>
<!ELEMENT uptime (#CDATA)>

<!ELEMENT request-id (#CDATA)>
<!ELEMENT error-status (#CDATA)>
<!ELEMENT error-index (#CDATA)>
]>
```

Príloha A.1 obsahuje ukážku SNMPv1 trap správy, príloha A.2 zasa zobrazuje príklad SNMPv2c trap správy.

2.7 SNMP Agent

SNMP Trap generátor dokáže vystupovať vo funkcii agenta, teda komunikovať a odpovedať NMS. Komunikácia je inicializovaná na strane NMS. Agent SNMP Trap generátora vie odpovedať na základné požiadavky ako GetRequest, SetRequest, GetNextRequest. Kedže svoje údaje čerpá z vopred získaného alebo vytvoreného súboru, je v ňom možné pred spustením agenta nastaviť vlastnú odpoveď na jednotlivé požiadavky.

Formát súboru je jednoduchý, prehľadný, na riadku konfiguračného súboru sa vždy nachádza nasledovná trojica oddelená tabulátorom: OID, typ a hodnota ktorú OID reprezentuje. Prípustné typy OID sú zobrazené v tabuľke 1.1. Tento konfiguračný súbor je jednoducho editovateľný vstavaným editorom SNMP Trap generátora alebo ľubovoľným iným editorom textu. Ukážka konfiguračného súboru s niekoľkými vybranými objektami sa nachádza v prílohe A.3.

Funkcionalitu agenta je možné využiť na testovanie funkcií NMS ako napríklad autodetekcia servera. NMS sa pýta na presne špecifikované OID, kde predpokladá názov servera. Ak nájde požadovanú odpoveď, server sa spolu so svojím označením zobrazí v NMS. Dá sa využiť aj na kontrolu NMS, zisťovať ním či NMS pravidelne kontaktuje agentov svojimi požiadavkami a zisťuje ich stav.

2.7.1 Získavanie informácií pre vstavaného agenta

SNMP Trap generátor dokáže vykonať SNMP walk. Je to proces, kedy SNMP Trap generátor prejde celý strom MIB zariadenia na ktorom SNMP walk vykonáva. Tento proces je bližšie popísaný v implementácii SNMP trap generátora, v kapitole 2.3. Výstupom SNMP walk je konfiguračný súbor, ktorý očakáva agent SNMP Trap generátora ako svoj zdroj informácií pre verné napodobenie reálneho zariadenia.

Kapitola 3

Scenáre možného použitia

Nasledujúce scenáre použitia detailne popisujú spôsoby použitia SNMP Trap generátora pri praktickom využívaní. Sú to návody, akým spôsobom je možné SNMP Trap generátor používať pri testovaní NMS, ako využívať jeho funkcionalitu. Majú slúžiť ako príručka pre užívateľa.

3.1 Počiatočná inicializácia

SNMP Trap generátor nevyžaduje žiadnu dodatočnú inštaláciu, stačí mu nainštalovaný Python vo verzii 2.6.4. Počiatočná konfigurácia sa vykoná spustením SNMP Trap generátora s parametrom *--setup*. Takto je SNMP Trap generátor pripravený na použitie. Bližšie možnosti všetkých detailov nastavenia sú popísané v dokumentácii SNMP Trap generátora a zároveň aj formát konfiguračného súboru *--config.py* je samovysvetľujúci.

3.2 Spustenie webového rozhrania

Pri používaní webového rozhrania je najprv potrebné spustiť webový server príkazom *trapgen.py -i --int_port <číslo portu>*. Štandardne je použité „loopback“ rozhranie, dá sa však špecifikovať aj iné sieťové rozhranie parametrom *--int_address*, potrebné pri používaní zo vzdialeného počítača. Potom je možné sa prihlásiť cez webový prehliadač na adrese a porte, kde beží SNMP Trap generátor, teda napríklad pri lokálnom použití *http://localhost:<číslo portu>*. Štandardné meno a heslo na prihlásenie sa do SNMP Trap generátora je totožné: *fero*.

Pri spúšťaní bez webového rozhrania nie je nutné zadávať meno ani heslo.

3.3 Odchytávanie a prehrávanie trapov

Jedná sa o typické použitie SNMP Trap generátora, kedy sú najprv odchytené SNMP trap správy generované skutočným zariadením a potom prehrávané SNMP Trap generátorom podľa potreby.

3.3.1 Odchytávanie trapov

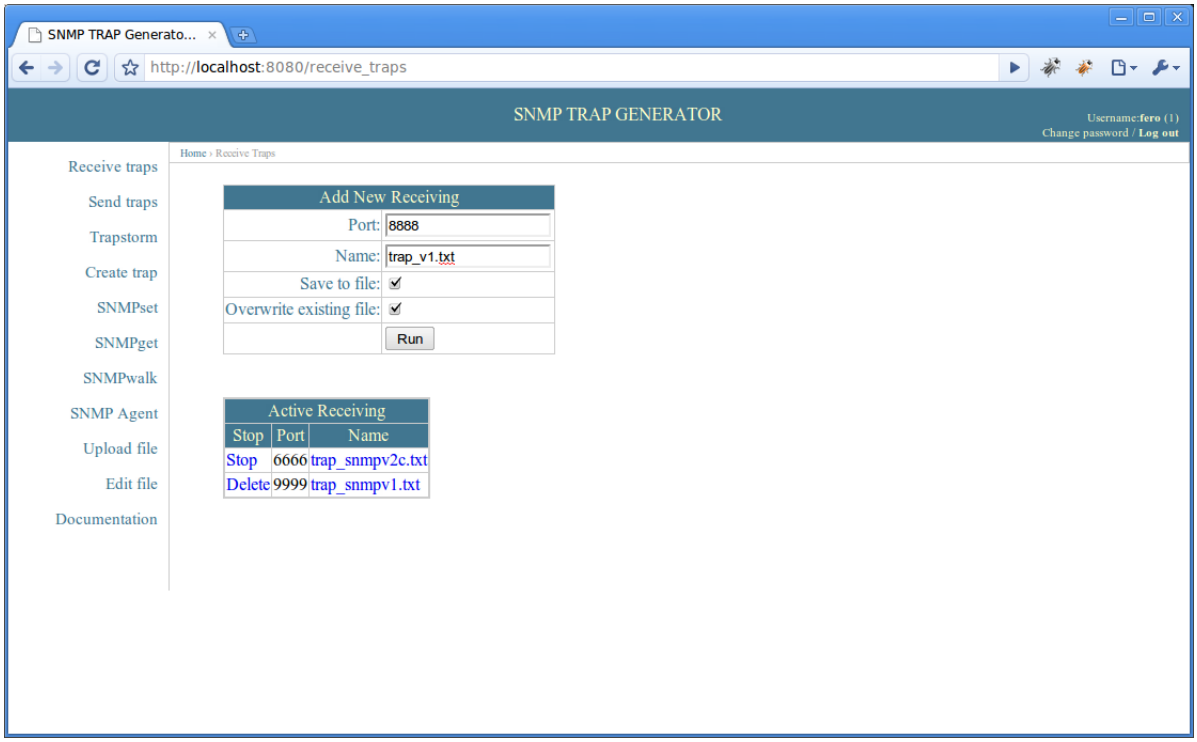
Súbory odchytených SNMP trap správ sú štandardne umiestnené v predvolenom adresári SNMP Trap generátora s názvom *Catched*, pokiaľ nebola pri vypĺňaní mena súboru zadaná úplná cesta.

Pri používaní webového rozhrania:

- Po spustení a prihlásení sa do webového rozhrania, v hlavnom menu zvolením *Receive traps* je zobrazený formulár s tabuľkou pod názvom *Add New Receiving*. V nej je potrebné vyplniť port, tj. položku s názvom *Port*, kde bude SNMP Trap generátor očakávať prichádzajúce trapy a tiež názov súboru, položku *Name*, ktorý bude odchytené trapy uchovávať. Zaškrtávacie políčko *Save to file* určuje, či SNMP Trap generátor bude odchytené SNMP trap správy ukladať do súboru alebo budú prístupné iba z webového rozhrania po dobu jeho aktivity. Políčko *Overwrie existing file* slúži na ochranu už existujúcich súborov. V prípade zvolenia názvu už existujúceho súboru nedovolí spustiť prijímanie SNMP trap správ a upozorní na to používateľa.

