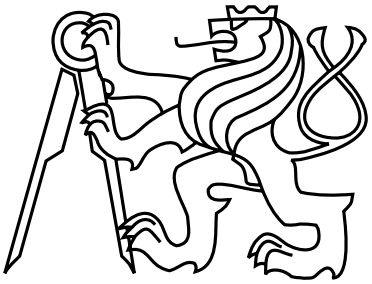
České vysoké učení technické v Praze
Fakulta elektrotechnická
Katedra počítačů



Diplomová práce

SNMP/XML brána

Bc. Tomáš Hroch

Vedoucí práce: Ing. Peter Macejko

Studijní program: Elektrotechnika a informatika, strukturovaný, Navazující magisterský

Obor: Výpočetní technika

13. května 2009

Poděkování

Na tomto místě bych rád poděkoval vedoucímu práce panu Ing. Peteru Macejkovi za jeho rady a připomínky, které vedly ke zdárnému dokončení práce. Zároveň můj dík patří i rodině za podporu po celou dobu mých studií.

Prohlášení

Prohlašuji, že jsem práci vypracoval samostatně a použil jsem pouze podklady uvedené v přiloženém seznamu.
Nemám závažný důvod proti užití tohoto školního díla ve smyslu §60 Zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon).

V Praze dne 13. 5. 2009 .....

Abstract

The main aim of this project is to develop system, which could connect two communication protocols. The first protocol is widely used SNMP and the second one is optimized XML-based protocol. The document describes implementation of the system which includes several optimalizations to reduce memory usage. The program is written in C++ programming language and is designed for OS Linux.

Abstrakt

Cílem tohoto projektu je implementovat systém, jenž by dovoloval spojení dvou komunikačních protokolů. Prvním z nich je široce rozšířený protokol SNMP a druhým je navržený optimalizovaný XML protokol. V dokumentu se kromě samotné implementace programu klade důraz na optimalizace, které snižují nároky na operační paměť. Program je napsán v jazyce C++, cílový operační systém je Linux.

Obsah

Seznam obrázků

Seznam tabulek

Kapitola 1

Úvod

Správa velkých počítačových sítí je v dnešní době naprosto samozřejmým úkolem většiny administrátorů. Velké množství spravovaných sítí se neomezuje pouze na lokální prostředí dané firmy či instituce. Může být naopak rozprostřena v rámci jednoho města, státu či dokonce několika států najednou. Efektivní spravování takové komunikační infrastruktury je úkolem velice náročným.

Jedním z protokolů, který takovouto vzdálenou správu umožňuje, je SNMP. Na jeho základě bylo vybudováno bezpočet aplikací, které mají za úkol sledovat provoz na síti, zatížení určitého systému a v neposlední řadě umožnit administrátorovi vzdálenou správu daného přepínače, routeru či pracovní stanice.

Protokol SNMP byl navržen v dřívějších dobách a nemusí plně vyhovovat dnešním požadavkům, ať už na bezpečnost nebo efektivní využití přenosových médií. Pan Ing. Peter Macejko se ve své diplomové práci ([1]) zaobíral použitím technologií XML a návrhu protokolu, který by umožňoval minimálně stejnou funkcionalitu jako protokol SNMP a tento zefektivnil.

Tato práce se zaobírá vytvořením protokolové brány, která by umožnila použít navržený XML protokol ke správě strojů, které stále používají protokol SNMP. Cílem je vytvořit softwarový produkt, který bude plnit úkol prostředníka mezi správcem a spravovaným strojem. Hlavními problémy jsou implementace navrženého XML protokolu a jeho spojení s několika verzemi protokolu SNMP. Důležitým aspektem vývoje je i orientace na snížení nároků na operační paměť. Proto bude v každé části programu kladen důraz na efektivní správu datových struktur.

V kapitole 2 bude podrobně popsán protokol SNMP, jeho komunikační struktury a typy zpráv.

Kapitola 3 se zabývá rozborem navrženého XML protokolu.

Analýza systému a diskuse o možných směrech implementace bude popsána v kap. 4.

V kapitole 5 bude probrána detailní funkčnost implementovaného programu.

Ověření funkce a další testování systému bude popsáno v kapitole 6.

Kapitola 2

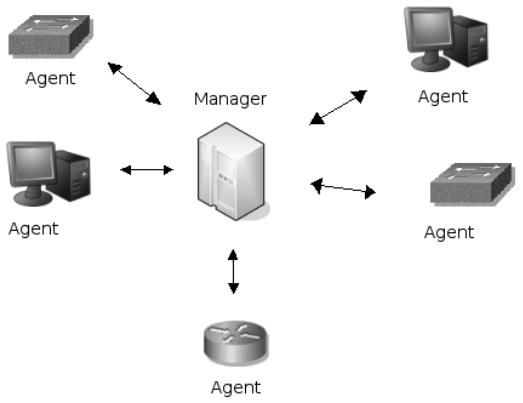
SNMP

SNMP, nebo-li Simple Network Management Protocol, je v dnešní době jeden z nejrozšířenějších protokolů na správu počítačové sítě. Je to aplikační protokol, který je součástí TCP/IP rodiny protokolů. Byl vyvinut skupinout IETF (Internet Engineering Task Force) a přijat jako standard v roce 1989. Umožňuje sledovat síťový provoz, hledat a řešit problémy, které se při provozu vyskytnou.

2.1 Správní struktura

SNMP je tvořen sadou standardů, které popisují správu sítě, zahrnující samotný komunikační protokol, definici databázové struktury (SMI) a datové objekty (MIB).

Základním funkčním principem je model Klient - Server ([2]). Struktura spravované sítě se tak dělí na tři klíčové elementy - spravované zařízení, agenta a manažera (viz obrázek 2.1).



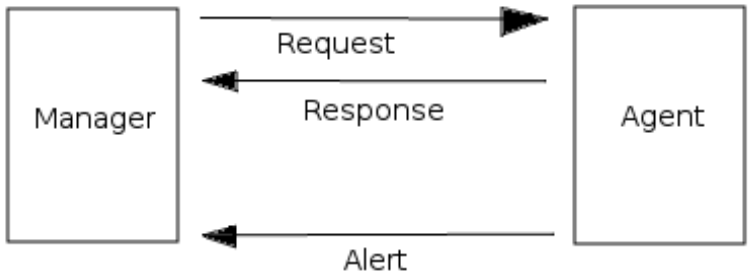
Obrázek 2.1: Základní princip fungování SNMP spravované sítě

- **Spravovaný systém** - je zařízení (přepínač, router, atd.), na kterém je spuštěn SNMP agent. Toto zařízení shromažďuje sledované informace a pak je dává k dispozici manažerovi pomocí SNMP protokolu.

- **Agent** - je software určený pro pro správný překlad požadavků manažera a jejich vykonání na sledovaném systému. Navíc může při sledování posílat manažerovi upozornění, že něco není se systémem v pořádku.

- **Manažer** (NMS - Network Managemet System) - je aplikace, která sleduje a spravuje všechny systémy na sledované síti. Tento systém získává od agentů data, zpracovává je do vizuální podoby, čímž dává možnost administrátorovi mít přehled o celé síti. Zároveň umožňuje měnit sledované parametry přímo u agenta.

Komunikace můžeme rozdělit do dvou kategorií dle toho, kdo ji započal. Základní schéma je vyjádřeno na obrázku 2.2 ([3]).



Obrázek 2.2: Komunikace mezi SNMP manažerem a agentem

V první části schématu je vyobrazeno standardní chování managera, který posílá dotazy agentovi, který mu odpovídá. Přesný výpis příkazů a zpráv, které si mohou tyto dva systémy mezi sebou vyměňovat, bude diskutován dále v této kapitole.

Druhá část schématu popisuje moment, kdy na sledovaném systému nastala nějaká extrémní situace (např. zatížení síťového spoje se blíží k maximu) a agent informuje manažera pomocí zprávy Alert (v SNMP jsou to zprávy TRAP či INFORM, obě budou diskutovány dále).

Je nutné zmínit, že SNMP protokol pracuje nad transportním protokolem UDP, který je nepotvrzovaný. Není tedy zaručeno, že bude komunikace probíhat bezchybně. Je možné, že některé dotazy a příkazy vůbec nedojdou ke svému cíli, o čemž se druhá strana nikdy nedozví. Tento fakt může být překážkou při správě rozsáhlých sítí, kde jsou špatné síťové spoje.

2.2 SMI, MIB standardy

Jak již bylo zmíněno dříve, SNMP je sada standardů, která kromě komunikačního protokolu musí definovat i strukturu sledované databáze a samotná data. Tyto informace byly definovány ve standardech SMI a MIB.

SMI

SMI je zkratkou pro Structure and Identification of Management Information for TCP/IP-based Internets. Tento standard ([4]) popisuje a definuje základní datové struktury a typy, které protokol využívá. Jednotlivé objekty jsou pojmenovány a organizovány, aby bylo možno k těmto datům logicky přistupovat. Dle standardu musí mít každý objekt jméno, syntaxi a kódování. Jméno jednoznačně identifikuje objekt. Datový typ (číslo, řetězec) je určen syntaxí. Kódování zajišťuje správnou serializaci dat při přenosu mezi systémy.

Objekty, identifikovány svým jménem (OID), jsou seřazeny do hierarchické struktury. K identifikaci je použito Abstract Syntax Notation One (ASN.1). Každý OID identifikátor je složen ze skupiny přirozených čísel, které vyjadřují jeho pozici v pomyslném stromu. Strom má kořen, který je spojen hranami s očíslovanými uzly. Každý uzel může mít vlastní děti, čímž tvoří vlastní podstrom. Takto je možno pokračovat dále do značné hloubky stromu. Tento standard též specifikuje, jaké objekty tvoří počátek správní databáze.

MIB

MIB je zkratka pro Management Information Base. Je to soubor definicí, které popisují parametry a vlastnosti sledovaného zařízení. Existuje více než 100 různých MIB, které popisují různá zařízení. Každý takovýto soubor definic musí splňovat předpisy SMI, aby byla zaručena správná interpretace objektů. Každý objekt (někdy také nazýván MIB objekt) je unikátně identifikován svým OID a všechny dohromady jsou uspořádány do stromové struktury tak, jak to bylo popsáno v minulém odstavci.

Objekty v dané databázi se dělí na *skalární* a *tabelární*. Skalární objekty reprezentují jeden parametr sledovaného zařízení (např. počet ethernetových karet v přepínači), kdežto tabelární objekty jsou spojením několika spřízněných objektů (např. routovací tabulka je spojením jednotlivých záznamů, coby řádků dané tabulky).

V rámci hierarchického uspořádání jsou vyhrazeny vyšší úrovně stromu (blíže kořenu) jednotlivým standardizujícím organizacím, nižší úrovně jsou poté zadány jednotlivými společnostmi. Každý výrobce si může definovat svojí privátní větev, do které umístí specifické informace daného zařízení.

MIB, které nebyly standardizovány a oficiálně schváleny, jsou umístěny do větve experimentální.

2.3 Verze SNMP protokolu

Celkem byly doposud standardizovány tři verze protokolu SNMP. Každá z nich definuje svoje specifické datové typy a používané datové rámce pro komunikaci.

SNMPv1

V první verzi protokolu byly definovány dvě skupiny datových typů:

- Základní datové typy (Simple data types)
- Aplikační typy

Základní typy jsou definovány v SNMPv1 SMI a definují základní používané hodnoty:

- **INTEGER** - celá čísla od -2^{31} do $2^{31} - 1$
- **OCTET STRING**
- **OBJECT IDENTIFIER** - identifikace jednotlivých objektů v rámci normy ASN.1

Aplikační specifické typy pak jsou:

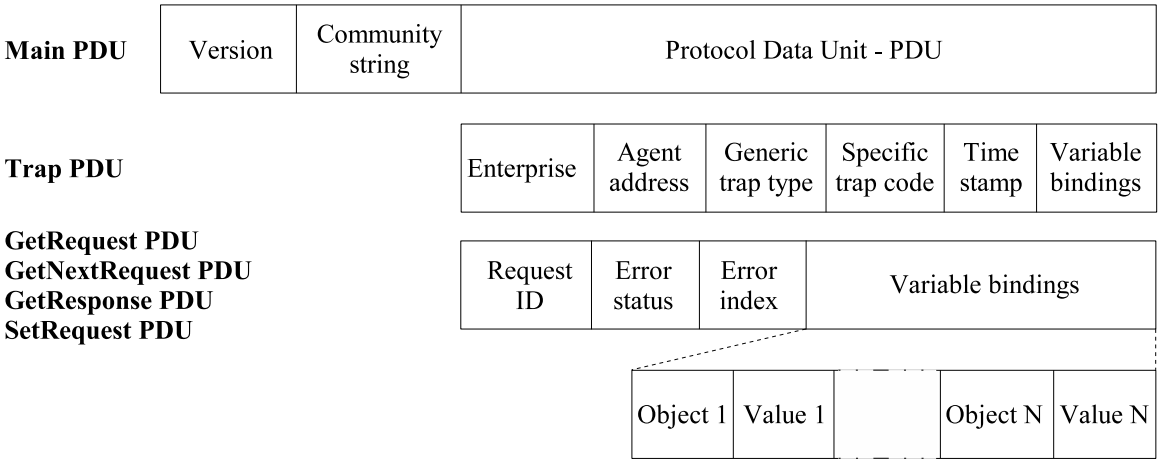
- **Network Address** - obecná síťová adresa pro podporu mnoha rodin protokolů
- **IpAddress** - přímo definovaný typ pro IP adresu. SMIv1 podporuje pouze 32 bitovou adresu (IPv4)
- **Counter** - čítač, vyjádřen celým číslem bez znaménka; jeho hodnota se pouze zvyšuje a to až do maxima a pak se vrací zpět na nulu
- **Gauge** - je definována jako nezáporné celé číslo; může hodnotu zvyšovat i snižovat a to v definovaných mezích minima a maxima
- **Time Ticks** - počet hodinových tiků od nějaké události, měřeno v setinách vteřiny
- **Opaque** - typ dovolující přenášet libovolná data v kódování ASN.1. Tato data jsou zakódována jako OCTET STRING a následně přenesena médiem.
- **Integers** - celočíselný typ, který předefinovává specifikaci v SMI
- **Unsigned Integer** - celočíselný typ bez znaménka, který stejně jako předchozí předefinovává specifikaci.

Komunikační mechanismus mezi manažerem a agentem je definován pomocí datových rámců, které je možné v rámci SNMPv1 přenášet. Tyto jsou:

- **Get Request** - získání hodnoty uzlu identifikovaného OID (zpráva od manažera agentovi)

- **Get Next Request** - žádost o hodnotu uzlu následujícího po zaslaném OID (od manažera k agentovi)
- **Set Request** - nastavení hodnoty uzlu specifikovaném OID (od manažera k agentovi)
- **Get Response** - odpověď agenta manažerovi na Get a Set zprávy. Obsahují požadovanou hodnotu
- **Trap** - zpráva od agenta manažerovi, která upozorňuje na nastalé situace na monitorovaném systému.

Strukturu jednotlivých SNMP paketů zobrazuje obrázek 2.3.