- Po spustení odchytávania je zobrazené práve spustené odchytávanie zobrazené v tabuľke *Active Receiving*, kde sú zobrazené všetky aktívne odchytávania spolu s ich detailami či prípadnými chybovými hláseniami. Tabuľka tiež ponúka možnosť prehliadať si čiastkové výsledky vyhľadávania, v prípade neukladania odchytených správ do súboru obsahuje odkaz na zobrazenie už odchytených trap správ.

- Ukončiť odchytávanie trap správ je možné vykonať z tabuľky *Active Receiving*, a odkazom s názvom *Stop*. Zmazať ukončené odchytávanie z tabuľky je umožnené odkazom *Delete*.



Obrázok 3.1: Odchytávanie SNMP trap správ.

Pri priamom spustení z príkazového riadku:

- Pri spustení z príkazového riadku je potrebné špecifikovať, že sa jedná o odchytávanie trapov. Na to slúži parameter *-r* alebo *--receive*. Ďalej je nutné zadať port, parameter *--rcv_port* a názov súboru kam sa budú odchytené trapy ukladať, parameter s názvom *--rcv_filename*.

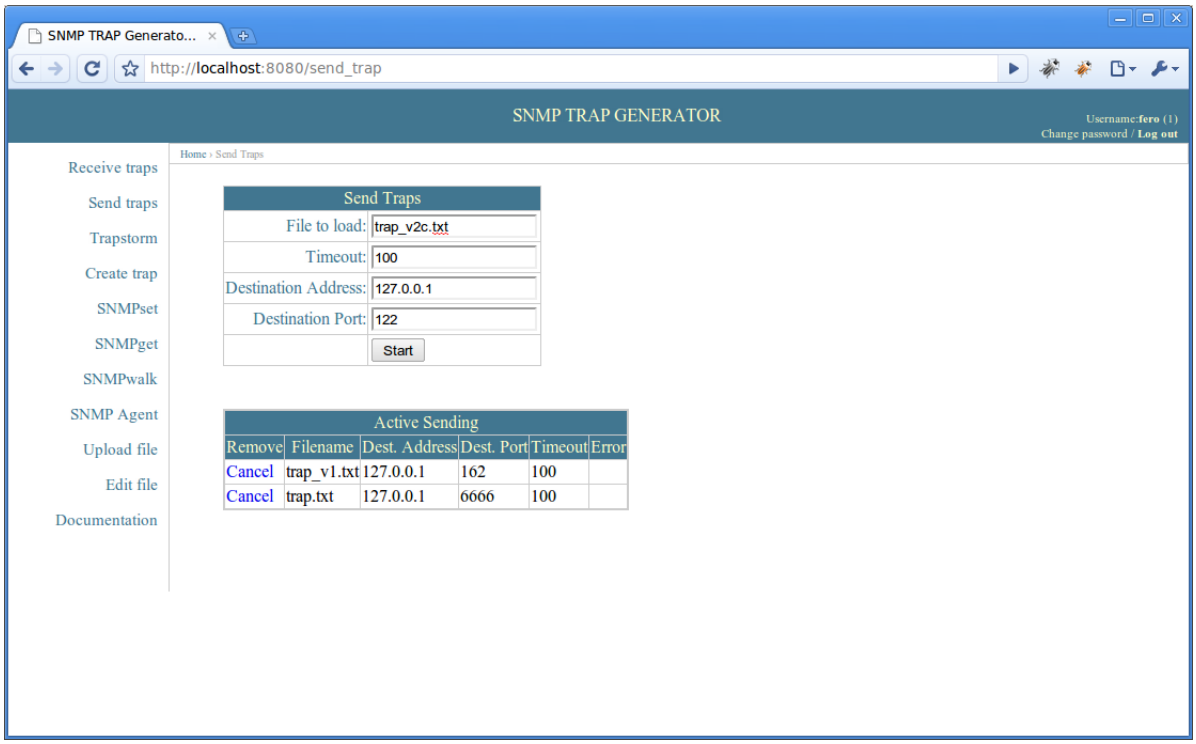
- Pre správne ukončenie odchytávania je potrebné zaslať signál č. 15 SNMT Trap generátoru.

3.3.2 Prehrávanie trapov

Súbory s odchytenými trapmi, ktorých názvy nie sú zadané plnou cestou, SNMP Trap generátor štandardne očakáva v predvolenom adresári s názvom *Catched*, pokiaľ nebolo v konfiguračnom súbore SNMP Trap generátora nastavené inak.
Pri používaní webového rozhrania:

- Po spustení a prihlásení sa do webového rozhrania, v hlavnom menu zvolením odkazu *Send traps* je zobrazený formulár s tabuľkou pod názvom *Send Traps*. Prvou položkou formulára je *File to load*, kde sa očakáva názov alebo úplná cesta k súboru, ktorý bude prehrávaný. *Timeout* určuje dĺžku prestávky v sekundách medzi odosielaním jednotlivých trap správ. *Destination Address* je IP adresa a *Destination Port* je číslo portu, kam budú trap správy odosielané.

- Aktívne prehrávania spolu s detailami o posielaní sú zobrazené v tabuľke *Active Sending*. Tu sa tiež nachádza odkaz *Cancel*, ktorým je možné predčasne ukončiť aktívne odosielanie SNMP trap správ.



Obrázok 3.2: Prehrávanie SNMP trap správ.

Pri priamom spustení z príkazového riadku:

- Pri spustení z príkazového riadku je potrebné špecifikovať, že sa jedná o odosielanie trapov. Na to slúži parameter *-s* alebo *--send* . Prehrávanie priamo z príkazového riadku si vyžaduje tie isté parametre ako odosielanie z webového rozhrania. Je potrebné špecifikovať názov súboru, v tomto prípade parametrom *--send_file*. IP

adresu a port kam sa budú odchytené trapy odosielať nastavujú parametre s názvami *--dst_address* a *--dst_port*. Dĺžku prestávky medzi odosielaním jednotlivých trapov určuje parameter *--send_timeout*, uvádza sa v sekundách.

- Predčasné ukončenie odosielania je možné vykonať klasickým ukončením procesu SNMP Trap generátora.

3.4 Trapstorm

Trapstorm je súčasné odosielanie väčšieho množstva trapov, ktoré simuluje vznik neštandardnej situácie naraz na viacerých zariadeniach. Jeho úlohou je testovať správanie sa i výkon NMS v takýchto náročných situáciách. Pre jeho používanie je potrebné mať vopred odchytené trapy z viacerých zariadení. Následne je možné v cykle odosielať trap správy, vždy po jednom zo všetkých zvolených súborov, oddelených nastaviteľnou časovou pauzou.