Obrázek 2.3: Schéma datových paketů protokolu SNMPv1 a v2 ([1])

Hlavní část datového paketu je tvořena poli Version a Community string. První popisuje verzi SNMP protokolu použitou při komunikaci a druhé je heslo pro přístup k položkám MIB. Blíže k bezpečnosti v dalším odstavci. Druhá část paketu se liší dle typu zprávy. Paket odeslaný agentem při výskytu monitorované události obsahuje informace, které popisují druh problému.

- *Enterprise* - identifikuje typ zařízení, které zprávu poslalo
- *Agent adress* - adresa zařízení, kde běží agent
- *Generic trap type* - specifikuje, zda-li se jednalo o některý z předdefinovaných typů událostí (linkDown, linkUp, coldStart, aj.)
- *Specific trap type* - identifikuje jednu z mnoha specifických událostí

Druhým typem zprávy jsou dotazy a odpovědi, které zasílá manažer a agent na ně odpovídá. Jednotlivá pole mají následující význam:

- *Request ID* - pořadové číslo dotazu (aby manažer věděl, na co přišla odpověď)
- *Error status* - je nastaven pouze u odpovědi a obsahuje druh problému, který se při dotazu objevil
- *Error index* - asociuje problém s instancí objektu.

Společným polem pro oba dva typy paketu jsou *Variable bindings* - jsou to dvojice polí, kde jedna část identifikuje objekt a druhá část je jeho hodnota. Například při dotazu příkazem GET se nastaví název objektu a v odpovědi přijde nastavena i hodnota.

Bezpečnost v této verzi protokolu je založena pouze na takzvaném *community string*u, který vystupuje jako heslo. Existují pouze dvě úrovně zabezpečení přístupu a to - pouze pro čtení (read only) a čtení-zápis (read-write access). Je patrné, že se používají pouze dvě hesla, každé pro jednu úroveň. Je to velice slabé zabezpečení, vezmeme-li v úvahu, že toto heslo se posílá nezašifrované a každý, kdo dokáže odchytit jednotlivé pakety, si může tento řetězec přečíst. Tento nedostatek se pokoušejí odstranit až další verze protokolu.

SNMPv2

Druhá verze protokolu SNMP byla zaměřena na odstranění nedostatků verze první. Bohužel bylo vydáno několik soupeřících specifikací, označované názvy SNMPv2c, SNMPv2u, SNMPv2*, které byly vzájemně nekompatibilní. Nicméně zlepšení oproti první verzi bylo několik. Byly definovány nové datové typy, nové zprávy a zlepšená práce s chybami.

Nové datové typy zahrnují rozšíření podpory z 32-bitových čísel na 64-bitová (Integer32, Integer64, Counter32, ...).

Přidané zprávy jsou:

- **Get Bulk** - tento operátor se snaží efektivněji využít přenosovou kapacitu kanálu tím, že od agenta si vyžádá sérii informací pomocí jediného dotazu
- **Inform** - stejná funkcionalita jako zpráva Trap ve verzi 1, ale nutné je potvrzení od manažera, že zprávu přijal (Response paket)
- **Response** - odpověď na předcházející Inform zprávu (od manažera k agentovi)

Ostatní zprávy SNMPv2 přebírá z předchozí verze a zachovává jejich strukturu. Stejně tak je to i s bezpečností, kde je stále použito heslo ve smyslu community stringu.

SNMPv3

Třetí verze protokolu SNMP je definována sadou standardů, které nepostihují celkovou funkčnost protokolu jako takového, ale dodávají do systému chybějící prvky, hlavně bezpečnosti. Přímo v jednom ze standardů [5] je řečeno, že tato verze může být chápána jako SNMPv2 s dodatečnými administrativními a bezpečnostními schopnostmi.

SNMPv3 definuje tři základní služby:

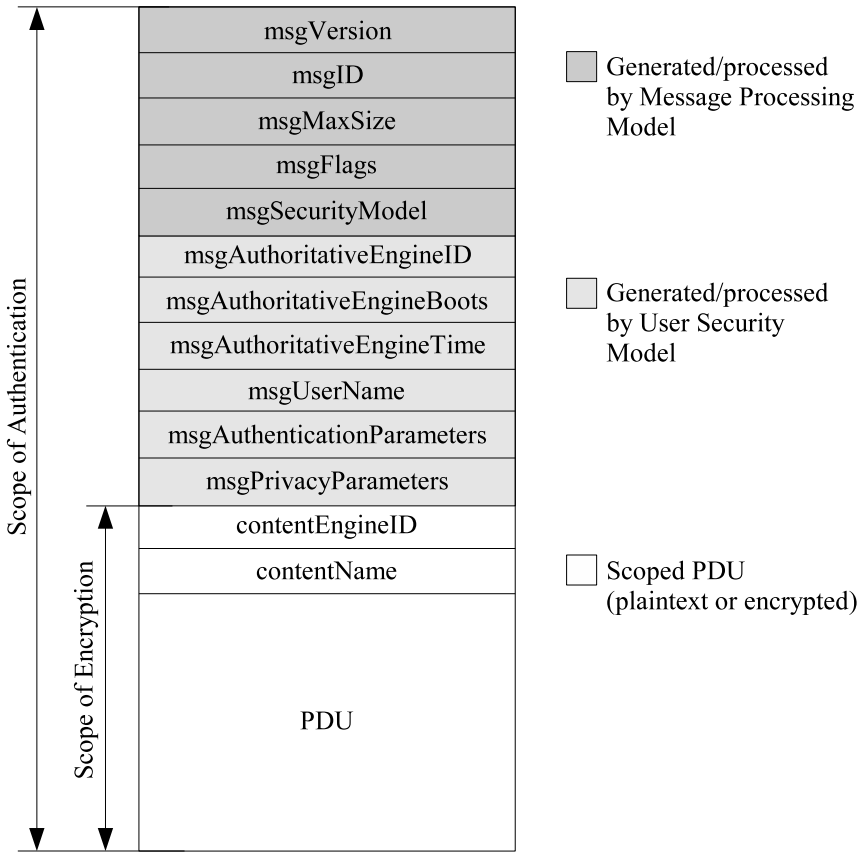
- *Autentifikaci* - datový přenos od manažera k agentovi může být autentifikován, aby se zajistilo ověření identity odesílajícího.
- *Soukromí* - šifrování přenášených zpráv.
- *Přístupová práva* - agent může definovat přístupová práva, omezovat přístup manažerům pouze k některým akcím a částem dat.

Základním principem SNMPv3 je modularita. Každá SNMP entita je tvořena SNMP řídícím systémem a vlastní aplikací. Řídící systém má za úkol přijímat, odesílat, šifrovat a dešifrovat všechny zprávy a dále spravuje a kontroluje monitorované objekty. Tyto funkce jsou poté k dispozici jedné či více aplikací.

Stejně jako předchozí verze, je SNMPv3 založena primárně na transportním protokolu UDP, ale není na něj vázána. Pro přenos dat tak může být použit i jiný protokol. Vlastní aplikační protokol SNMP je rozdělen do dvou úrovní. První zpracovává datové pakety (PDU processing layer) a druhá zpracovává zprávy (message processing layer). Nejvyšší úroveň - PDU processing layer - se stará o zpracování příkazů (Get, Get Next, ...), které přijdou v daném paketu. Zpracovaný paket pak předá nižší úrovni - message processing layer - která tomuto paketu dodá hlavičku, kde jsou uložena bezpečnostní data.

Na obrázku 2.4 je vyobrazen formát SNMPv3 zprávy. První část je tvořena systémem zpracování zpráv. Nese informace ohledně verze protokolu, identifikaci zprávy, maximální délce zprávy a nastavení bezpečnostího modelu. Druhá část je generována bezpečnostním systémem a obsahuje informace o kódování a autorizaci. Třetí část obsahuje samotná data.

Důležitou součástí nového standardu je i systém přístupových práv (VACM - View-Based Access Control Model). Tento model umožňuje nakonfigurovat agenta tak, že specifickému manažerovi bude umožněn přístup pouze k části MIB. Je možné omezit manažera pro přístup pouze k části databáze monitorovaných dat a zároveň ještě omezit operace, které nad touto množinou může provádět. Omezení přístupu se provádí pro definované skupiny, kde součástí jedné skupiny může být více manažerů.



Obrázek 2.4: Schéma datového paketu protokolu SNMPv3 ([1])

Kapitola 3

XML protokol

Pan Ing. Peter Macejko ve své diplomové práci navrh systém vzdálené správy strojů pomocí komunikačního protokolu, využívajícího XML. V této kapitole budou shrnuty všechny navržené postupy od mapování SNMP informačního modelu až, po komunikační struktury využívané správcem a spravovanými zařízeními.

3.1 Obecný informační model

Informační model je nedílnou součástí celého systému správy dat. Do něj jsou mapována veškerá monitorovaná data a jsou zde i vyjádřeny vztahy mezi daty. Ve skutečnosti omezuje počet a druh možných dotazů. Výsledný systém, který byl pro popis jednotlivých zařízení navrženi, vychází z několika různých přístupů abstrakce a popisu dat.

Nejprve byla analýza problému založena na dvou možnostech - přímém mapování MIB stromu do XML dokumentu, kdy by jednotlivé uzly přesně odpovídaly MIB struktuře; objektově orientovaném využívajícím objektové paradigma. První přístup má výhodu ve snadném převodu MIB databáze do nového formátu, ale naopak ztrácí výhodu snadné rozšiřitelnosti, která je vlastní XML technolgoii. Problém objektového mapování je nejednoznačné rozmístění uzlů ve stromu na objekty. Takovéto mapování by bylo nutno provádět neautomatizovaně, tj. za asistence člověka.

Výsledkem analýzy problému je systém využívající kousek od obou přístupů. Na nejvyšší úrovni abstrakce je každé zařízení složeno z modulů. Každý modul obsahuje jistou funkcionalitu, která je úplně oddělena od těch zbývajících. Mezi tyto moduly patří i v této práci navrhovaná brána, která propojuje zařízení bez XML podpory s ostatními částmi sítě. V této chvíli se jedná pouze o obecný návrh každého zařízení.

3.1.1 Odvození typů

Odvozování typů je založeno na principu dědičnosti. Definice jako takové jdou od abstraktního až po detailní popis.

3.1.2 Definice modulů

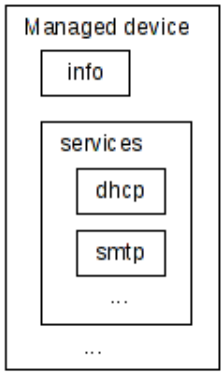
Jak již bylo řečeno, každé zařízení se skládá z modulů. Jednotlivé moduly jsou též popsány XML schématem. Každé takové schéma musí splňovat přesné požadavky na poskytované informace.

Musí být detailně popsána funkčnost, přiděleno unikátní jméno, typ a cesta ve stromové struktuře, použitá pro adresaci jednotlivých uzlů. SNMP moduly mají definován kořenový element, který je využit pro spojování více MIB informačních bází dohromady.

Přesný popis je možné nalézt v [1] v kapitole 5.2.3.

3.1.3 Popis zařízení

Pro popis zařízení je využito XML schéma, stejně jako pro popis dalších částí (modulů apod.). Na obrázku 3.1 je znázorněno, jak vypadá zařízení popsané od nejvyšší úrovně.




```
<xsd:schema ..... >
  <xsd:element name="device" type="thisDeviceType"\>
  <xsd:complexType name="thisDeviceType">
    <xsd:sequence>
      <xsd:element name="info" type="basicEl:infoType">
      <xsd:element name="services">
        <xsd:sequence>
          <xsd:element name="dhcp" type="ISC:dhcpServerType"/>
          <xsd:element name="smtp" type="pfix:smtpServerType"/>
        <xsd:sequence>
      <xsd:element>
      ...
    </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
```

Obrázek 3.1: Popis zařízení v XML schématu ([1])

3.1.4 Oznamovací zprávy

Oznámení jsou takové zprávy, které jsou zasílány manažerovi v případě, že se na monitorovaném zařízení vyskytne nějaká událost (shodné s SNMP Trap zprávami). V rámci SNMP jsou tyto zprávy součástí datového modelu, nicméně tyto specifické uzly MIB stromu nenesou žádná data, a jsou tudíž použité pouze při generování typu chyby či události.

V navrženém systému jsou všechna možné upozornění (ať již předdefinované, či definované administrátorem) umístěny ve speciálním uzlu stromu `notifications`, kde je velice jednoduché dohledat, jaké události mohou způsobit zaslání oznamuvací zprávy. Každý modul pak může mít specifikován speciální typ `NotificationType`, který popisuje právě onu událost.

3.1.5 Adresace dat

Pro adresaci dat je možno využít postupů XPath a XQuery. Jednotlivými výrazy ať už v jednom či druhém případě se bude manažer dotazovat na jednotlivé uzly v rámci spravované databáze.

3.2 Mapování MIB do XML

V předchozí části byl rozebrán čistě obecný model popisu zařízení. Pro monitorované stroje, které nejsou kompatibilní s XML protokolem, musí existovat brána, která bude překládat dotazy z jednoho protokolu na druhý a stejně tak i odpovědi. Je tedy nutné přesně definovat postup přepisu MIB na XML.

Je nutné vyřešit tři základní problémy - jak importovat jednotlivé MIB do sebe; jak předefinovat datové typy a jak konvertovat celý MIB strom.

3.2.1 Importy

V rámci jednotlivých MIB jsou časté odkazy na báze vyšší úrovně, kdy pak na nižších úrovních definujeme jenom část podstromu. V rámci XML budou definovány odkazy jako prostory jmen, které jsou odvozeny od názvu daného MIB. Odkaz na jiné schéma bude proveden použitím odkazů na typ s příslušným názvem prostoru jmen.

3.2.2 Datové typy

V SNMP, jak bylo řečeno v předchozí kapitole, existuje několik druhů datových typů. Jednoduché (integer, string,...), aplikačně rozšířené (Gauge, IpAddress,...) a uživatelem definované.

Jednoduché typy budou mapovány na jejich XML ekvivalent. Na obrázku 3.2 je soupis všech aplikačně rozšířených typů a jejich popis pomocí XML schématu (v rámci standardu SMIv1).

Součástí SMI je i možnost definovat vlastní typy. I pro tyto případy je nutno uvést definici překladu. Existují tři základní omezení při vytváření vlastních typů - výčet, délka řetězce a rozmezí hodnot. Všechny tyto typy jsou detailně popsány a vyobrazeny v [1], kapitola 5.3.1.