Pri používaní webového rozhrania:

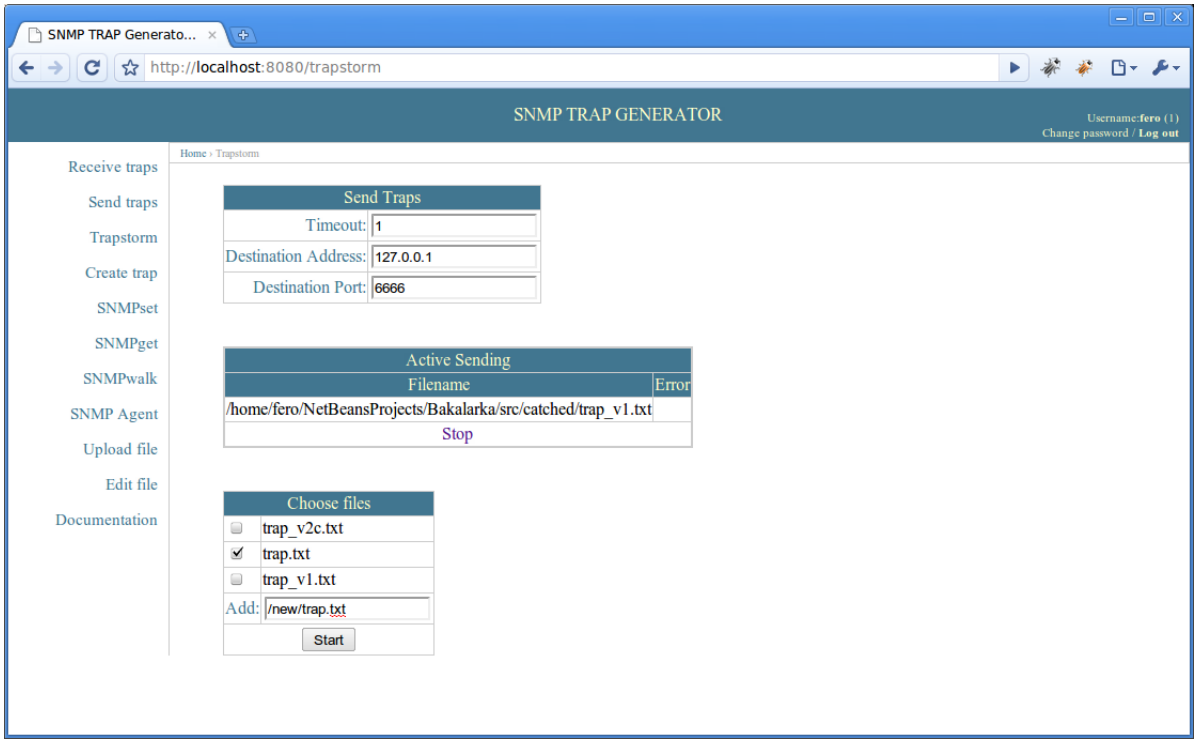
- Po spustení a prihlásení sa do webového rozhrania, v hlavnom menu zvolením *Trapstorm* je zobrazený formulár s tabuľkami *Send Traps*, *Active Sending* a *Choose files*. Políčko *Timeout* určuje dĺžku prestávky v sekundách medzi cyklami odosielania trapov. *Destination Address* je IP adresa a *Destination Port* je číslo portu, kam budú trapy odosielané. Tabuľka *Choose files* obsahuje zaškrtávacie políčka s názvami súborov uložených v predvolenom adresári SNMP Trap generátora pre uchovávanie odchytených trap správ. Zaškrtnutím sa vybraný súbor stane súčasťou odosielania. Políčko *Add* slúži na pridávanie súborov, ktoré sa nenachádzajú v predvolenom adresári. Očakáva sa plná cesta k súborom, viaceré súbory je potrebné oddeliť medzerou.

- Aktívne odosielanie je zobrazené v tabuľke *Active Sending*. Tá tiež obsahuje odkaz *Stop*, ktorým je možné predčasne ukončiť aktívne odosielania.

Pri priamom spustení z príkazového riadku:

- Pri spustení z príkazového riadku je potrebné špecifikovať, že sa jedná o Trapstorm. Na to slúži parameter *-t* alebo *--trapstorm*. Prehrávanie priamo z príkazového riadku si vyžaduje zadať IP adresu a port, kam sa budú odchytené trapy odosielať. To špecifikujú parametre *--storm_address* a *--storm_port*. Dĺžku prestávky medzi cyklami odosielania trapov určuje parameter *--storm_timeout*, uvádza sa v sekundách. Názvy súborov sa zadávajú ako argumenty, oddelené medzerou. SNMP Trap generátor očakáva buď úplné cesty k súborom, alebo vyhľadáva súbory podľa zadaného názvu vo svojom predvolenom adresári.

- Predčasné ukončenie odosielania je možné vykonať ukončením procesu SNMP Trap generátora.



Obrázok 3.3: Trapstorm.

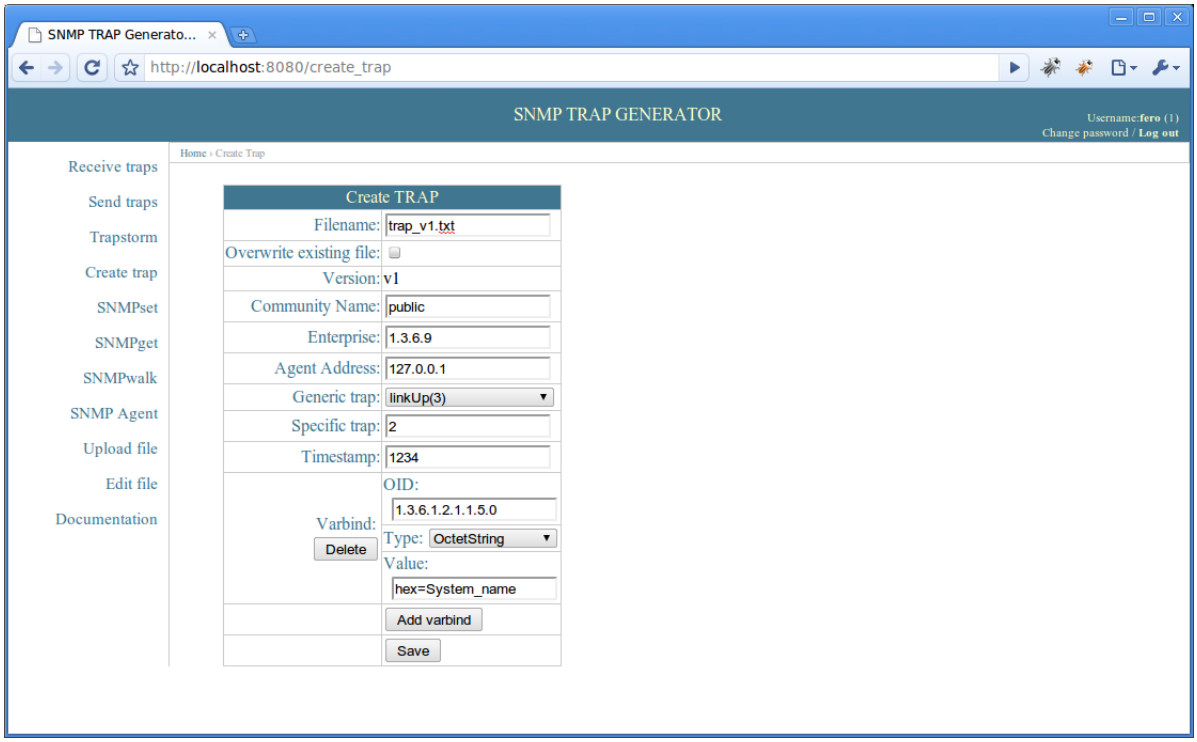
3.5 Vytvorenie vlastného trapu

SNMP Trap generátor umožňuje vytvoriť vlastné SNMPv1 aj SNMPv2c trapy podľa špecifikácie protokolu. Po spustení a prihlásení sa do webového rozhrania, v hlavnom menu zvolením *Create trap* je zobrazený formulár s názvom *Create TRAP*. Položka *Filename* očakáva názov, pod akým bude uložený súbor s novovytvorenou SNMP trap správou. Trapy sú štandardne ukladané v predvolenom adresári SNMP Trap generátora s názvom *Catched*, pokiaľ neboli zmenené jeho nastavenia alebo nebola zadaná úplná cesta v názve súboru. Zaškrtávacie políčko *Overwrite existing file* slúži ako ochrana pred prepísaním už existujúceho súboru s rovnakým názvom, pri jeho zaškrtnutí sa kontrola nevykonáva. Potrebné je ešte zvoliť verziu trap správy a tiež vyplniť *Community Name*, čo je reťazec starajúci sa o bezpečnosť v SNMP protokole.

Doposiaľ sa jednalo o spoločné položky oboch verzií protokolu, teraz na základe výberu verzie protokolu bude zobrazený formulár prislúchajúci danej verzii protokolu obsahovať nasledujúce položky:
Pri zvolení SNMPv1:

- Položka *Enterprise* očakáva Object identifier, ktorý bližšie určuje typ zariadenia, ktoré vygenerovalo trap správu.
- Položka *Agent Address* očakáva IP adresu SNMP agenta, ktorý trap správu vygeneroval.
- Položka *Generic Trap* má špecifikovať niektorý z preddefinovaných generických typov trap správ.
- Položka *Specific Trap* špecifikuje trap v závislosti na implementácii.

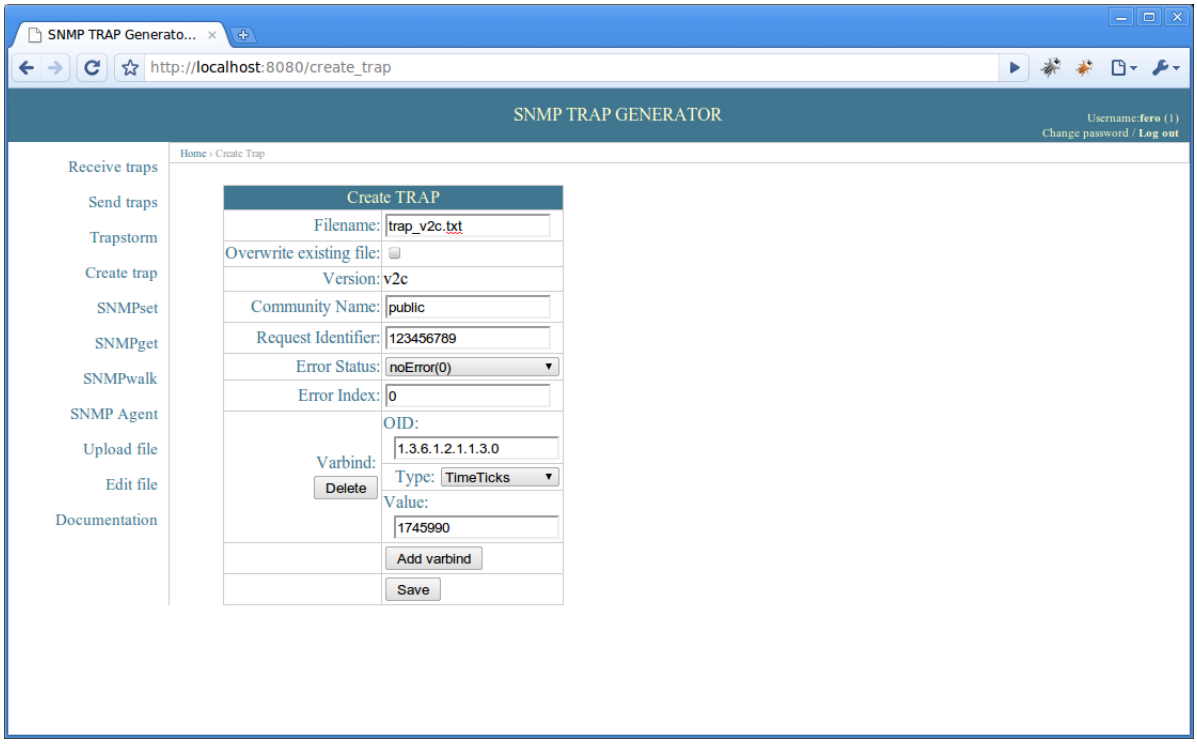
- Položka *Time Stamp* obsahuje čas, ktorý uplynul od poslednej reinicializácie SNMP agenta.
- *Varbind* obsahuje trojicu *OID* - *Type* - *Value*. Zvolená hodnota musí odpovedať zvolenému typu. Tu je potrebná znalosť MIBu zariadenia, teda i zamýšľaného OID a jeho typu.



Obrázok 3.4: Vytvorenie SNMPv1 trap správy.

Pri zvolení SNMPv2c:

- Položka *Request ID* je číslo, ktoré sa používa k párovaniu požiadavkov s odpoveďami. Za normálnych okolností ho generuje zariadenie, ktoré posiela správu, odpovedajúce zariadenie na ňu odpovedá správou s rovnakým identifikačným číslom.
- Položka *Error Status* hovorí o úspešnosti výsledku požiadavku.
- Položka *Error Index* bližšie špecifikuje objekt, ktorý vygeneroval chybu. Má zmysel pri nenulovej hodnote *Error Status*.
- *Varbind* obsahuje trojicu *OID* - *Type* - *Value*. Zvolená hodnota musí odpovedať zvolenému typu. Tu je potrebná znalosť MIBu zariadenia, teda i zamýšľaného OID a jeho typu.



Obrázok 3.5: Vytvorenie SNMPv2c trap správy.

Tvorba vlastnej SNMP trap správy ponúka viaceré možnosti ako navodiť bežne neočakávané situácie a sledovať reakcie NMS. Umožňuje:

- Vytvoriť alebo upraviť existujúci SNMP trap verzie 1 i verzie 2c, ktorý nebude obsahovať v sebe žiadne užitočné informácie, jeho časť obsahujúca varbindy bude prázdna.
- Vytvoriť alebo upraviť existujúci SNMP trap verzie 1 alebo verzie 2c, ktorý bude v sebe obsahovať varbind s vymysleným OID, nejakým typom a tiež hodnotou. Typ a hodnota musia byt kompatibilné.
- Vytvoriť alebo upraviť existujúci SNMP trap verzie 2c, postupne meniť *Error Status* podľa možností 0-5, prípadne aj v kombinácii so zmenou *Error Index*.

3.6 Používanie rôznych verzií SNMP za sebou idúcimi trap správami

Cieľom je sledovať reakciu a správanie sa NMS na prichádzajúce trapy rôznych verzií protokolu SNMP z jedného zariadenia. Na vytvorenie takejo situácie je potrebné odchytiť trap správy oboch verzií protokolu SNMP, teda SNMPv1 i SNMPv2c, alebo je možné ich vytvoriť, napríklad vstavaným webovým nástrojom na vytváranie trapov. Z trapov vytvoriť súbor, v ktorom budú striedavo alebo ľubovoľne rozmiestnené trapy rôznych verzií. Ten následne odoslať, či už za pomoci webového rozhrania alebo príkazového riadku.

3.7 Simulácia prehrievania zariadenia

Pri znalosti zariadenia je možné vytvoriť také poradie odosielania trapov, ktoré bude simulovať priebeh zahrievania a tiež následného ochladzovania nejakého senzoru a pozorovať reakciu NMS. Hlavne u teplotných senzorov sa jedná o veľmi realistickú simuláciu, pretože prehrievanie senzoru nastáva postupne, je nepravdepodobné aby zmena nastala dostatočne rýchlo a dokázala preskočiť nejaký hraničný stav. Zároveň je dosť nebezpečné vystavovať skutočné zariadenie kritickým situáciám k akým prehrievanie bezpochyby patrí. Generovanie týchto udalostí, bližšie popísané v kapitole 1.7 a ich simulácia SNMP Trap generátorom neprináša žiadne riziko pre použité zariadenie.

3.8 SNMP walk, SNMP set, SNMP get

SNMP walk slúži na zistenie konfiguračných informácií pre vstavaného agenta SNMP Trap generátora. Prechádza celým MIB zariadenia pomocou správ GetNextRequest a ukladá tieto informácie do súboru. SNMP set a SNMP get slúžia ako doplnenie funkcionality SNMP Trap generátora, je možné ich použiť na zistenie alebo nastavenie hodnoty konkrétneho OID, či už na vstavanom agentovi alebo na reálnom zariadení.