3.2.3 MIB strom

Navržený systém využívá při mapování celého stromu oddělení definicí typů od samotné struktury stromu. Typy jsou definovány globálně a zároveň separátně od struktury a to z důvodu možného použití typů v rámci jiného modulu a zároveň při omezení přístupových práv do dané oblasti stromu. V MIB jsou objekty definovány makry (specifikované v

SMI	XML Schema
NetworkAddress	<xsd:simpleType name="NetworkAddress"> <xsd:restriction base="xsd:string"/> </xsd:simpleType>
IpAddress	<xsd:simpleType name="IPAddress"> <xsd:restriction base="xsd:string"> <xsd:pattern value=" (([1-9] ?[0-9] | 1[0-9][0-9] | 2[0-4][0-9] | 25[0-5]) \.){3} ([1-9] ?[0-9] | 1[0-9][0-9] | 2[0-4][0-9] | 25[0-5]) "/> </xsd:restriction> </xsd:simpleType>
Counter	<xsd:simpleType name="Counter"> <xsd:restriction base="xsd:unsignedInt"/> </xsd:simpleType>
Gauge	<xsd:simpleType name="Gauge"> <xsd:restriction base="xsd:unsignedInt"/> </xsd:simpleType>
TimeTicks	<xsd:simpleType name="TimeTicks"> <xsd:restriction base="xsd:unsignedInt"/> </xsd:simpleType>
Opaque	<xsd:simpleType name="Opaque"> <xsd:restriction base="xsd:string"/> </xsd:simpleType>

Obrázek 3.2: Mapování aplikačních typů SMIv1 do XML schématu ([1])

```
...
    <xsd:element name="NodeName" type="MIBName:NodeNameType"/>
...
<xsd:simpleType name="NodeNameType">
  <xsd:annotation>
    <xsd:documentation xml:lang="en">DescrText</xsd:documentation>
    <xsd:appinfo>
      <status>StatusType</status>
      <access>AccessType</access>
      <oid>AbsoluteOID</oid>
    </xsd:appinfo>
  </xsd:annotation>
  <xsd:restriction base="NodeType"/>
</xsd:simpleType>
```

Tabulka 3.1: Mapování makra OBJECT-TYPE, jednoduchý typ (SMIv1) ([1])

```
...
    <xsd:element minOccurs="0" maxOccurs="unbounded"
             name="NodeName" type="MIBName:NodeNameType"/>
...
<xsd:complexType name="NodeNameType">
  <xsd:sequence>
    <xsd:element name="..child.." type="..childType.."/>
    ...
  </xsd:sequence>
</xsd:complexType>
```

Tabulka 3.2: Mapování SEQUENCE, makro OBJECT-TYPE ([1])

```
...
  <xsd:element name="atTable">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ..SEQUENCE.. />
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
...
```

Tabulka 3.3: Mapování SEQUENCE OF, makro OBJECT-TYPE ([1])

SMI), které popisují několik základních typů uzlů. SMIv1 specifikuje OBJECT-TYPE a TRAP-TYPE makra.

OBJECT-TYPE makro definuje uzel, který obsahuje nějaká data. Může to být samotná hodnota, položka, nebo celá řádka tabulky. Mapování pak závisí na položce `SyntaxType` v samotné definici makra.

Pakliže je hodnota položky základním, rozšířeným či uživatelsky definovaným typem, bude vytvořena globální definice typu a položka bude tvořena elementem s jednoduchým typem. Schematicky vyjádřeno v tabulce 3.1.

Jestli bude hodnotou `SEQUENCE`, bude vytvořen "řádkový" typ (tabulka 3.2).

Hodnota `SEQUENCE OF` pak vyjadřuje množinu řádkových typů (tabulka 3.3).

Dalším typem objektu jsou upozornění definované pomocí TRAP-TYPE makra. Tyto definují uzly bez hodnot, pouze specifikují danou událost. V navrženém systému tedy nemusí být součástí stromu, ale pouze globálních typových definicí. Bude použit jednoduchý typ popisující čas a den (datetime type) se speciálním elementem v části `appinfo`.

```
<xsd:simpleType name="NodeNameType">
  <xsd:annotation>
    <xsd:documentation xml:lang="en">DescrText</xsd:documentation>
    <xsd:appinfo>
      <enterprise>EnterpriseName</enterprise>
      <variable>VariableType</variable>
      <reference>ReferenceType</reference>
      <trapNumber>TrapNumber</trapNumber>
    </xsd:appinfo>
  </xsd:annotation>
  <xsd:restriction base="xsd:dateTime"/>
</xsd:simpleType>
```

Tabulka 3.4: Mapování TRAP-TYPE makra ([1])

3.3 Zprávy

Navrhovaný systém by měl využívat spolehlivého a potvrzovaného přenosového protokolu, na rozdíl od nepotvrzovaného SNMP. Zaroveň by mělo být možno přenášet zprávy v co nejjednodušším formátu. Proto bylo rozhodnuto o použití protokolu HTTP, který využívá přenosový protokol TCP, čímž je zajištěn spolehlivý přenos. Všecha data se budou přenášet pomocí HTTP zprávy POST.

HTTP je bezestavový protokol. Veškerá komunikace se sestává z dvojice dotaz a odpověď. Na serveru se neudržují jakékoliv další informace ohledně probíhajícího spojení. Tato nenáročnost dovoluje implementaci na velice různorodém hardwaru.

Bezpečnost přenosu může být řešena za použití tunelování paketů (IPSec, STunel,...), nebo je možno využít výhody HTTPS (HTTP over SSL).

Veškerá přenášená data budou ve formátu XML dokumentu s kořenovým uzlem `message`. Tento uzel má několik atributů, které specifikují jeho zpracování a přístupová práva. Jsou to `queue`, `password`, `context`. První atribut určuje frontu (může být založeno na prioritním zpracování), ve které bude požadavek zpracován. V principu ale nejsou agenti ani brány povinni takovouto funkčnost implementovat. Zprávy pak budou zpracovány sekvenčně a odpovědi budou generovány v přesném pořadí tak, jak přišly dotazy. Zbylé dva atributy slouží pro vymezení přístupu uživatele (`context`) na určitý podstrom dat.

Bylo již naznačeno, že zpráva může obsahovat několik jednotlivých dotazů. Struktura zprávy je vyjádřena na obrázku 3.3.

Dotazy a odpovědi, které definují komunikaci mezi manažerem a klientem, jsou popsány níže. Přesné XML schéma definující úplnou strukturu zpráv obsahuje ([1], příloha D).

```
<message context="honza">
  <get msgid="123">...</get>
  <set msgid="234">...</set>
  <get msgid="2222">...</get>
</message>
```

Obrázek 3.3: Struktura XML zprávy

DISCOVERY

Tato zpráva je první, kterou zašle manažer agentovi, aby zjistil, jaká monitorovaná data jsou k dispozici. Povinným atributem je číslo verze protokolu (`protocolVersion`) a nepovinným je `fullDescription` pro bližší specifikace typů spravovaných dat.

```
<message context="honza">
  <discovery protocolVersion="1.0" msgid="123" />
</message>
```

PUBLICATION

Agentova odpověď na manažerův dotaz DISCOVERY. V rámci zprávy je uvedeno, jakou verzi protokolu agent používá a jaká data spravuje. Tato data jsou pak manažerem zpracována a použita jako informační model.

```
<publication msgid="123">
  <info>
    <xpath>1.0</xpath>
    ...
  </info>
  <dataModel>
    ... XML schema popisující spravovaná data ...
  </dataModel>
</publication>
```

Pakliže agent nepodporuje danou verzi protokolu, musí odpovědět chybovou zprávou:

```
<publication msgid="123">
  <error code="1">Protocol not supported</error>
</publication>
```

GET

Tímto dotazem se manažer ptá agenta na hodnotu nějakého uzlu. Pro specifikaci jakého je nutno použít XPath či XQuery.

```
<get msgid="123">
  <xpath>
    device/data/interface
  </xpath>
</get>
```

SET

Zpráva SET je určena pro nastavení hodnoty uzlu. Struktura je podobná zprávě GET, ale obsahuje navíc element `value`.

```
<set msgid="123">
  <xpath>
    device/data/interface/status
  </xpath>
  <value>4</value>
</set>
```

RESPONSE

Odpověď na zprávy GET a SET. V případě GET nese zpráva příslušná data. Pakliže je to odpověď na SET, je to pouze potvrzení, že hodnota uzlu byla úspěšně nastavena.

```
<response msgid="123">
  <value>4</value>
</response>

<response msgid="123" />
```

EVENT

Pro oznamování asynchronních událostí je tu zpráva EVEN (stejná funkcionalita jako TRAP u SNMP). Přenášené informace specifikují, která událost vyvolala toto oznámení, kdo to poslal, datum a čas, případně nějaká další data, která by mohla být při řešení problému užitečná.

```
<event msgid="123" timestamp="" senderID="router1" eventSpec="/device/notifications/dhc
  <data>
    <value valueLocation="/data/services/dhcp/leases/free">0</value>
    <value valueLocation="/data/services/dhcp/leases/used">50</value>
  </data>
</event>
```

Je nutné, aby doručení této zprávy bylo potvrzeno. Což bude dodrženo použitým protokolem.

SUBSCRIBE

Touto zprávou se manažer přihlásí k opakovanému zasílání dat. Potvrzením je pak první doručení dat - zpráva DISTRIBUTION - nebo chybové zprávy, že je něco v nepořádku. Je možné specifikovat ještě nepovinný atribut `frequency` - doba ve vteřinách, po které mají být opakovaně zasílány zprávy. Další nepovinné atributy `distrid` a `delete` jsou využity pro editaci či smazání daného přihlášení.

```
<subscribe msgid="123" frequency="150">
  <xpath>/device/data/interface/status</xpath>
</subscribe>
```

DISTRIBUTION

Zpráva obsahuje data, o která si manažer řekl. Je nutné, aby odesílaná data byla ve stejném pořadí, ve kterém byla ve zprávě SUBSCRIBE. Povinný atribut `distrid` je určený k identifikaci příchozích dat u manažera.

```
<distribution msgid="123" distrid="5678">
  <value>1</value>
  <valuea>500</value>
</distribution>
```

Příjem těchto dat je též nutné potvrdit, což zajistí transportní protokol.

Kapitola 4

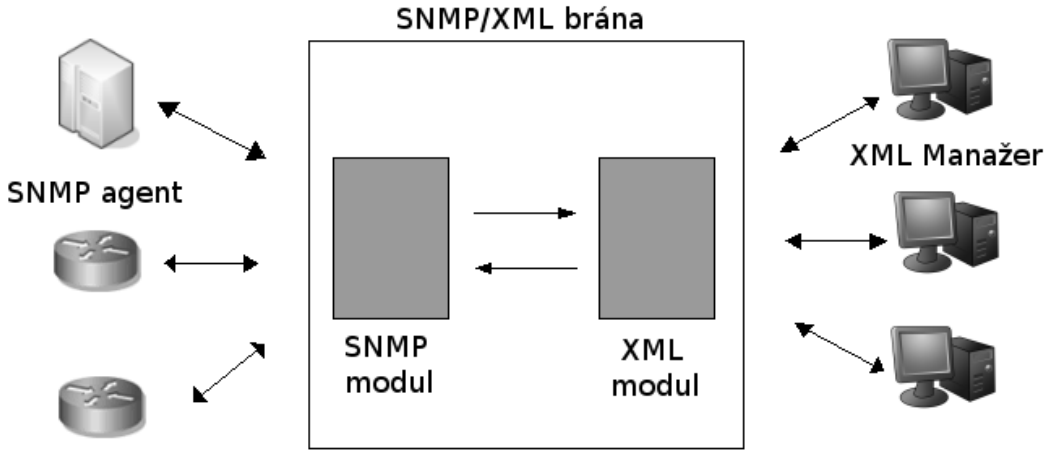
Návrh systému

V předchozích dvou kapitolách byla rozebrána teoretická část problému. V této kapitole shrneme požadavky vyplývající z teorie, které je nutno zakomponovat do výsledného systému. Nejprve bude schématicky vyjádřena obecná funkcionalita systému, která se následně bude rozebírat detailněji.

4.1 Teoretické požadavky

Nároky na systém, které vyplývají z teorie můžeme rozdělit do tří částí - implementace SNMP protokolu, implementace navrženého XML protokolu a propojení těchto dvou protokolů dohromady.

Hlavním požadavkem, který vyplývá i ze zadání práce, je vytvořit modulární systém, který bude nejenom spojovat současné verze protokolů, ale bude počítat i s potenciálním rozšířením do budoucna. Obecné schéma navrhovaného systému zobrazuje obrázek 4.1.



Obrázek 4.1: Schéma navrhovaného systému

Zde je vidět, že oba dva protokolové moduly jsou na sobě nezávislé a jejich interakce spočívá v předávání si zpráv. Nyní přejděme k detailnějším požadavkům na výše zmíněné části systému.

V rámci *SNMP protokolu* je požadováno

- implementace komunikačních struktur protokolů SNMPv1 a SNMPv2
- převzetí bezpečnostního schématu z tohoto protokolu

XML orientovaná část programu má za úkol

- implementovat komunikační struktury navrženého protokolu
- navrhnout efektivní správu XML struktur v paměti
- poskytnout XML manažerům transparentní získání dat z monitorovaných zařízení
- mapovat rozšířenou množinu funkcí v rámci XML protokolu do SNMP
- s manažery komunikovat pouze přes HTTP/HTTPS protokol

Spojením protokolů je myšlen přechod od databázových struktur jednoho protokolu k druhému. V našem případě je to transformace SNMP MIB do XML, jak bylo vysvětleno v kapitole 3.

4.1.1 XML

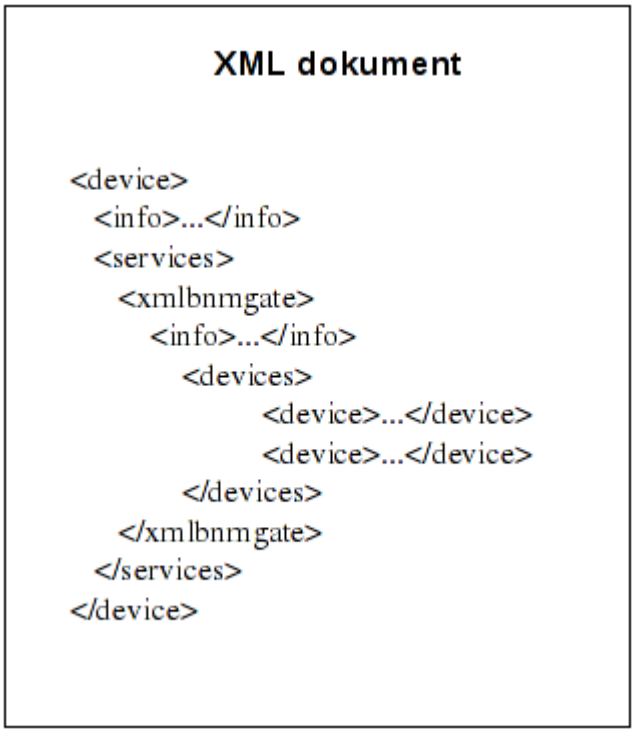
Nejprve se zaměříme na reprezentaci dat, které budou v rámci XML popisovat jak bránu, tak monitorované zařízení. Z předchozích kapitol vyplynulo, že bude použito částečně objektového přístupu a přímého mapování MIB. Strukturu dat bude popisovat XML dokument, strom, který má strukturu vyjádřenou na obrázku 4.2.