3.8.1 SNMPwalk

Pri používaní webového rozhrania:

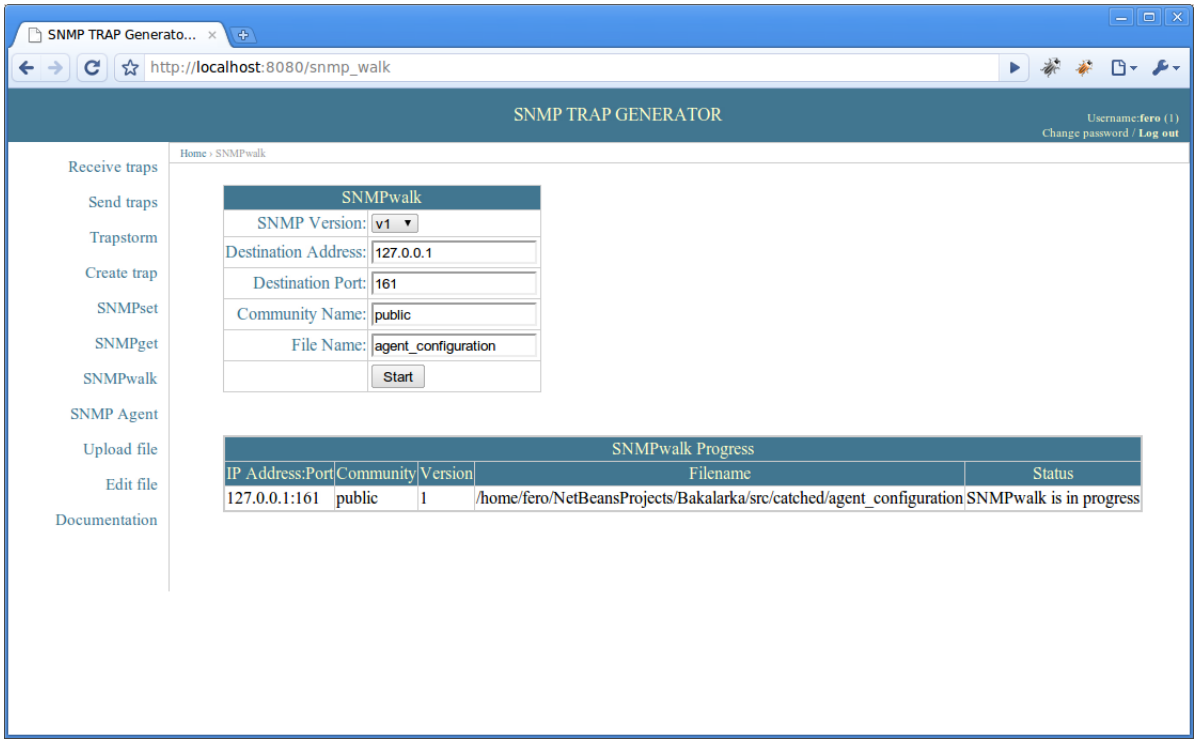
- Po spustení a prihlásení sa do webového rozhrania, v hlavnom menu zvolením *SNMPwalk* je zobrazený formulár s názvom *SNMPwalk*. Ponúka možnosť zvoliť verziu SNMP protokolu, a to buď SNMPv1 alebo SNMPv2c. Položky *Destination Address* a *Destination Port* určujú IP adresu a port zariadenia, kde má byť SNMP walk vykonávaný. *Community name* je autentifikačný reťazec slúžiaci na overovanie práv k prístupu na dané zariadenie. Musí sa zhodovať s reťazcom nastaveným na cieľovom zariadení a mať aspoň práva na čítanie. *File Name* očakáva buď úplnú cestu k súboru do ktorého budú uložené konfiguračné údaje, alebo je súbor so zadaným názvom uložený v predvolenom adresári.

- Po spustení SNMP walk je zobrazená tabuľka s názvom *SNMPwalk Progress*. Tá zobrazuje v akom stave sa SNMPwalk nachádza, či beží, skončil alebo nastala nejaká chyba. Po ukončení je možné zobraziť výsledky odkazom *Show*. Na odstránenie SNMPwalk z tejto tabuľky slúži odkaz *Clear*.

Pri priamom spustení z príkazového riadku:

- Pri spustení z príkazového riadku je potrebné zadať parameter *--snmpwalk*, ktorý špecifikuje že sa jedná o SNMP walk. SNMP verzia je určená parametrom nazývaným *--snmpwalk_version*. Autentifikačný reťazec slúžiaci na overovanie prístupových práv k danému zariadeniu sa nastavuje parametrom *--snmpwalk_community*. IP adresa a port zariadenia, kde sa má SNMP walk vykonávať sa špecifikuje parametrami *--snmpwalk_address* a *--snmpwalk_port*. Ako názov súboru, kde budú uložené výsledky SNMP walk je očakávaná úplná cesta k súboru, inak je súbor so zadaným názvom uložený v prednastavenom adresári. Parameter očakávajúci názov súboru sa volá *--snmpwalk_file*.

- Predčasné ukončenie SNMP walk je možné vykonať klasickým ukončením procesu SNMP Trap generátora.



Obrázok 3.6: Ukážka SNMP walk.

3.8.2 SNMPset

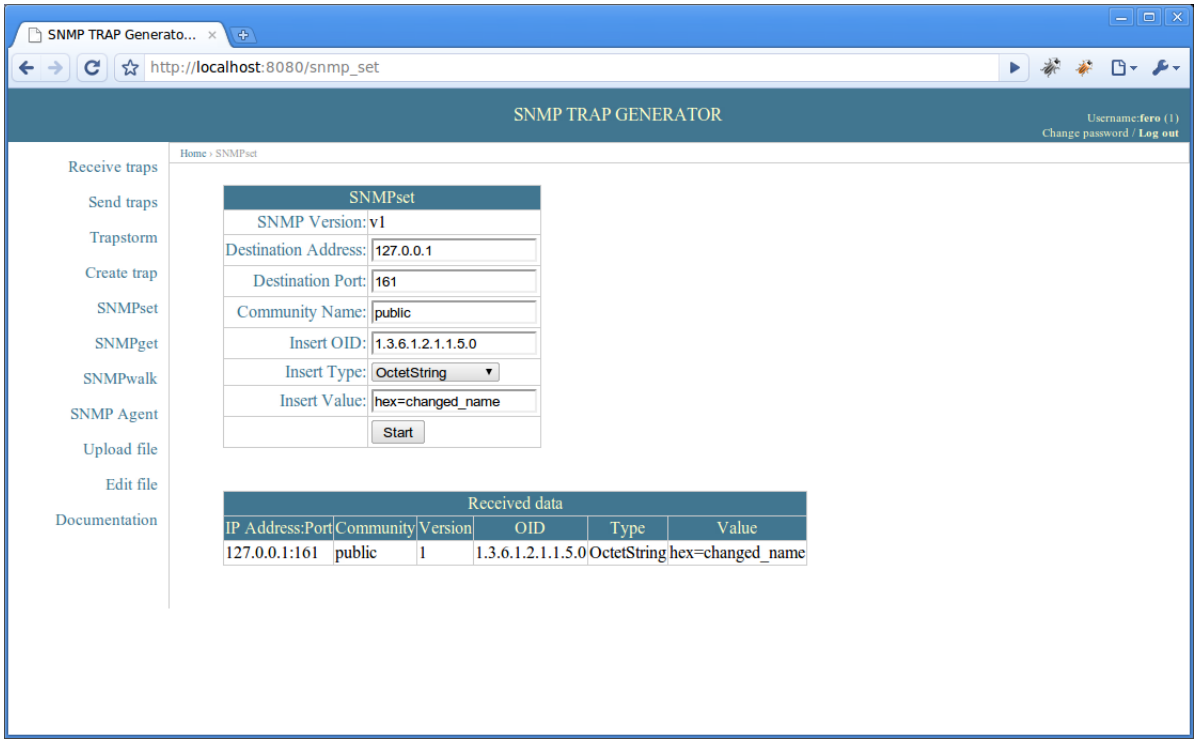
Pri používaní webového rozhrania:

- Po spustení a prihlásení sa do webového rozhrania, v hlavnom menu zvolením *SNMPset* je zobrazený formulár s názvom *SNMPset*. Ako prvé je potrebné zvoliť jednu z verzií SNMP protokolu, buď SNMPv1 alebo SNMPv2c. Následne zobrazené políčka sú totožné pre obe verzie protokolu. Je potrebné nastaviť *Destination Address* a *Destination Port*, tj. IP adresa a port kam bude SetRequest poslaný. *Community Name* určuje reťazec, ktorý bude použitý pri autentifikácii požiadavku. Tento reťazec musí mať na zariadení právo zapisovať, inak nebude SetRequest úspešný. Kombinácia položiek *OID*, *Type* a *Value* je na užívateľovi, očakáva sa znalosť MIB zariadenia, kam bude SetRequest zasielaný a zvolenie správnej kombinácie OID, typu OID a tiež prípustnej hodnoty na základe typu.

- Po odoslaní požiadavku sa zobrazí tabuľka *Received data*, ktorá obsahuje odpoveď potvrdzujúcu zmenu nastavení, prípadne je zobrazená informácia o chybe.

Pri priamom spustení z príkazového riadku:

- Pri spustení z príkazového riadku sa SNMPset špecifikuje parametrom *--snmpset*, SNMP verzia parametrom *--snmpset_version*. Reťazec, ktorý bude použitý pri autentifikácii požiadavku sa zadáva prostredníctvom parametra *--snmpset_community*. IP adresa a port sa určujú parametrami *--snmpset_address* a *--snmpset_port*. Je možné špecifikovať súbor parametrom *--snmpset_file* obsahujúci OID, typ a hodnotu ktoré sa majú nastaviť alebo tieto parametre nastaviť priamo ako argumenty v poradí OID, typ a hodnota.



Obrázok 3.7: Ukážka SNMP set.

3.8.3 SNMPget

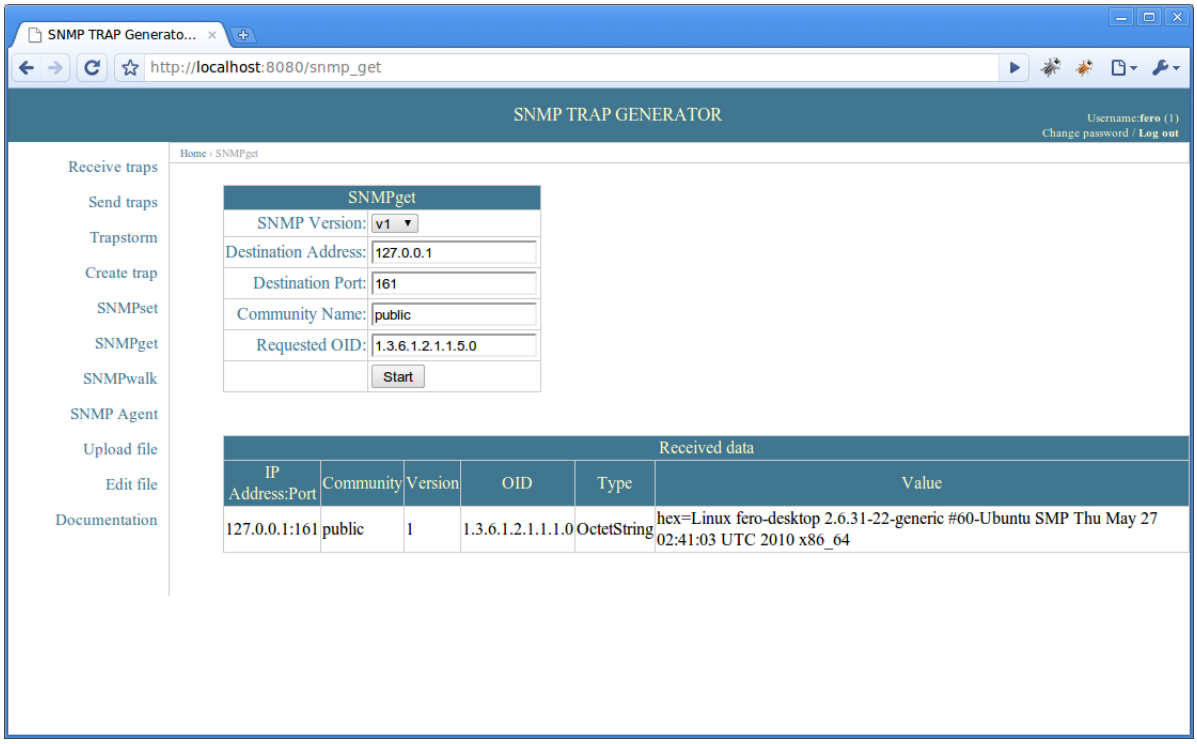
Pri používaní webového rozhrania:

- Po spustení a prihlásení sa do webového rozhrania, v hlavnom menu zvolením *SNMPget* je zobrazený formulár s názvom *SNMPget*. Prvou položkou voľba jednej z verzií SNMP protokolu, buď SNMPv1 alebo SNMPv2c. Nasledujú položky *Destination Address* a *Destination Port*, tj. IP adresa a port kam bude GetRequest zaslaný. *Community Name* určuje reťazec, ktorý bude použitý pri autentifikácii požiadavku. Tento reťazec musí mať na zariadení aspoň právo čítať. Položka *Requested OID* očakáva identifikátor objektu, ktorého hodnotu užívateľ požaduje.

- Po odoslaní požiadavku sa zobrazí tabuľka *Received data*, ktorá obsahuje získané informácie, prípadne je zobrazená informácia o chybe.

Pri priamom spustení z príkazového riadku:

- Spustenie SNMPget z príkazového riadku sa špecifikuje parametrom *--snmpset*, SNMP verzia parametrom *--snmpget_version*. Reťazec, ktorý bude použitý pri autentifikácii požiadavku sa zadáva prostredníctvom parametra *--snmpget_community*. IP adresa a port sa určujú parametrami *--snmpget_address* a *--snmpget_port*. Požadované OID sa zadávajú ako argumenty.



Obrázok 3.8: Ukážka SNMP get.

3.9 Agent odpovedá na GetRequest, SetRequest, SNMP walk

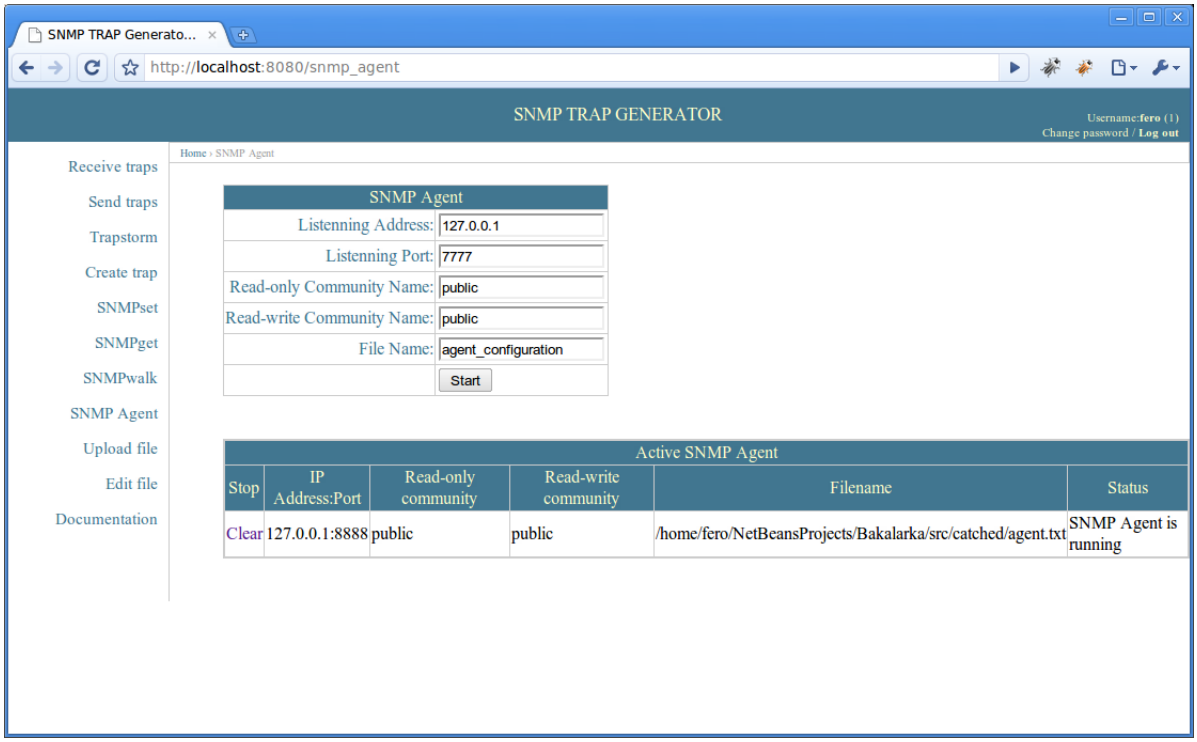
Odpovedanie na GetReqeust, SetRequest a GetNextRequest požiadavky dokáže testovať schopnosť NMS komunikovať s agentami spravovaných zariadení. Jedná sa napríklad o možnosť automatického detekovania prítomnosti nového zariadenia schopného komunikácie cez SNMP v sieti. To zvyčajne prebieha tak, že NMS osloví zariadenie na dobre známom porte a očakáva odpoveď. Z úspešnej odpovede dokáže zistiť napríklad názov zariadenia. Táto funkcionalita je zvyčajne označovaná pojmom „autodiscovery“. Taktiež je možné testovať, či NMS pravidelne kontaktuje agentov a požaduje informácie o ich stave. Vypnutím agenta je možné otestovať správanie sa NMS pri zistení nedostupnosti agenta, čo vlastne znamená vznik neštandardnej situácie.

Agent SNMP Trap generátora ku svojej činnosti najprv potrebuje získať informácie o zariadení, ktoré má simulovať. Na to slúži SNMPwalk. Ním sa vytvorí súbor, ktorý agent použije pri svojej inicializácii. Detailnejšie bol postup získania konfiguračného súboru popísaný v kapitole 3.8.1.