Kořenový uzel specifikuje celé zařízení vystupující jako protokolová brána, obsahuje tyto elementy:

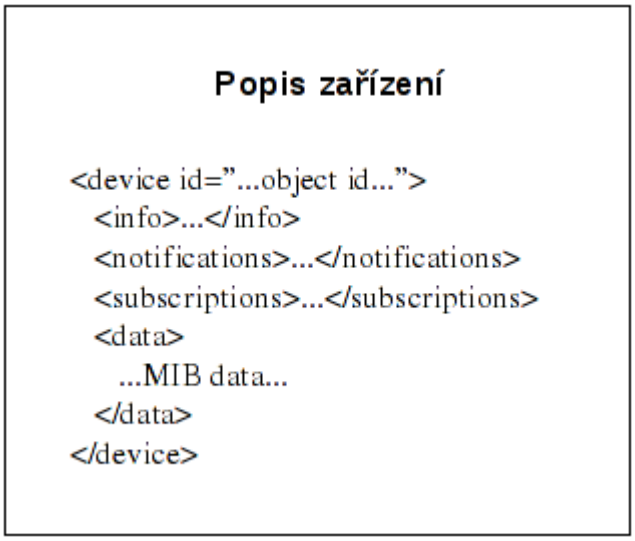
- **info** - tento element obsahuje text, kterým je popsáno dané zařízení
- **services** - element vymezující poskytované služby (při širší implementaci může obsahovat služby DNS, DHCP, apod.)
- **xmlbnmGate** - naše služba poskytující spojení XML a SNMP protokolu
- **device** - je podelementem **xmlbnmGate** a vymezuje jedno monitorované zařízení

Prvky **device** jsou do XML dokumentu přidávány na základě informací v konfiguračním souboru (viz kapitola 4.2).

Strukturu elementu **device** popisuje obrázek 4.3. Každý takovýto element bude obsahovat následující informace:



Obrázek 4.2: Obecná struktura XML dokumentu

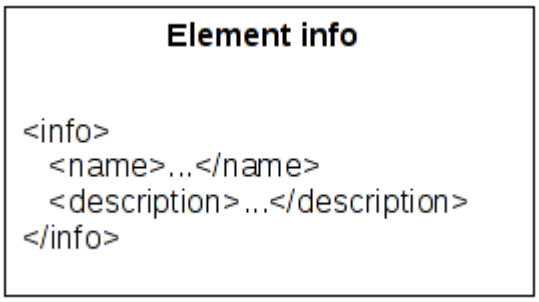


Obrázek 4.3: Struktura elementu device

- **info** - stejně jako kořenový element popisuje dané zařízení
- **notifications** - obsahuje elementy a typy upozornění (TRAP zprávy v rámci SNMP), na které manažer čeká
- **subscriptions** - obsahuje informace o datech, které si nechává manažer posílat v pravidelných intervalech (více v popisu komunikace)
- **data** - dětmi tohoto elementu jsou veškerá data přímo z MIB.

Samotný element má atribut *id*, což je jeho identifikace v rámci xml dokumentu. Dle tohoto unikátního čísla je pak možné v sadě dotazů rozpoznat, ke kterému zařízení se dotaz vztahuje.

Element **info** obsahuje elementy, které specifikují jméno a popis zařízení (viz obrázek 4.4).



Obrázek 4.4: Struktura elementu info

Jednotlivé podelementy uzlu **subscriptions** musí z podstaty věci obsahovat informace, které určují, jaké objekty chce manažer pravidelně sledovat, identifikovat manažera, aby mu mohla být data doručena a specifikovat časový interval, tj. frekvenci sledování příslušné veličiny.

Potomci uzlu **notifications** určují, které typy událostí jsou sledovány u daného zařízení. V rámci konfigurace systému je nezbytné, aby pro každé zařízení bylo jasně definováno, kam mají být příslušné zprávy o událostech zasílány. Tato informace bude součástí konfiguračního souboru a systém ji bude interně zpracovávat. Samotný XML tag bude obsahovat unikátní identifikační číslo (OID) dané události, aby jej bylo možno poté identifikovat a správně formulovat XML zprávu manažerovi.

Mapování dat z MIB bylo obecně popsáno v kapitole 3 a přesný algoritmus bude specifikován v následující kapitole. Pro adresaci jednotlivých objektů je, jak bylo již nastíněno v předchozí kapitole, použito mechanismů XPath či XQuery. Dotaz na položku z MIB může vypadat následovně

```
/iso/org/dod/internet/mgmt/mib-2/...
```

Důležitým faktem je, že manažer se ptá přímo na uzly datového stromu. Nemusí tedy uvádět cestu v rámci celého stromu, který popisuje strukturu brány.

Zprávy

Zprávy, které budou posílány mezi manažerem a bránou, mají formu XML dokumentu. Schématicky je znázorněna a popsána v kapitole 3, obrázek 3.3.

Kořenový element message obaluje veškerá posílaná data. Může obsahovat několik dílčích dotazů, nastavení a ostatních informací, které budou vykonávány postupně jedna po druhé. V rámci teorie byla nastíněna možnost použití několika různých front, které by byly specifikovány identifikátorem a zaručovaly by různou prioritu zpracování. Navrhovaný systém bude podporovat rozdělení komunikačních front dle monitorovaných zařízení. Zajistí se tím správné pořadí vykonání příchozích požadavků.

Komunikace mezi manažerem a bránou je na XML úrovni omezena na zprávy

- Get
- Set
- Discovery
- Publication
- Subscription
- Distribution
- Event

Přesná struktura a popis funkce jednotlivých zpráv byla popsána v předchozí kapitole.

Komunikační protokol

Od protokolu SNMP se XML část komunikace liší také tím, že bude probíhat na spolehlivém a potvrzovaném protokolu - HTTP. Každá zpráva, která je poslána, musí mít potvrzeno doručení, což tento aplikační protokol, využívající transportního protokolu TCP, nabízí.

Informace budou posílány ve formátu HTTP POST zprávy. Strukturu dotazu a odpovědi zobrazuje obrázek 4.5 a komunikaci obrázek 4.6.

Otázka bezpečného přenosu dat byla řešena v předchozí kapitole a byl zvolen protokol HTTPS. Zajištění distribuce a zpracování certifikátů bude diskutováno dále v této kapitole.

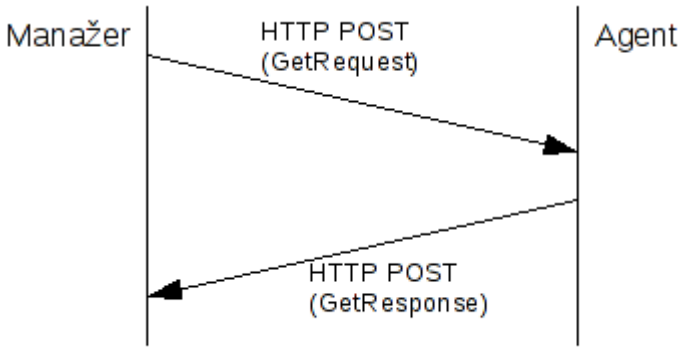
HTTP dotaz

```
POST /xmlbnm HTTP/1.1
Host: hrocht2.org
Content-Type: application/xml

<message context="tst">
   <get msgid="123">
      ...
   </get>
   ...
</message>
```

HTTP odpověď

```
HTTP/1.1 200 OK
Content-Type: application/xml

<message>
   <response msgid="123">
      ...
   </response>
   ...
</message>
```

Obrázek 4.5: HTTP zprávy předávané mezi manažerem a bránou



Obrázek 4.6: HTTP komunikace mezi manažerem a agentem/bránou

4.1.2 SNMP

Druhou část komunikace tvoří SNMP protokol. Z kapitoly 2 vychází seznam zpráv, které je nutné implementovat:

- Get
- Set
- Response
- GetNext
- Trap

V rámci komunikace se v této práci budeme zaobírat verzemi SNMPv1 a SNMPv2. Samotná implementace a mapování SNMP zpráv na XML dotazy bude diskutována až v kapitole 5.

Bezpečnost se v SNMP omezuje pouze na komunitní heslo. V rámci XML protokolu je bezpečnost založena jednak na šifrovaném přenosu dat mezi manažerem a bránou, druhak na přístupovém heslu, které vymezuje čtecí či zápisová práva při přístupu k zařízení.

4.2 Struktura programu

Před samotným návrhem jednotlivých funkčních elementů je nutno zvolit, jak bude program fungovat a jevit se globálně. Vzhledem k nabízeným službám a komunikaci je možné zvolit koncepci podobnou webovým službám (založených na principu SOAP). Druhým možným přístupem je zvolit na pozadí běžící aplikaci - démona, který bude po celou dobu svého běhu monitorovat a zpracovávat příchozí požadavky.

4.2.1 SOAP vs. démon

Kompozice struktury jako webové služby založené na SOAP architektuře má několik předností. Je tím hlavně přenositelnost a jednoduchost nasazení. Vše, co je potřeba k běhu, je aplikační server. Nainstalování a spuštění služby je již pak otázkou okamžiku. Samotná struktura kódu je též o něco jednodušší než v případě démona. Je nutné se starat pouze o příchozí požadavek a jeho zpracování.

Bohužel tento přístup má ale i mnoho nevýhod. Zaprvé je to reakční doba, za kterou je systém schopen zaslat manažerovi odpověď. Pro samotné zpracování požadavku je nutno v paměti (či v souboru) udržovat XML reprezentaci MIB tak, jak bylo popsáno dříve. Při použití tohoto postupu se po každém přijatém požadavku musí načíst informace ze souboru a teprve poté je možno data zpracovat.

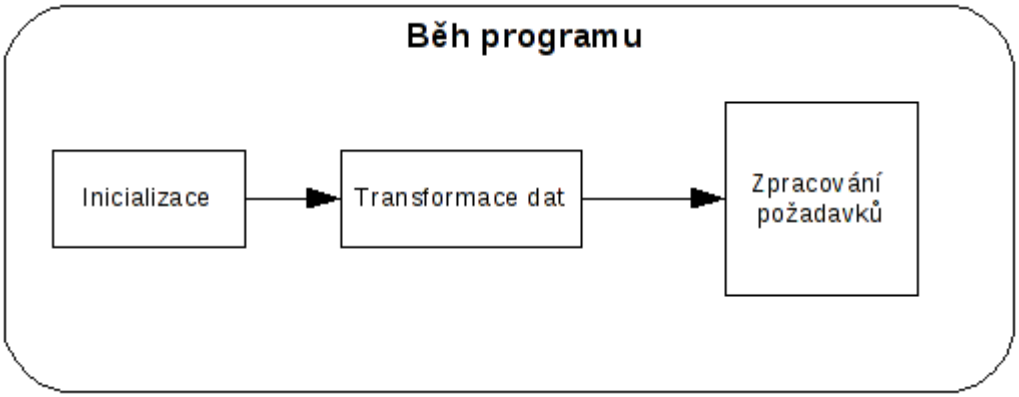
Dalším sporným bodem je periodické zasílání zpráv manažerům, kteří o to požádali zprávou Subscription. V takovém případě musí běžet jeden proces, který v určených intervalech zasílá SNMP dotazy na monitorované zařízení, což je neslučitelné se základní myšlenkou webových služeb.

Asi největší nevýhodou tohoto přístupu je transformace dat z MIB do XML. Při spuštění webové služby je nutné, aby všechna zařízení již měla své monitorované informace uloženy v XML formátu, protože při požadavku již není čas data transformovat. Tento akt by musel být od služby oddělen a buď svěřen periodicky se spouštěnému skriptu, nebo by jej administrátor musel provést pokaždé, když se změní počet, druh či monitorované údaje jednotlivých zařízení.

Oproti tomu stojí druhý přístup - strukturovat aplikaci jako démona. Je pravdou, že výsledný kód aplikace je složitější, přenositelnost horší a není zde možné mluvit o platformové nezávislosti (co se implementace v C++ týče). Nicméně získáme tím výhodu v podobě relativně malé reakční doby, protože veškeré informace jsou za běhu uloženy v operační paměti a není je nutno načítat z pevného disku. Systém periodického monitorování zařízení může být jednoduše spravován jedním vláknem procesu, zatímco ostatní vlákna se starají o příchozí a odchozí požadavky. Transformace dat pak může být bez úhony součástí samotného programu.

4.2.2 Navrhovaná aplikace

Fáze běhu navrhovaného systému zobrazuje diagram na obrázku 4.7.



Obrázek 4.7: Fáze běhu programu

4.2.2.1 Inicializace

V první, inicializační, fázi je načten konfigurační soubor, jehož specifikace bude popsána v kapitole 5. Tento soubor obsahuje veškeré informace o monitorovaných zařízeních, stejně tak jako základní nastavení protokolové brány. Každé zařízení musí mít definováno SNMP

spojení (adresu), seznam MIB, které vyjadřují všechny nabízené informace. Dále bude obsahovat nastavení ohledně asynchronních událostí a jakému manažerovi je nutno je přeposílat. Důležitým nastavením je i verze SNMP protokolu, jakou zařízení podporuje.

Protokolová brána bude mít sama o sobě speciální část, která bude definovat komunikační porty, na kterých budou přijímány a zpracovávány požadavky, cesty k různým logovacím souborům a cesty k adresáři s MIB a XML soubory.

Součástí inicializace je i ověření, zda-li všechna monitorovaná zařízení fungují. Pakliže některé nebudou funkční, systém je ze seznamu vyškrtne a nebude je nabízet manažerům ke správě.

4.2.2.2 Transformace dat

Transformace dat představuje samotné mapování MIB do XML tak, jak bylo popsáno dříve. Pro každé zařízení může být specifikováno několik různých MIB, jak veřejné, tak proprietární. Pro každé zařízení je tedy nutno vytvořit XML dokument, popisující veškeré MIB informace.

V tomt místě jsou možné dvě cesty, jak vybudovat výstupní XML dokument. Jednou možností je zahrnout veškeré informace o všech zařízeních do jednoho souboru, který potom bude rozesílán každému manažerovi, jenž si o něj řekne. Tato varianta je sice praktická, ale neefektivní. Pro zařízení s velkým množstvím informací by byl výsledný dokument opravdu veliký. Kdyby manažer chtěl spravovat pouze jediné zařízení, byl by stejně nucen stáhnout velký objem dat, než by mu bylo dovoleno pracovat dále.

Proto bude použito následujícího schématu. Systém bude při prvním kontaktu s manažerem publikovat pouze informace týkající se počtu a typu zařízení, které spravuje. Jednotlivá zařízení budou mít svůj samostatný soubor s daty. Manažer si pak bude moci zvolit pouze určitá data, která ho zajímají. Tím se velmi sníží počáteční zatížení linky.

4.2.2.3 Zpracovávání požadavků

Po úspěšném průchodu oběma předchozími fázemi se program dostává do situace, kdy vyčkává na příchozí požadavky, ať již ze strany manažerai, či SNMP zařízení.

Jak již bylo popsáno na začátku této kapitoly, o komunikaci se starají dva moduly - SNMP a XML. Proto taky komunikační rozhraní se dělí na dvě části.

SNMP modul bude komunikovat pomocí protokolu UDP, posíláním SNMP zpráv. Tato část rozhraní bude blíže popsána v následující kapitole.

Komunikace v rámci XML je na bázi HTTP protokolu. Pro zpracování mnoha požadavků, které je nutno očekávat, bude použito HTTP serveru. V tomto případě se nám nabízí dvě možnosti řešení - využijeme nějakého již stávajícího webového serveru (Apache, Tomcat, ...), na kterém budeme spouštět CGI script a tak komunikovat s naším programem, nebo do aplikace nějaký jednoduchý server naimplementujeme. Výhodou již existujícího řešení by byla pouhá konstrukce komunikačního kanálu mezi protokolovou

branou a zmíněným serverem. Je ale pravděpodobné, že bude potřeba mít větší kontrolu nad přijímanými a odesílanými zprávami, což vlastní implementovaný server poskytuje. Přednostně tedy bude vybrána varianta s embedded HTTP serverem.