Pri používaní webového rozhrania:

- Po spustení a prihlásení sa do webového rozhrania, v hlavnom menu zvolením *SNMP Agent* je zobrazený formulár s názvom *SNMP Agent*. Prvou položkou je *Listenning Address*, tá určuje IP adresu a následne položka *Listenning Port*, tá určuje port, kde bude SNMP agent očakávať prichádzajúce požiadavky. *Read-only Community Name* a *Read-write Community Name* určuje reťazce, ktoré budú použité pri autentifikácii prichádzajúcich požiadavkov. *Read-only Community Name* určuje reťazce, ktoré nebudú môcť zapisovať a vykonávať zmeny požiadavkami SetRequest, budú ale môcť klásť GetRequest alebo GetNextRequest požiadavky. SNMP agent umožní vykonávať zmeny iba požiadavkom, ktoré majú *Community string* zhodný

s niektorým z reťazcov v *Read-write Community Name*. Položka *File Name* očakáva buď úplnú cestu k súboru obsahujúcemu konfiguračné údaje, inak je názov súboru vyhľadávaný v predvolenom adresári.

- Po spustení je zobrazená tabuľka *Active SNMP Agent*, tá detailnejšie popisuje aktivitu či chyby SNMP agenta. Ponúka odkaz *Clear* na zastavenie behu SNMP agenta.



Obrázok 3.9: Ukážka nastavenia SNMP agenta.

Pri priamom spustení z príkazového riadku:

- Pri spustení z príkazového riadku je potrebné špecifikovať, že sa jedná o spustenie SNMP agenta. Na tento účel slúži parameter *--snmpagent*. IP adresa a port, kde má SNMP agent očakávať požiadavky sú určené parametrami *--snmpagent_address* a *--snmpagent_port*. Reťazce sprístupňujúce informácie SNMP agenta na čítanie sa nastavia parametrom *--rocommunity*, jednotlivé reťazce sa oddeľujú čiarkou. Právo používať SetRequest na zapisovanie sa nastavujú parametrom *--rwcommunity*. Posledným parametrom je *--snmpagent_file*. Ako konfiguračný súbor agenta je očakávaná úplná cesta k súboru, inak je súbor so zadaným názvom vyhľadávaný v prednastavenom adresári.

- Predčasné ukončenie SNMP agenta je možné vykonať klasickým ukončením procesu.

Kapitola 4

Zhodnotenie projektu

SNMP Trap generátor sa stal aplikáciou, ktorá našla svoje praktické uplatnenie pri vývoji a testovaní častí NMS. Vychádzala z praktických potrieb a špecifických požiadavkov, ktoré dokázala naplniť. Stala sa súčasťou vývojového procesu a pomáha pri zdieľaní skutočného hardware medzi vývojármi a testermi. Využívaná je hlavne programátormi pre jednoduchšie a rýchlejšie testovanie vlastnej práce ako keby museli využívať ozajstný server. Zároveň uvoľňuje hardware pre rozsiahlejšie testovanie. SNMP Trap generátor zvyšuje ich pohodlie, keďže dokáže fungovať „lokálne“ v ich vlastnom vývojovom prostredí, na ich vlastnom stroji. Schopnosť pracovať automaticky je zasa využívaná ako jedna z foriem automatizovaného testovania. Jedná sa zvyčajne o prvé testy. Trapstorm funkcionalita zasa testuje situácie, ktoré bolo náročné testovať v reálnom prostredí, či už kvôli časovej náročnosti nastavovania všetkých strojov alebo kvôli ich dostupnosti a taktiež existujúcej sieťovej architektúre.

SNMP Trap generátor by bolo možné rozšíriť o SNMPv3, keďže aj táto verzia SNMP protokolu sa objavuje v spravovaných zariadeniach. Ďalším doplnkom by sa mohol stať MIB browser, ktorý by umožňoval nejaké grafické zobrazenie MIB objektov zariadenia. Vhodným doplnkom by mohol byť aj preklad OID na čitateľnejšie mená, napríklad OID 1.3.6.1.2.1.1.5 by sa preložilo podľa MIB daného zariadenia ako .iso(1) .org(3) .dod(6) .internet(1) .mgmt(2) .mib-2(1). system(1) .sysName(5). Taktiež by bolo možné SNMP Trap generátor rozšíriť na aplikáciu, ktorú by mohlo naraz využívať viacero užívateľov. SNMP Trap generátor je na toto rozšírenie pripravený, potrebné je akurát vylepšiť prácu so súbormi, aby sa nestalo že si užívatelia budú navzájom siahať na tie isté dáta.

Využitie frameworku Django prinieslo značné zjednodušenie vývoja webového rozhrania, poskytlo všetku potrebnú funkcionalitu, zároveň zbavilo nutnosti dodatočných inštalácií a rôznych závislostí. Pri používaní nenastali na žiadne problémy a dostatočná dokumentácia bola veľkým prínosom pri tvorbe SNMP Trap generátora.

PySNMP framework si vyžadoval úpravy, aby sa prispôsobil potrebám SNMP Trap generátora. SNMP Trap generátor potreboval využívať funkcionalitu PySNMP rozdielnym spôsobom, akým zamýšľali jeho tvorcovia. Pri zmenách bolo potrené zdĺhavo študovať zdrojové kódy, nakoľko dokumentácia nebola na príliš vysokej úrovni. Pri použití sa objavili aj drobné chyby, napríklad v zlej definícii jedného z dátových typov, ako o tom hovorí kapitola 2.2.1.

Literatúra

[1] *OSI model*,
http://en.wikipedia.org/wiki/OSI_model (3. augusta 2010)

[2] *International Organization for Standardization*,
http://www.iso.org/iso/home.html (3. augusta 2010)

[3] *Intelligent Platform Management Interface*,
http://www.intel.com/design/servers/ipmi/ (3. augusta 2010)

[4] *Intel*,
http://www.intel.com (3. augusta 2010)

[5] *IPMITooL*,
http://ipmitool.sourceforge.net/ (3. augusta 2010)

[6] *Net-SNMP*,
http://www.net-snmp.org/ (3. augusta 2010)

[7] Gambit Communications: *MIMIC Simulator*,
http://www.gambitcomm.com/site/ (3. augusta 2010)

[8] *Looking for HP OpenView?*,
https://h10078.www1.hp.com/cda/hpms/display/main/hpms_content.jsp?zn=bto&cp=1-10^36657_4000_100__&jumpid=reg_R1002_USEN (3. augusta 2010)

[9] *PySNMP*,
http://pysnmp.sourceforge.net/ (3. augusta 2010)

[10] *Structure and Identification of Management Information for TCP/IP-based Internets*,
http://tools.ietf.org/html/rfc1155#section-3.2.3.6 (3. augusta 2010)

[11] *Django*,
http://www.djangoproject.com/ (3. augusta 2010)

[12] *ASN.1 tools for Python*,
http://pyasn1.sourceforge.net/ (3. augusta 2010)

[13] *Nagios*,
http://www.nagios.org/ (3. augusta 2010)

[14] *Tivoli software*,
http://www-01.ibm.com/software/tivoli/ (3. augusta 2010)

[15] *CA NSM*,
`http://www.ca.com/us/system-management.aspx`(3. augusta 2010)

[16] *Microsoft System Center Operations Manager*,
`http://www.microsoft.com/systemcenter/en/us/operations-manager.aspx`
(3. augusta 2010)

[17] *Security in SNMPv3 versus SNMPv1 or v2c*,
`http://www.aethis.com/solutions/snmp_research/snmpv3_vs_wp.pdf` (3. augusta 2010)

[18] *The TCP/IP Guide: A TCP/IP Reference You Can Understand!*,
`http://www.tcpipguide.com/` (3. augusta 2010)

[19] *Simple Network Management Protocol*,
`http://en.wikipedia.org/wiki/Simple_Network_Management_Protocol` (3. augusta 2010)

[20] *SNMP: Simple Network Management Protocol*,
`http://www.javvin.com/protocolSNMP.html` (3. augusta 2010)

[21] *SNMP*,
`http://www.rabbit.com/documentation/docs/modules/SNMP/ModSnmp.html`
(3. augusta 2010)

[22] *Essential SNMP*,