Ve spojitosti s protokolem HTTP je nutné zmínit použití certifikátů pro zabezpečený přenost a použití protokolu HTTPS. Kdyby bylo využito externího webového serveru, bude ponechána veškerá zodpovědnost a konfigurace na administrátorovi serveru, který se bude muset postarat o obdržení a distribuci certifikátu. Jestli bude server součástí protokolové brány, bude nutno přiložit certifikát k aplikaci a v konfiguračním souboru zajistit jeho použití.

4.3 Správa protokolové brány

V rámci spravovaných zařízení se naskýtá otázka, jestli by bylo možno spravovat i samotnou bránu přes navržený XML protokol (je myšlena aplikace jako taková).

Navržený systém tuto skutečnost neumožňuje. Samotná brána nebude vykazovat vlastnosti agenta. Ke správě stroje, na kterém brána poběží, bude nutné použít XML či SNMP agenta, který tuto funkcionalitu bude zajišťovat. Je možné pak v konfiguračním souboru nastavit, aby brána nabízela komunikaci se SNMP agentem na tomtéž stroji. XML agent bude komunikovat s manažerem přímo na definovaném portu. Implementace takového agenta ale již přesahuje rámec této práce.

Správa aplikace je pak omezena pouze na konfigurační soubor a bude ji nutné při každé změně restartovat (jak je tomu například i u konfigurace webových serverů). Je to z důvodu zachování integrity poskytovaných dat. Při změně informačních bází jednotlivých zařízení, přidání či odebrání monitorovaných strojů, je nutno přegenerovat všechny XML dokumenty, které jsou pak distribuovány manažerům. Nekorektnost dat, které manažer obdržel a které by byly aktuální, kdyby se změnily za plného provozu, by mohla mít pak vážné následky na data, která by manažeři dostali zpět.

4.4 XML manažerská aplikace

Součástí této práce je i implementace základního XML manažera tak, aby dovolil ukázat veškeré funkční aspekty protokolové brány.

Program bude mít implementován celý XML komunikační protokol tak, jak byl navržen v kapitole 3. Bude podporovat základní příkazy - Discovery, Publication, Get, Set, Subscription, Distribution.

Správu monitorovaných zařízení zahajuje komunikací buď přímo s agentem či bránou. Vyžádá si od nich dokument, který popisuje nabízené informace. Pakliže se jedná o komunikaci s bránou, tak nejprve zjistí, jaká zařízení jsou k dispozici a pak si některé vybere a teprve pak požádá o jejich XML popis dat. Algoritmus budování XML stromu použitelného pro další interakci se zařízeními je stejný jako v případě brány a bude popsán v další kapitole.

Aplikace bude napsána v jazyce C++ stejně jako protokolová brána. Využití podpory grafického rozhraní je možné.

Kapitola 5

Implementace

Vlastní implementace protokolové brány je v této kapitole vysvětlena v rámci několika částí. Nejprve je popsána struktura programu, popis tříd, jejich vztah mezi sebou a hlavní funkce, kterou plní. Následuje přehled všech použitých knihoven pro lepší pochopení dalšího popisu funkce aplikace. Další část tvoří vysvětlení výkonného cyklu programu krok za krokem od spuštění až po zpracování jednotlivých zpráv. Jsou vysvětleny dosažené výsledky při paměťové optimalizaci a detailně rozebrán systém komunikačních front. Jako poslední je popsán XML manažer, který byl navržen v předchozí kapitole.

5.1 Třídy, vlastnosti a jejich vztahy

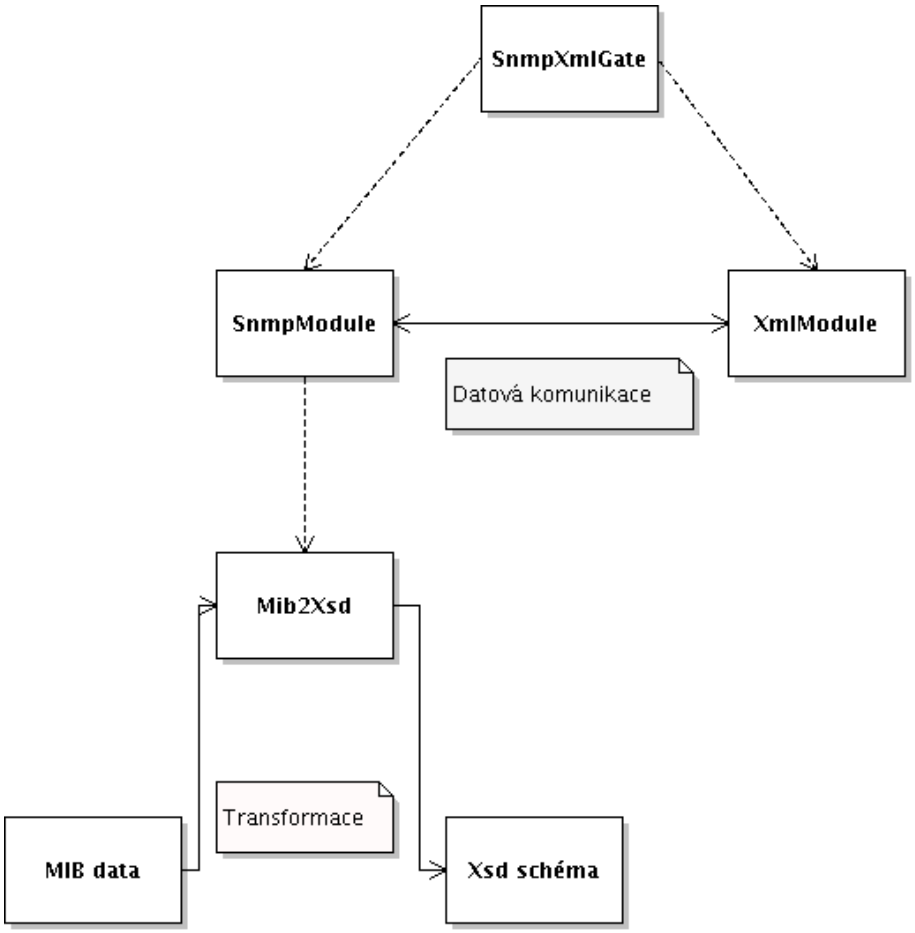
Jak bylo předesláno v předchozí kapitole, při tvorbě byl dáván velký důraz na modularitu jednotlivých součástí programu. V praxi to znamená, že jednotlivé komunikační jednotky brány nemají tušení, jak ta druhá provádí konkrétní operace, ale jediným spojením jsou přesně definované struktury obsahující veškeré nutné informace.

Strukturu programu tvoří čtyři třídy:

- **SnmpXmlGate** - hlavní třída, která ovládá běh programu. Při spouštění inicializuje veškeré datové struktury, připraví oba komunikační moduly a předá jim řízení.
- **SnmpModule** - komunikační modul pro zpracování veškerých SNMP požadavků. Plní dvě funkce. První je získávání dat z agentů v reakci na dotaz od manažera (resp. XML modulu). Druhou je zpracování asynchronních událostí (Trap, Notification), které se na agentech vyskytnou a získané informace zašle definovaným manažerům.
- **Mib2Xsd** - transformační třída pro překlad MIB databáze do XML schématu. Netvoří však jenom tento popis, ale zároveň i vytváří finální XML dokument, se kterým pak hlavní program pracuje.

- **XmlModule** - druhý komunikační modul pro zpracování XML požadavků. Tato třída je vlastní implementací navrženého XML protokolu a je pro tuto práci stěžejní.

Vzájemné vztahy těchto tříd jsou vyjádřeny na obrázku 5.1.



Obrázek 5.1: Schéma tříd aplikace

5.2 Použité knihovny

Při tvorbě programu bylo použito několik knihoven, které zajišťují potřebné využívané funkce.

- ***Xerces-C++ a Xalan-C++*** - knihovny zajišťující manipulace s XML dokumenty, vyhledávání pomocí XPath výrazů ([6], [7]).
- ***Net-SNMP*** - knihovna napsaná v jazyce C, určená pro interpretaci a manipulaci s daty MIB databází, zasílání a přijímání datových SNMP paketů ([8]).

- ***libMicroHTTPD*** - volně dostupný HTTP server s podporou použití SSL protokolu při spojení ([9]).
- ***libCUrl*** - volně dostupná knihovna určená pro klientskou stranu spojení. Podporuje mimo jiné protokoly HTTP, FTP a umožňuje použít i SSL certifikáty pro zabezpečení daného spojení ([10]).

Knihovny Xerces, Xalan, libCUrl a libMicroHTTPD využívá XML modul, který má na starosti běh HTTP serveru, operace s XML dokumenty a periodické zasílání informací.

SNMP modul využívá knihoven Net-SNMP a libCUrl. Druhou jmenovanou využívá vlákno pro zpracování asynchronních událostí (viz dále v této kapitole).

5.3 Popis výkonného cyklu

Detailní popis funkcí jednotlivých tříd je součástí popisu fungování protokolové brány. Jak bylo napsáno v předchozí kapitole, v rámci běhu programu je několik přesně definovaných fází, ve kterých se systém může nacházet. Tyto jsou vyobrazeny na obrázku 4.7. Některé z těchto fází byly oproti návrhu lehce pozměněny, či rozšířeny, dle potřebného rozsahu splňovaných funkcí.

5.3.1 Inicializační fáze

Je důležité připomenout, že již v návrhu bylo stanoveno jako optimální řešení, že program bude běžet jako démon. Tomu odpovídá i manipulace s programem. Byl vytvořen spouštěcí skript, který ovládá běh programu - spouští, zastavuje či restartuje (popis instalace, struktura spouštěcího skriptu a použití je v příloze B). Jediným vstupním parametrem systému je konfigurační soubor, který je formátován jako dokument XML (přesná struktura souboru se všemi povolenými a nutnými elementy je v příloze A). Tento obsahuje veškeré nutné informace pro běh systému.

Konfigurační soubor

Informace jsou strukturovány do elementů, kdy každý popisuje jedno spravované zařízení. Každé zařízení je nutné identifikovat unikátním číslem, které bude použito později při komunikaci s manažery. Dále je nutné specifikovat SNMP přístupové informace k agentovi:

- IP adresu či url
- verzi SNMP protokolu
- MIB báze, které popisují zařízení
- přístupová hesla (community string) pro zápis a čtení

Poslední položkou, která je čistě volitelná, je možné definovat manažery, kteří obdrží informace o asynchronních událostech. Každý manažer je definován IP adresou či url a portem, na který se budou zasílat jednotlivé zprávy.

Specifickým elementem je definice samotné brány. Tento element obsahuje veškeré informace výše popsány, ale jsou přidány ještě povinné elementy:

- **logFile** - identifikuje soubor, do kterého budou ukládány veškeré textové výstupy
- **snmp** - obsahuje informace o cestě k souborům definujícím MIB databáze a portu, na kterém budou poslouchány asynchronní události
- **xml** - definuje xml modul systému. Je zde verze XML protokolu; přístupová práva pro čtení a zápis; cesta k ukládání XSD popisu zařízení; porty pro poslouchání požadavků a odesílání odpovědí.
- **security** - obsahuje případné informace o certifikátu a klíči použitém při přístupu přes protokol HTTPS.

Zpracování a ověření informací

Poté, co je bezchybně zpracován vstupní soubor, je přistoupeno ke zpracování informací ohledně jednotlivých zařízeních. Tuto fázi má na starosti třída *SnmpModule*. Nejprve je ověřeno, jestli dané zařízení vůbec funguje a je na něm přítomen SNMP agent. Odešle se tedy základní požadavek a je očekávána jakákoliv odpověď. Když je agent aktivní, je zařazen do seznamu spravovaných zařízení. Jinak systém tuto položku vynechá a nebude ji dále brát v úvahu.

Následuje zpracování seznamu MIB souborů, které popisují nabízené informace. Tyto soubory je nutné vlastnit a přiložit je do dříve specifikovaného adresáře, odkud si je systém načte a později zpracuje. Jestli všechny soubory existují a jsou načteny, zařízení je finálně "odsouhlaseno" (viz schéma 5.2).

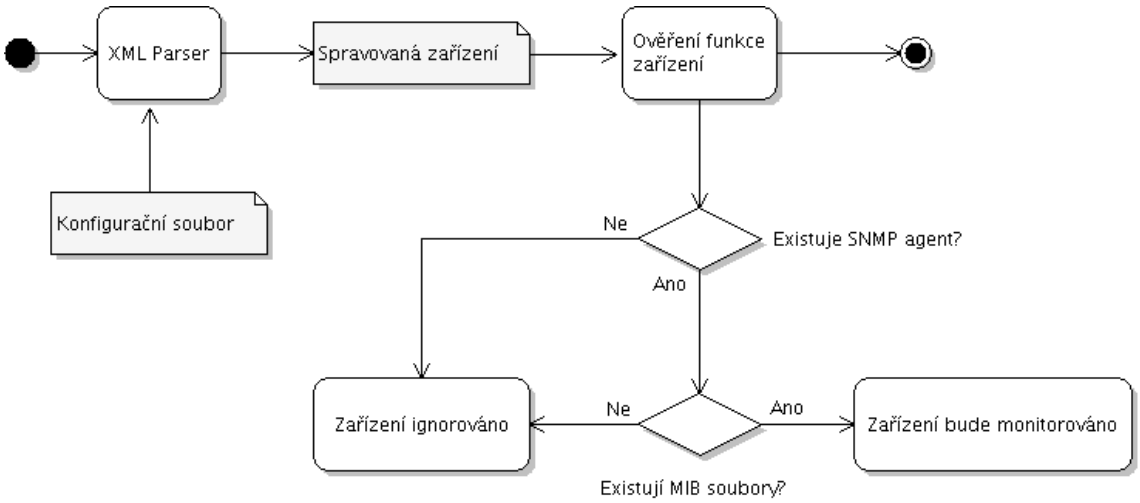
Po zpracování údajů o všech zařízeních, inicializační fáze končí a systém započne fázi transformační.

5.3.2 Transformační fáze

V této chvíli přebírá úlohu třída ***Mib2Xsd***, která se stará o převod MIB popisu databáze do XSD formátu. Pro manipulaci se SNMP je použita knihovna ***net-snmp*** ([8]).

Samotný převod a generování XSD popisu je rekurzivní sestup po jednotlivých uzlech virtuálního stromu MIB databáze.

Pro každé zařízení systém načte všechny specifikované MIB soubory a započne s transformací. Proces se řídí pravidly, které byly definovány v kapitole 3. Jednotlivé uzly jsou popsány svým typem, který obsahuje jejich unikátní identifikační číslo (OID), typ dat a přístupová práva. Samotné uzly jsou pak zařazeny do stromové hierarchie v rámci dokumentu.



Obrázek 5.2: Inicializační fáze

Informace, vztahující se k jednotlivým zařízením, jsou generovány do oddělených souborů. Samostatný soubor pak obsahuje schéma brány a pouze identifikaci spravovaných zařízení. Tento systém byl vytvořen pro ušetření komunikační zátěže mezi manažerem a bránou. Bližší popis výhod tohoto přístupu dále v této kapitole.