`http://oreilly.com/catalog/esnmp/chapter/ch02.html` (3. augusta 2010)

Dodatok A

Ukážky SNMP trap správ

A.1 Ukáža SNMPv1 trap správy

```
<?xml version="1.0" ?>
<all_traps>
  <trap>
    <version>
      v1
    </version>
    <community>
      public
    </community>
    <pdu>
      <pdu-type>
        TrapPDU
      </pdu-type>
      <enterprise>
        1.3.6.1.4.1.42.2.175.103.2
      </enterprise>
      <agent_address>
        10.18.141.137
      </agent_address>
      <generic_trap>
        6
      </generic_trap>
      <specific_trap>
        2
      </specific_trap>
      <uptime>
        176938
      </uptime>
      <varbinds>
        <varbind>
          <varbind-oid>
            1.3.6.1.4.1.42.2.175.103.2.1.2.0
          </varbind-oid>
          <varbind-value>
```

```
            <varbind-value-type>
              OctetString
            </varbind-value-type>
            <varbind-value-content>
              hex=/SYS/MB/V_+1V8
            </varbind-value-content>
          </varbind-value>
        </varbind>
        <varbind>
          <varbind-oid>
            1.3.6.1.4.1.42.2.175.103.2.1.9.0
          </varbind-oid>
          <varbind-value>
            <varbind-value-type>
              OctetString
            </varbind-value-type>
            <varbind-value-content>
              hex=Upper Non-recoverable going low
            </varbind-value-content>
          </varbind-value>
        </varbind>
      </varbinds>
    </pdu>
  </trap>
</all_traps>
```

A.2 Ukáža SNMPv2c trap správy

```
<?xml version="1.0" ?>
<all_traps>
  <trap>
    <version>
      v2c
    </version>
    <community>
      public
    </community>
    <pdu>
      <pdu-type>
        TrapPDU
      </pdu-type>
      <request-id>
        1355447613
      </request-id>
      <error-status>
        0
      </error-status>
      <error-index>
        0
```

```
      </error-index>
      <varbinds>
        <varbind>
          <varbind-oid>
            1.3.6.1.2.1.1.3.0
          </varbind-oid>
          <varbind-value>
            <varbind-value-type>
              TimeTicks
            </varbind-value-type>
            <varbind-value-content>
              1744093
            </varbind-value-content>
          </varbind-value>
        </varbind>
        <varbind>
          <varbind-oid>
            1.3.6.1.4.1.42.2.175.103.2.1.2.0
          </varbind-oid>
          <varbind-value>
            <varbind-value-type>
              OctetString
            </varbind-value-type>
            <varbind-value-content>
              hex=/SYS/MB/V_BAT
            </varbind-value-content>
          </varbind-value>
        </varbind>
       </varbinds>
    </pdu>
  </trap>
</all_traps>
```

A.3 Ukážka z konfiguračného súboru pre SNMP agenta

```
(1.3.6.1.2.1.1.1.0) OctetString hex=Linux fero-desktop 2.6.31-22-generic
     #60-Ubuntu SMP Thu May 27 02:41:03 UTC 2010 x86_64
(1.3.6.1.2.1.1.2.0) ObjectIdentifier 1.3.6.1.4.1.8072.3.2.10
(1.3.6.1.2.1.1.3.0) TimeTicks 1619273
(1.3.6.1.2.1.1.4.0) OctetString hex=Root <root@localhost> (configure
     /etc/snmp/snmpd.local.conf)
(1.3.6.1.2.1.1.5.0) OctetString hex=cawko
(1.3.6.1.2.1.1.6.0) OctetString hex=Unknown (configure
     /etc/snmp/snmpd.local.conf)
(1.3.6.1.2.1.1.8.0) TimeTicks 0
...
(1.3.6.1.2.1.1.9.1.3.1) OctetString hex=The SNMP Management Architecture
     MIB.
(1.3.6.1.2.1.1.9.1.3.2) OctetString hex=The MIB for Message Processing
```

```
     and Dispatching.
(1.3.6.1.2.1.1.9.1.3.3) OctetString hex=The management information
     definitions for the SNMP User-based Security Model.
(1.3.6.1.2.1.1.9.1.3.4) OctetString hex=The MIB module for SNMPv2 entities
(1.3.6.1.2.1.1.9.1.3.5) OctetString hex=The MIB module for managing TCP
     implementations
(1.3.6.1.2.1.1.9.1.3.6) OctetString hex=The MIB module for managing IP
     and ICMP implementations
(1.3.6.1.2.1.1.9.1.3.7) OctetString hex=The MIB module for managing UDP
     implementations
(1.3.6.1.2.1.1.9.1.3.8) OctetString hex=View-based Access Control Model
     for SNMP.
...
(1.3.6.1.2.1.4.22.1.3.2.78.128.192.1) internet 78.128.192.1
(1.3.6.1.2.1.4.22.1.3.2.78.128.192.3) internet 78.128.192.3
(1.3.6.1.2.1.4.22.1.3.2.78.128.194.245) internet 78.128.194.245
(1.3.6.1.2.1.4.22.1.3.2.78.128.195.191) internet 78.128.195.191
...
(1.3.6.1.2.1.25.1.4.0) OctetString hex=BOOT_IMAGE=/boot/vmlinuz-2.6.31-22-
     generic root=UUID=afd60e92-9ce9-48af-bba9-899102e3be9d ro
     ipv6.disable=1 quiet splash
...
(1.3.6.1.2.1.25.4.2.1.4.1) OctetString hex=/sbin/init
(1.3.6.1.2.1.25.4.2.1.4.2) OctetString hex=kthreadd
(1.3.6.1.2.1.25.4.2.1.4.3) OctetString hex=migration/0
(1.3.6.1.2.1.25.4.2.1.4.4) OctetString hex=ksoftirqd/0
(1.3.6.1.2.1.25.4.2.1.4.5) OctetString hex=watchdog/0
(1.3.6.1.2.1.25.4.2.1.4.6) OctetString hex=migration/1
(1.3.6.1.2.1.25.4.2.1.4.7) OctetString hex=ksoftirqd/1
(1.3.6.1.2.1.25.4.2.1.4.8) OctetString hex=watchdog/1
...
(1.3.6.1.2.1.25.4.2.1.5.9714) OctetString hex=/dev/sdb1 /media/SQ004141P03
     -o rw,nosuid,nodev,uhelper=devkit,uid=1000,gid=1000,dmask=0077
(1.3.6.1.2.1.25.4.2.1.5.9717) OctetString hex=/dev/sdb5
     /media/7E38F7C038F7760D -o rw,nosuid,nodev,uhelper=devkit,uid=1000,
     gid=1000,dmask=0077

...
(1.3.6.1.2.1.25.3.2.1.3.1025) OctetString hex=network interface lo
(1.3.6.1.2.1.25.3.2.1.3.1026) OctetString hex=network interface eth0
...
```

Dodatok B

CD-ROM

Na priloženom CD sa nachádza táto bakalárska práca v digitálnej podobe spolu s adresárom obsahujúcim SNMP Trap generátor.

Štruktúra priloženého CD:

- *bakalarska_praca.pdf*: Bakalárska práca.
- *bakalarska_praca.tar.gz*: Zdrojové kódy bakalárskej práce.
- *snmp_trap_generator*: Adresár obsahujúci SNMP Trap generátor.
  - *catched*: Adresár obsahujúci odchytené správy a ukážky správ.
  - *config.py*: Konfiguračný súbor SNMP Trap generátora.
  - *database*: Databáza užívateľov SNMP Trap generátora.
  - *prog_documentation.pdf*: Programátorská dokumentácia.
  - *readme*: Súbor popisujúci základné použitie SNMP Trap generátora.
  - *source_code*: Adresár obsahujúci zdrojové kódy SNMP Trap generátora.
  - *trapgen.py*: Základ SNMP Trap generátora, slúži na jeho spustenie.
  - *user_documentation.pdf*: Užívateľská dokumentácia.

Pre použitie SNMP Trap generátora je potrebné z priloženého CD skopírovať obsah adresára *snmp_trap_generator* na lokálny disk. SNMP Trap generátor pre svoje správne fungovanie potrebuje pri počiatočnej inicializácii upraviť konfiguračné súbory, preto ho nie je možné spúšťať priamo z priloženého CD.

Následná inicializácia a spustenie SNMP Trap generátora z jeho lokálneho umiestnenia je popísaná v kapitole 3.1.