Oproti navrženému systému mapování MIB do XSD ([1]) byla vynechána nutnost použití prostorů jmen pro jednotlivé MIB databáze. Vzhledem k tomu, že každý element je popsán unikátním jménem a OID, nedojde při zpracování požadavků ke konfliktu.

Abychom ušetřili procesorový čas, je paralelně s generováním schématu vytvářen i XML dokument, který za běhu slouží k vyhledávání a ostatním potřebným operacím. Oproti schématu vypadá element specifikující uzel v MIB databázi jako na obrázku 5.3.

Tento dokument je pak spravován a využíván třídou ***XmlModule***.

5.3.3 Komunikační moduly

Po úspěšném zpracování a trasformaci MIB databází je možné přistoupit k inicializaci komunikačních jednotek a otevření síťových spojení k bráně. V této fázi předává třída ***SnmpXmlGate*** řízení třídám ***SnmpModule*** a ***XmlModule*** . Tyto moduly pak zajišťují kompletní provoz protokolové brány.

XmlModule

Dle návrhu programu v kapitole 4 bylo uvedeno, že pro komunikaci mezi bránou a manažery bude použit HTTP server zabudovaný do aplikace, abychom měli větší kontrolu nad posílanými daty. Proto byl použit nejvhodnější kandidát - microHTTP server ([9]). Tento server je napsán v jazyce C.

```
<element>
  <element restriction="string" access="read-
only" />
    ...
</element>
```

```
<internet>
  <mgmt>
    <mib-2>
      <system>
       <sysDescr restriction="string"
access="read-only" />
        ...
     </system>
    </mib-2>
  </mgmt>
</internet>
```

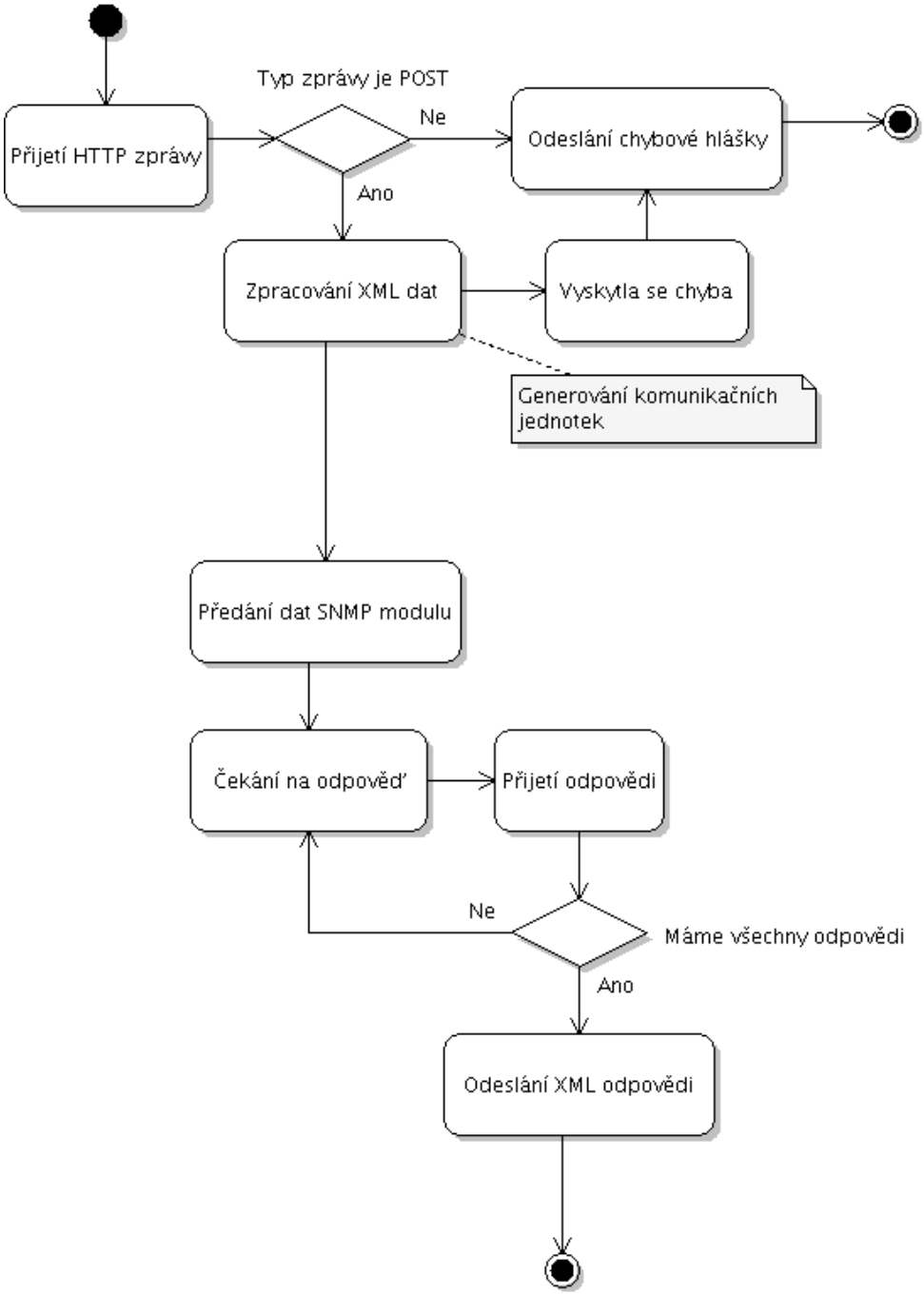
Obrázek 5.3: Příklad struktury hlavního XML dokumentu

Jednotlivá spojení jsou zpracovávána samostatnými vlákny, aby bylo zaručeno co nejrychlejší odbavení požadavků. Funkční cyklus vlánka je znázorněn na schématu 5.4.

Nejprve je zpracována HTTP zpráva. Systém operuje pouze s HTTP zprávami typu POST. Jakékoliv GET zprávy jsou zahazovány a zpět klientovi je zasláno chybové upozornění. Poté je na data použit XML parser, který zpracuje požadavek a ujistí se, že je ve formátu specifikovaném v kapitole 3. Všechny zprávy jsou zpracovávány najednou, aby byla zaručena integrita dat a "atomicita" operací. Více v kapitole 5.3.5. Přesné zpracování jednoho požadavku je popsáno dále v této kapitole.

Pro komunikaci vláken s druhým modulem byly vytvořeny fronty požadavků (pro směr od XML modulu k SNMP modulu) a jedna odpovědní fronta, ze které se poté vybírají vyřízené požadavky. Když jsou všechny zprávy v pořádku zpracovány, jsou odeslány do fronty k příslušnému monitorovanému zařízení, nebo jsou rovnou zařazeny jako odpovědi (obsahují-li chybu, která znemožňuje její další zpracování). Zařazení do fronty k vláknu má na starosti SNMP modul a zároveň i on probouzí příslušné vlákno ke zpracování požadavků.

Vlákno se následně přesune do čekajícího cyklu, kde kontroluje odpovědní frontu na svoje požadavky do doby, než se dostaví všechny odpovědi. Poté vygeneruje odpovědní zprávu a zašle manažerovi. Je na serveru, jak s vlákny poté naloží a jak je recykluje. Takzvané zamrznutí a nekonečné čekání vlákna je ošetřeno maximální prodlevou pro jeden požadavek v SNMP modulu. Tím je zaručeno, že manažer určitě dostane odpověď.



Obrázek 5.4: Zpracování manažerského požadavku

Jednotka komunikace

Než přistoupíme k popisu zpracování jednotlivých XML zpráv, je důležité popsat, jak spolu komunikují oba moduly. Z navrženého protokolu jasně vyplývá, že XML manažer komunikuje s bránou tak, jako by to byl samotný XML agent. O protokolu SNMP nesmí vědět. Proto byl již při návrhu brán zřetel na striktní oddělení těchto dvou protokolů. Pro komunikaci byla navržena struktura, která v sobě obsahuje veškeré informace o požadavku a odpovědi, ale je protokolově neutrální.

Důležité položky struktury, které oba moduly využívají jsou:

- **typ zprávy** - definuje, jaký požadavek byl vyslán (Get, Set). Dle toho se pak vytvářejí SNMP požadavky na agenta.
- **seznam uzlů** - seznam dvojic (jméno, hodnota), které manažer požaduje.
- **chyba** - identifikace a textová reprezentace chyby, která v celém zpracujícím procestu nastala.

Tato struktura pak obsahuje informace o jednom požadavku (Get, Set, Subscribe), který byl v XML zprávě obsažen.

SnmpModule

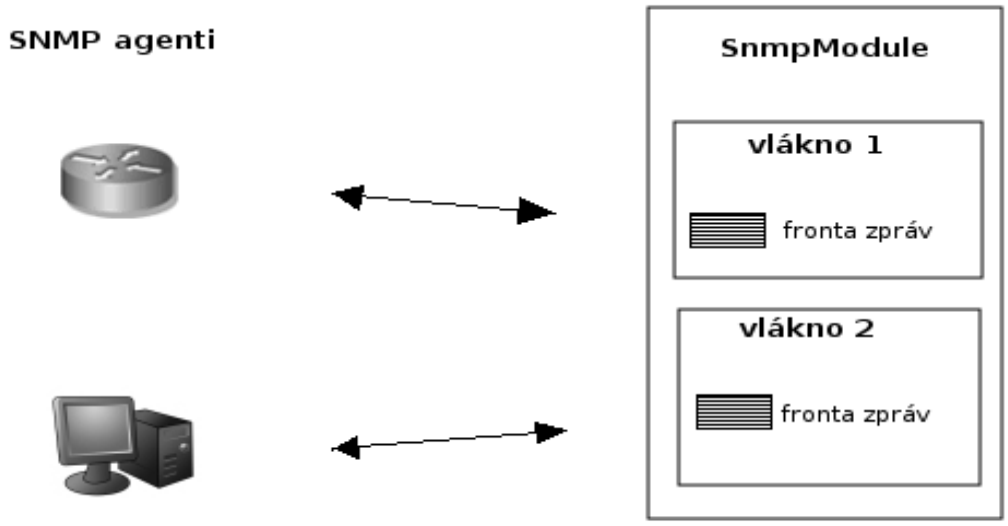
Komunikace se SNMP agentama je založen na systému front. Každé monitorované zařízení má na straně protokolové brány k dispozici jedno obslužné vlákno a frontu zpráv. Schématicky naznačeno na obrázku 5.5. Zpracování a vytváření SNMP zpráv se děje pomocí knihovny **net-snmp**.

Práce komunikačních vláken probíhá v "nekonečném" cyklu. Vstupní informací pro vlákno je identifikační číslo monitorovaného zařízení, ke kterému patří. S touto informací získá přístup k frontě zpráv a dalším potřebným strukturám, příslušející pouze danému agentovi. V cyklu se pak opakuje několik dílčích bloků stále dokola.

Vlákno zjistí, jestli v příslušné frontě je přítomen nějaký požadavek na vyřízení. Když není, vlákno se uspí. Jinak vyjme všechny dotazy, tím frontu vyprázdní a požadavky zpracuje. Tento funkční blok závisí na typu zprávy, která se má vyřídit.

Zprávy Get a Set, které mají za úkol dostat nebo nastavit hodnotu koncového uzlu stromu, plně korespondují se SNMP. Je vytvořen SNMP packet, do kterého je vloženo heslo (community string) na základě přístupových práv, které manažer má, je naplněno daty, které se mají nastavit či získat a dotaz je odeslán.

Po přijetí odpovědi jsou data opět z paketu vyjmuta, upravena do požadovaného formátu v rámci komunikační struktury. Pakliže se vyskytla chyba při zpracování, opět je nastaven chybový příznak ve struktuře, aby manažer dostal informace o problému. Takto vygenerovaná odpověď je poslána do odpovědní fronty a je probuzeno vlákno, které na data čeká (samotná funkce je ve správě XML modulu).



Obrázek 5.5: Komunikační vlákna brány pro spojení se SNMP agenty

Složitější je zpracování požadavku na uzel, který není koncový a obsahuje pod sebou celý podstrom dalších uzlů (z hlediska MIB databáze). Poté je nutno opakovat vytváření paketů a zasílání dotazů ve formě SNMP zprávy GetNextRequest. Až poté, co jsou získána všechna data, je navrácena odpověď klientskému vláknu v XML modulu.

5.3.4 Zprávy

Zpracování jednotlivých druhů zpráv se odvíjí od navrženého modelu. Následuje popis chování systému při přijetí jednotlivých požadavků.

Discovery

Jediným omezením při zpracování požadavku je verze XML protokolu, kterou manažer posílá ve zprávě. Když se rozchází s nastavenou verzí v systému, odpovědí je zpráva Publication a chybové hlášení.

Jedním z implementačních požadavků bylo též snížit množství dat mezi manažerem a bránou. Předpokládáme, že brána spravuje desítky SNMP zařízení. Celý dokument, popisující všechny informace, který je odpovědí na tento požadavek by pak nabyl obrovských rozměrů. Pro zlepšení protokolu tak systém zašle manažerovi pouze hlavní popis brány s identifikací jednotlivých zařízení, bez ostatních dat. Manažer si může vybrat, které zařízení chce spravovat a zašle novou zprávu Discovery s volitelným atributem **objectid**. Po zpracování mu brána zašle pouze XSD dokument popisující právě to jedno zařízení.

Get a Set

Zpracování těchto požadavků prochází několika fázemi v rámci samotného XML modulu.

Nejprve je nutné zjistit, jestli má manažer právo tyto operace provádět. Heslo, které se zasílá v elementu *message*, určuje, jestli je povoleno čtení, či zápis. Nastavení hesel probíhá při inicializační fázi, kdy jsou čtena z konfiguračního souboru. Celý přístupový model je založen na dvou heslech XML protokolu - pro čtení a zápis. Každé SNMP zařízení má pak definována opět dvě hesla, která jsou odlišná od těch prvních jmenovaných. Systém na základě dodaného XML hesla zjistí, jaké má uživatel práva a použije příslušné SNMP heslo pro požadavek. Může se stát, že mangažer zadal špatné heslo a systém mu přístup odepře. Pak je odpovědí na tyto požadavky chybová zpráva.

Pokud uživatel má dostatečné oprávnění, je nutné zjistit, zda-li uzly, na které se dotazuje, opravdu existují. Dotazování probíhá pomocí jazyka XPath. Pokud je dotaz špatně formulován, nebo element neexistuje, systém požadavek zamítne. V druhém případě mohou nastat dvě eventuality.

Element je koncovým uzlem stromu, tj. obsahuje pouze hodnotu danou definovaným typem. Systém takový dotaz interpretuje jako jediný SNMP dotaz GET. Když se ale jedná o kořenový uzel (obsahuje podstrom uzlů), je nutné tento dotaz interpretovat jako SNMP zprávu GETNEXT a získat všechny hodnoty daného podstromu (o což se stará SNMP modul).

Jestli bylo požadavkem nastavení určité hodnoty, pak je součástí datové struktury jméno uzlu a příslušná hodnota.

Vygenerovaná komunikační jednotka je vložena do fronty požadavků a pak současně se všemi ostatními odeslána SNMP modulu ke zpracování.

Subscribe

Touto zprávou se manažer upisuje k zasílání pravidelných informací o daném agentovi, nebo tyto údaje mění a maže. Každý takový záznam se zanese do hlavního XML dokumentu, aby bylo při úpravách a mazání možné vyhledávat pomocí XPath výrazu. Reakce na přidání a úprava vnitřních struktur je ponechána na speciálním vláknu, které slouží k obsluze daných záznamů. Bližší specifikace dále v této kapitole.

Jediný problém při implementaci nastal v rámci mazání jednotlivých záznamů. Vzhledem k tomu, že v navrženém protokolu je odpovědí manažerovi zpráva Distribution s daty, která požaduje, nebo chyba, je nutno při smazání záznamu poslat opět zprávu Distribution, ale nyní prázdnou.

Práva k mazání a úpravám daného záznamu jsou ponechána pouze na znalosti unikátního identifikátoru daného záznamu.

Změny oproti navrženému XML protokolu

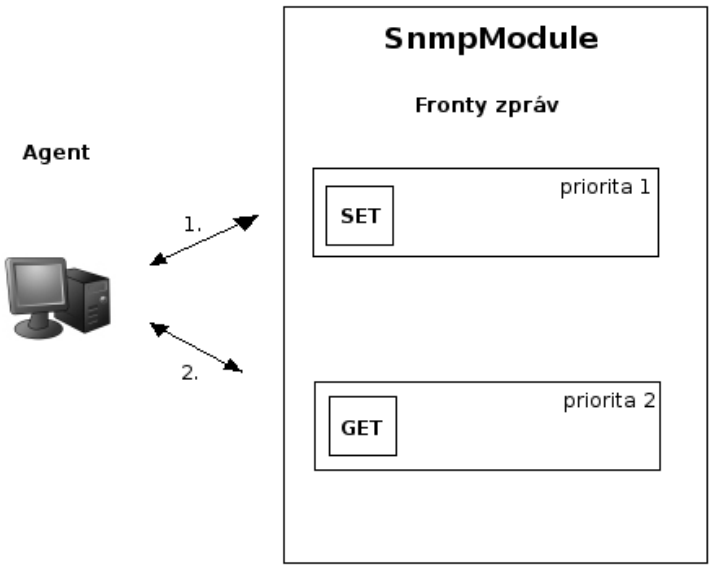
Oproti navrženému XML protokolu bylo nutno udělat několik drobných změn, abychom mohli celý systém implementovat. Jedná se pouze o minoritní změny - přidání volitelného atributu **objectid** ke všem typům požadavků, které manažer zasílá. Je to z důvodu identifiace monitorovaného zařízení, což by bez tohoto zásahu nebylo vůbec možné. Zároveň tento zásah umožňuje ponechat stejnou strukturu protokolu (co se týče dotazů a odpovědí) pro stroj agenta a brány.

Abychom nezměnili dosavadní protokol, byl tento atribut zvolen jako volitelný. Jeho nepřítomnost signalizuje, že dotaz je mířen na samotné zařízení brány, která následně přepošle požadavek na SNMP agenta běžícího na tom samém stroji.

5.3.5 Fronty a atomicita požadavků

Systém předávání požadavků pomocí front vychází z navrženého protokolu ([1]), ale byl přepracován pro potřeby této protokolové brány. Navrhovány byly fronty, které by zpracovávaly požadavky dle priority a mamažer by si mohl říci, které dotazy budou mít jakou prioritu. To bohužel zcela nevyhovuje požadavku zajištění posloupnosti provádění dotazů a tím pádem i integritě dat.

Předpokládejme situaci, kdy manažer v jedné XML zprávě zašle dotaz na hodnotu jednoho uzlu s nižší prioritou a zároveň také nastavení jiné hodnoty s vyšší prioritou (viz obrázek 5.6). Stalo by se, že při zpracovávání se nastavení provede dříve a manažer by se nikdy nedozvěděl starou hodnotu.



Obrázek 5.6: Chyba prioritního zpracování požadavků

Druhým problémem je "atomicita" prováděných operací. Kdyby systém zpracovával jeden požadavek z přijaté zprávy, čekal na odpověď agenta a pak přešel na další, mohlo

by se stát, že druhý manažer by mohl do tohoto cyklu vstoupit a narušit tak výsledek celé posloupnosti operací.

Proto byl systém přepracován tak, že každé monitorované zařízení má v SNMP modulu jedno obslužné vlákno a frontu požadavků (jak bylo vysvětleno dříve v této kapitole). Nyní se v XML modulu zpracují všechny požadavky v rámci jedné zprávy, vygenerují se příslušné komunikační struktury a ty jsou následně všechny najednou vloženy do příslušné fronty požadavků. Tato funkce je ponechána na SNMP modulu, který zajistí, pomocí systému výlučného přístupu k dané frontě, aby všechny zprávy tvořily souvislý blok, který bude sekvenčně zpracován.

Je možné, že uživatel v rámci jedné zprávy zašle dotazy na více zařízení. Každé takové zařízení obdrží svůj blok dotazů individuálně.

Předávání odpovědí ze SNMP do XML modulu slouží pouze jediná odpovědní fronta. Do ní se vkládají postupně vyřízené požadavky ze všech zařízení. Vždy, když je vložena odpověď, jsou probuzena klientská vlákna, zkontrolují frontu a případně si vyberou jim určené odpovědi. Zde by se mohlo zdát, že bude narušena posloupnost jednotlivých odpovědí v případě, ze komunikujeme s více zařízeními. Tento fakt je ošetřen unikátním identifikátorem každého požadavku v zasílané zprávě. Odpovědi poté obsahují stejný identifikační řetězec, díky kterému pak manažer pozná, k jaké zprávě se vztahuje obdržená hodnota.

5.3.6 Periodické zasílání informací

Jednotlivými zprávami typu Subscribe se manažer upíše k pravidelnému zasílání informací. Tuto činnost zahrnuje v sobě XML modul, kde na obsluhu těchto rozesílání je vyhrazeno speciální vlákno. To se stará o veškeré manipulace s jednotlivými záznamy, získávání informací od agentů (pomocí SNMP modulu) a jejich předávání dále.

Celý systém správy záznamů je založen na hlavním XML dokumentu. Každé spravované zařízení má ve své části element *subscriptions*, kam jsou ukládány informace o jednotlivých manažerech. Jeden příkladný záznam je schematicky vyjádřen na obrázku 5.7.

```
<subscriptions>
  <subscription distrid="1" frequency="4" manager="172.16.1.10">
    <xpath>/iso/org/dod/internet/mgmt/mib-2/tcp</xpath>
    ...
  </subscription>
  ...
</subsriptions>
```

Obrázek 5.7: Element subscription v rámci hlavního XML dokumentu

Každý záznam je identifikován jedinečným číslem a zároveň i zařízením, ve kterém se nachází. Dále element subscription obsahuje informace o frekvenci zasílání, url manažera,

kam se mají pakety doručovat a v poslední řadě jsou to jednotlivé XPath výrazy pro identifikaci uzlů z databáze dat. Veškeré úpravy - přidávání, změna a mazání - jsou ponechány ve správě tohoto vlákna.

Přidání záznamu

Když manažer zašle požadavek na zapsání, je obsluhujícím vláknem vytvořen nový element *subscription*, naplněn daty, umístěn do XML dokumentu a distribuční vlákno je upozorněno, že nastala změna. Jsou prohlédnuty všechny záznamy a vygenerovány interní datové struktury k správné obsluze.

Úprava záznamu

Vlákno, které zpracovává požadavek klienta, upraví daný záznam v XML dokumentu a oznámí distribučnímu vláknu změnu. To se pak, identicky jako v předchozím případě, přesvědčí, jaké změny nastaly a upraví interní data.

Smazání záznamu

Záznam, pakliže existuje, je označen jako smazaný, ale vlastní smazání provádí až distribuční vlákno.

Zasílání informací

Vlastní zasílání informací probíhá v jednoduchém cyklu. Jestli není jediný záznam k dispozici, vlákno se uspí a čeká na příchozí zprávy Subscribe. Jsou-li záznamy k vyřizování, zvolí se nejmenčí časový krok, který je nutný čekat (dle frekvence zasílání jednotlivého záznamu) a vlákno se uspí. Po probuzení je odečten prospaný čas ode všech záznamů a u těch, které jsou připraveny k vyřízení, předá distribuční vlákno připravené komunikační jednotky SNMP modulu, pro vyřízení vlastní komunikace. Poté, co dostane odpověď, inicializuje spojení pomocí knihovny ***libCUrl*** a zprávu Distribution zašle manažerovi na příslušnou adresu.

Pakliže zpráva nedojde z jakéhokoliv důvodu - manažer je vypnutý, nemůže být zahájeno spojení - je započítán neplatný pokus o odeslání. V momentu, kdy tento počet překročí stanovenou hranici, je daný záznam smazán a manažer bude muset požádat o nové zapsání zprávou Subscribe. Tento systém je zvolen pro zamezení zahlcení distribučního vlákna, které by pak v extrémním případě mohlo spravovat desítky až stovky nefunkčních záznamů, které si nikdo nevyzvedne.

5.3.7 Asynchronní události

Přijímání a zpracování veškerých asynchronních událostí, které agenti posílají na protokolovou bránu, má za úkol speciální vlákno ve správě SNMP modulu. Funkce tohoto vlákna je přijmout SNMP zprávu (Trap, Notification), zjistit o jaké zařízení se jedná, zvolit všechny manažery, kteří jsou nastaveni pro příjem XML zprávy Event a zprávu odeslat.

Veškeré nastavení monitorovaných zařízení z hlediska příjemců těchto zpráv je v konfiguračním souboru. Při inicializaci vlákna je zpracován tento seznam pro každé zařízení.

Vlákno se pak přesune do nekonečného cyklu, kde čeká na definovaném portu na příjem zprávy. Příjem a získání SNMP paketu z příslušného datového spojení je ponecháno na knihovně ***net-snmp***.

Samotné zpracování pak v sobě zahrnuje získání informací o události, inicializaci jednotlivých spojení pomocí knihovny ***libCUrl*** a zaslání XML zprávy Event jednotlivým manažerům. Z navrženého protokolu vyplývá povinnost získat potvrzení o doručení paketu. To za nás vykoná transportní protokol TCP, který je knihovnou libCUrl použit. Není-li možné paket doručit - manažer neexistuje či je vypnutý - pak je informace zahozena a brána se již o tento fakt nezajímá.

5.4 Paměťové optimalizace

Hlavním bodem zadání této práce není jenom implementace protokolové brány, ale i optimalizace využití operační paměti. Při vývoji se objevily dva body, kdy bylo nutno vymyslet optimalizaci systému. Oba dva body se opírají o manipulaci a uložení XML dokumentů, které v paměti zabírají velký prostor.

Prvním je správa XSD schémat, která popisují jednotlivá monitorovaná zařízení. Každou příchozí zprávou Discovery se manažer dotazuje na tato popisná schémata. Bylo by, pro rychlejší odbavení požadavku, snadnější uchovávat celá schémata v paměti. Za předpokladu, že jedno schéma zabírá minimálně několik stovek kB (při použití pouze standardní sady MIB databáze) paměti a že bychom spravovali několik desítek zařízení, mohla by se nutná paměť rozrůst do desítek MB. Proto se každé takovéto schéma ukládá do souboru na pevném disku a v případě potřeby je soubor načten a zaslán manažerovi. Časový rozdíl je relativně malý v porovnání s množstvím paměti, která se ušetří.

Druhým bodem je minimalizace velikosti hlavního XML dokumentu, který je udržován po celou dobu chodu protokolové brány v paměti. Není dost dobře možné, bez zásadních změn navrženého protokolu, sdílet části virtuálního stromu dat. Proto vycházejme z předpokladu, že na spravované síti reálně existují identická zařízení, která používají stejnou sadu MIB databází. Zároveň je tu druhý předpoklad, že v rámci běžné sítě se nepoužívá mnoho jednotlivých zařízení, která by v rámci SNMP protokolu nabízela ke správě odlišná data. Kdybychom měli u každého takového zařízení spravovat v hlavním XML dokumentu celý datový strom, nebylo by možné udržet tento v paměti a zároveň na něm provádět efektivně vyhledávání a jiné operace.

Proto byl stanoven takzvaný sdílený systém, který se dotýká i prvního bodu. Zařízení, která mají v konfiguračním souboru nastavenu stejnou sadu MIB databází, jsou prohlášena za shodná, co se týče dat, a je vytvořeno pouze jedno popisné XSD schéma a do hlavního dokumentu je uložen pouze jeden takovýto strom. Ostatní zařízení mají nastaveno identifikační číslo shodného agenta. Při jakémkoliv požadavku na vyhledání elementů ve stromu či popisné schéma, je vyhledáváno právě v tom jediném sdíleném stromu, nebo navráceno jediné sdílené schéma.

Tímto nejenom, že v rámci předpokladů ušetříme aktuální operační pamět, ale zároveň i místo na pevném disku, kde jsou popisná schémata ukládána. Z hlediska splnění zadání bylo tedy dosaženo optimálního řešení.

5.5 XML Manažer

Nedílnou součástí implementační části je vytvoření XML manažera, který by dovolil otestovat protokolovou bránu v praxi. Vzhledem k tomu, že není náplní práce vytvořit plnohodnotného klienta, který by postkytoval veškeré uživatelské pohodlí grafické aplikace a jiných možných aspektů, byl vývoj omezen na tvorbu aplikace ovládané pomocí příkazového řádku.

K tvorbě manažerské aplikace byly použity knihovny:

- ***libCUrl*** - zajišťující klientské spojení s HTTP serverem na bráně ([10])
- ***libMicroHTTPD*** - HTTP server použitý pro příjem asynchronních událostí a periodické distribuce dat ([9])
- ***libargtable2*** - knihovna zajišťující pohodlné zpracování vstupních parametrů programu ([11])

Aplikace funguje ve dvou možných formách - aktivní nebo pasivní (kompletní instalační a uživatelská příručka je v příloze C).

Aktivní mód

V tomto módu je součástí vstupních parametrů URL protokolové brány, XML heslo a soubor, který obsahuje jednotlivé požadavky, které chce uživatel provést. Systém načte vstupní soubor a pakliže je vše bez chyby zpracováno, vygeneruje XML zprávu dle navrženého protokolu.

Tuto zprávu pak odešle a čeká na odpověď. V případě, že byla odeslána zpráva Subscribe a jsou tedy očekávány periodické dodávky informací, je spuštěn HTTP server a jako v předchozím případě, aplikace čeká na odpovědi.

Při ukončení tohoto čekání zašle aplikace XML zprávu, kterou zruší veškeré Subscribe záznamy, které předtím odeslala.

Pasivní mód

Tento mód slouží jako takzvaný "poslouchací". Při něm se pouze aktivuje webový server a očekávají se příchozí zprávy ohledně asynchronních událostí. V tomto módu nelze odesílat žádné požadavky.

Kapitola 6

Testování

Cílem této práce bylo vytvoření funkčního prototypu protokolové brány v jazyze C++. Proto veškeré testování bude zaměřeno na ověření funkce a prověření reakceschopnosti na jednotlivé manažerské požadavky.

Testovací systém

Síť agentů spojená s bránou, na které byl program testován, se sestává ze dvou autonomních systémů. Prvním je notebook, na kterém je nainstalován SNMP agent. Druhým systémem je zařízení brány, na kterém kromě samotné protokolové aplikace běží též SNMP agent.

Testy se sestávají z ověření funkce pomocí odeslání jednotlivých požadavků dle navrženého protokolu a obdržení odpovědí. V každé zprávě je nejprve uvedeno, co manažer odeslal a následuje výpis, co brána odpověděla.

Test zprávy Discovery

Dotaz:

```
<message password="zapis">
  <discovery protocolVersion="1"  msgid="1"  />
</message>
```

Zpráva má vyvolat odpověď obecného popisu a seznamu všech zařízení, které brána spravuje. Schéma jednotlivých zařízení je natolik obsáhlé, že jej není možné do této práce vložit.

Odpověď (výstup zkrácen kvůli lepší čitelnosti):

```
<message>
  <publication msgid="1">
    <info> <xpath>1.0</xpath> </info>
    <dataModel>
      <xsd:schema attributeFromDefault="unqualified" ...>
  ...
      <xsd:element name="devices">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element id="0" name="device">
              <xsd:complexType>
                <xsd:sequence>
                  <xsd:element name="name">SNMP Gate</xsd:element>
                  <xsd:element name="description">
                  Popis brany
                  </xsd:element>
                  <xsd:element name="data"/>
                  <xsd:element name="notifications"/>
                </xsd:sequence>
              </xsd:complexType>
            </xsd:element>
            <xsd:element id="1" name="device">
              <xsd:complexType>
                <xsd:sequence>
                  <xsd:element name="name">notebook</xsd:element>
                  <xsd:element name="description">
                  Domaci notebook
                  </xsd:element>
                  <xsd:element name="data"/>
                  <xsd:element name="notifications"/>
                </xsd:sequence>
              </xsd:complexType>
            </xsd:element>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
    ...
    </xsd:schema>
  </datamodel>
  </publication>
</message>
```

Test zprávy Get

Dotážeme se na popis systému jednotlivých zařízení.

```
<message password="zapis">
  <get  msgid="1"  objectId="0"  >
    <xpath>/iso/org/dod/internet/mgmt/mib-2/system/sysDescr</xpath>
  </get>

  <get  msgid="2"  objectId="1"  >
    <xpath>/iso/org/dod/internet/mgmt/mib-2/system/sysDescr</xpath>
  </get>
</message>
```

Odpověď:

```
<message>
  <response msgid="1">
    <value>
      Linux zoo 2.6.26-1-amd64 #1 SMP Sat Jan 10 19:55:48 UTC 2009 x86_64
    </value>
  </response>

  <response msgid="2">
    <value>
      Linux zoo 2.6.26-hrosi-jadro #4 SMP Fri Nov 21 19:19:38 CET 2008 i686
    </value>
  </response>
</message>
```

Test zprávy Subscribe

V intervalech 5 vteřin se chceme dotazovat na počet TCP paketů, které přišly na vstup zařízení číslo 1 (domácí notebook).

```
<message password="zapis">
  <subscribe  msgid="1"  objectId="1"  frequency="5"  >
    <xpath>/iso/org/dod/internet/mgmt/mib-2/tcp/tcpInSegs</xpath>
  </subscribe>
</message>
```

Odpovědí nám jsou data, čímž brána potvrzuje přijetí a bezchybnost zápisu. Ihned manažer zapíná HTTP server, který poslouchá na definovaném portu a čeká na došlé informace (v našem případě jsme nechali přijmout 5 zpráv).

```
<!-- Prvni odpoved od serveru -->
<message>
  <distribution msgid="1" distrid ="1">
    <value>
      Counter32: 6114
    </value>
  </distribution>
</message>

------------------
Starting server for distributions
-----------------
Data received:
<message context="">
  <distribution msgid="1" distrid ="1">
    <value>
      Counter32: 6114
    </value>
  </distribution>
</message>
----------------------------
------------------------------
Data received:
<message context="">
  <distribution msgid="2" distrid ="1">
    <value>
      Counter32: 6114
    </value>
  </distribution>
</message>
----------------------------
------------------------------
Data received:
<message context="">
  <distribution msgid="3" distrid ="1">
    <value>
      Counter32: 6114
    </value>
  </distribution>
</message>
----------------------------
------------------------------
Data received:
```

```
<message context="">
  <distribution msgid="4" distrid ="1">
    <value>
      Counter32: 6114
    </value>
  </distribution>
</message>
-----------------------------
-----------------------------
Data received:
<message context="">
  <distribution msgid="5" distrid ="1">
    <value>
      Counter32: 6114
    </value>
  </distribution>
</message>
-----------------------------
```

Test zprávy Event

V této fázi je manažer nastartován pouze s HTTP serverem poslouchajícím na definovaném portu. Očekává se přijetí zprávy Event, nebo ukončení programu. V našem případě simulujeme nastalou událost pomocí aplikace `snmptrap`. Simulujeme výpadek jednoho síťového spojení (linkDown), což je jedna ze standardních událostí (generic trap).

Přijatá zpráva manažerem:

```
------------------
Starting server for notifications
-----------------------------
Data received:
<message>
  <event msgid="1"
    senderid="0"
    eventSpec="device/notifications/linkDown">
      <data>
      </data>
  </event>
</message>
-----------------------------
```

Kapitola 7

Závěr

Úspěšně se podařilo splnit zadání této práce. Po detailním prozkoumání jak navrženého, tak stávajícího komunikačního protokolu, byl implementován požadovaný systém pro jejich spojení. Byly vytvořeny postupy, které optimalizují paměťové nároky protokolové brány.

Při konstrukci jednotlivých částí programu se však vyskytly překážky, které nebyly vždy zcela uspokojivě vyřešeny. První z nich je použitý HTTP server, který byl vytvořen jako volně šiřitelný projekt a není tudíž plně optimalizován pro práci s mnoha desítkami současných dotazů. Též není optimalizována jeho práce s vlákny.

Další obtíží je knihovna pracující s protokolem SNMP. Tato vznikla opět jako otevřený projekt bez jakékoliv záruky funkčnosti. Vyskytly se proto problémy s funkcemi pro práci s MIB databázemi. Při transformační fázi tak není možné získat naprosto všechny informace ohledně jednotlivých uzlů datového stromu, např. popis. Nejsou to však nijak závažné informace, které by ohrozily hlavní funkci programu či navrženého protokolu.

Podobným problémem je i zpracování asynchronních událostí, kde knihovna ne zcela jasně definuje, jak přistoupit ke specifickým informacím.

Budoucí rozšíření programu je možné směřovat do optimalizace jednotlivých součástí programu. První je již výše zmiňovaný HTTP server, kde by se dalo implementovat vlastní řešení, nebo předělat systém na komunikaci se samostatným serverem.

Bylo by možné použít jinou knihovnu, než net-snmp, případně si komunikaci mezi SNMP agentem a bránou spravovat dle vlastního implementovaného přístupu.

Literatura

[1] Peter Macejko. XML Based Network Management. Diplomová práce, České vysoké učení technické v Praze, Fakulta elektrotechnická, Praha, 2006.

[2] Internetworking Technology Handbook - Simple Network Management Protocol. Cisco, Inc., http://www.cisco.com/en/US/docs/internetworking/technology/handbook/SNMP.html.

[3] Charles M. Kozierok. TCP/IP Simple Network Management Protocol Overview. http://www.tcpipguide.com/free/t_TCPIPSimpleNetworkManagementProtocolSNMPProtocol.htm.

[4] Keith McCloghrie Marshall T. Rose. *Structure and Identification of Management Information for TCP/IP-based Internets.* RFC1155, IETF, 1990.

[5] J. Case, R. Mundy, D. Partain, and Bob Stewart. *Introduction to Version 3 of the Internet-standard Network Management Framework.* RFC2570, IETF, 1999.

[6] Apache Xerces. Knihovna pro manipulaci s XML dokumenty v jazyce C++. The Apache Software Foundation, http://xerces.apache.org/.

[7] Apache Xalan. Knihovna pro vyhledávání v XML dokumentech v jazyce C++. The Apache Software Foundation, http://xalan.apache.org/.

[8] Net-SNMP. Knihovna pro práci s protokolem SNMP v jazyce C++. http://www.net-snmp.org/.

[9] Christian Grothoff. GNU libmicrohttpd. C++ HTTP server. http://www.gnu.org/software/libmicrohttpd/.

[10] cURL. C++ knihovna pro práci s HTTP dotazy. http://curl.haxx.se/.

[11] Stewart Heitmann. ANSI C command line parser. http://argtable.sourceforge.net/.

Dodatek A

Struktura konfiguračního souboru

Konfigurační soubor je samostatný XML dokument, který se skládá z několika definovaných částí. Hlavní element je `<devices>...</devices>`. Tento vymezuje všechna spravovaná a monitorovaná zařízení.

Existují dva typy zařízení - protokolová brána a jakékoliv jiné.

Běžné zařízení

Každé zařízení musí být specifikováno dle následujícího příkladu:

```
<device id="XY">
  <name>Název zařízení</name>
  <description>Popis pro lepší pochopení.</description>
  <snmpAddr>ip_adresa</snmpAddr>
  <protocolVersion>2</protocolVersion>
  <mibs>
    <mib>RFC1213-MIB.txt</mib>
    ...
  </mibs>
  <traps>
    <manager>
      <address>ip_adresa</address>
      <port>7878</port>
    </manager>
    ...
  </traps>
  <access>
    <read>public</read>
    <write>private</write>
  </access>
</device>
```

Nejprve je nutné definovat identifikační číslo zařízení. Id 0 je rezervováno pro protokolovou bránu a systém nepovolí zařazení jiného zařízení místo ní. Tímto identifikačním číslem jsou pak opatřeny jednotlivé zprávy.

Následuje jméno a popis zařízení. To vše pro lepší uživatelský přístup administrátora při správě brány. Je nutné definovat adresu snmp agenta (url či ip adresa) a verzi SNMP protokolu (je podporována verze 1 a 2).

Dalším prvkem jsou vyjmenované jednotlivé MIB databáze, které specifikují škálu nabízených informací. Tyto soubory s definicí dat musejí být přítomny na stroji brány. Budou zpracovávány v transformační fázi. Pakliže není specifikována ani jedna, je nahrána základní sada MIB databází dle knihovny net-snmp.

Element `<traps>...</traps>` definuje množinu manažerů, kteří budou obesláni, pakliže nastane nějaká asynchronní událost. Každý manažer je definován adresou a portem, na kterém poslouchá.

Posledním základním elementem je definice přístupu. Zde je možné definovat SNMP hesla (community řetězce) pro čtení a zápis. Tyto nemají nic společného s definovavými hesly XML protokolu (viz níže). Danému manažerskému požadavku se přiřadí takový řetězec, jaká práva mu příslušejí dle XML hesla.

Protkolová brána

Samotná brána má, oproti běžným zařízením, ještě několik speciálních XML elementů, které definují podstatné součásti chování programu. Úplný soupis elementů následuje:

```
<logFile>/tmp/snmpxmld.log</logFile>
<snmp>
  <mibPath>/usr/share/snmp/mibs/</mibPath>
  <listenPort>3111</listenPort>
</snmp>
<xml>
  <xsdPath>/tmp/</xsdPath>
  <listenPort>8888</listenPort>
  <transmitPort>2555</transmitPort>
  <access>
    <read>cteni</read>
    <write>zapis</write>
  </access>
  <protocolVersion>1</protocolVersion>
</xml>
<security>
  <key>key.pem</key>
  <certificate>cert.pem</certificate>
</security>
```

Je možné specifikovat logovací soubor, nebo používat již předdefinovanou cestu.

Následující elementy specifikují části SNMP a XML modulů. V rámci SNMP lze nastavit cestu k MIB souborům a port, na kterém jsou poslouchány asynchronní události.

XML modul je možné definovat o poznání více. Nejprve cesta, kam se budou ukládat Xsd popisné soubory.

`listenPort` specifikuje, na kterém portu bude puštěn HTTP server. Zde se očekávají požadavky od manažerů.

`transmitPort` definuje port, na který se odesílají zprávy Distribution.

Element přístupu specifikuje, stejně jako v SNMP části běžného zařízení, jaká jsou hesla pro čtení a zápis.

Posledním elementem je definice klíče a certifikátu, které HTTP server používá při spojení s manažery. Tento element je volitelný.

Příklad souboru

Následuje příklad funkčního konfiguračního souboru o dvou monitorovaných zařízeních.

```
<?xml version="1.0" encoding="UTF-8"?>
<devices>
  <!-- Brana -->
  <device id="0">
    <name>SNMP Gate</name>
    <description>Popis brany</description>
    <snmpAddr>localhost</snmpAddr>
    <protocolVersion>2</protocolVersion>
    <mibs>
      <mib>RFC1213-MIB.txt</mib>
    </mibs>
    <traps>
      <manager>
        <address>127.0.0.1</address>
        <port>7878</port>
      </manager>
    </traps>
    <access>
      <read>public</read>
      <write>private</write>
    </access>

    <!--
    Specificke elementy brany
    -->
```

```
    <logFile>/tmp/snmpxmld.log</logFile>
    <snmp>
      <mibPath>/usr/share/snmp/mibs/</mibPath>
      <listenPort>3111</listenPort>
    </snmp>
    <xml>
      <xsdPath>/tmp/</xsdPath>
      <listenPort>8888</listenPort>
      <transmitPort>2555</transmitPort>
      <access>
        <read>cteni</read>
        <write>zapis</write>
      </access>
      <protocolVersion>1</protocolVersion>
    </xml>
  </device>

  <!-- Monitorovany notebook -->
  <device id="1">
    <name>notebook</name>
    <description>Domaci notebook</description>
    <snmpAddr>192.168.1.100</snmpAddr>
    <protocolVersion>2</protocolVersion>
    <mibs>
      <mib>RFC1213-MIB.txt</mib>
    </mibs>
    <traps>
      <manager>
        <address>192.168.1.50</address>
        <port>1048</port>
      </manager>
    </traps>
    <access>
      <read>home</read>
      <write>house</write>
    </access>
  </device>
</devices>
```

Dodatek B

Uživatelská příručka protokolové brány

Samotná protokolová brána funguje jako démon. Je ovládána pomocí spouštěcího skriptu, který je k programu přiložen. Vzhledem k tomu, že za cílový operační systém je zvolen Linux, budeme se v dalším popisu adresářů opírat o jeho strukturu souborů.

Instalace

Pro úspěšnou instalaci je nejprve nutné program zkompilovat. Pro přeložení je nutné mít v systému nainstalovány následující knihovny (pro každou knihovnu platí, že je nutné mít hlavičkové soubory, nikoliv jenom runtime verzi).

- **Xerces-C++** - verze 2.8.0 a vyšší
- **Xalan-C++** - verze 1.10 a vyšší
- **libsnmp** - verze 5.4.1
- **libmicrohttpd** - 0.4.0
- **libcurl** - 7.18.2

Zkompilovaný program zkopírujeme do libovolného adresáře (nejlépe však `/usr/bin` či `/usr/share/bin`). Spouštěcí skript umístíme do adresáře `/etc/init.d/`. Je nutné jej však upravit a nastavit přesné umístění binárního souboru, který bude spouštěn.

Spuštení a běh

Samotné spuštění a běh není nikterak náročné. Jediným parametrem, který program přijímá je umístění konfiguračního souboru. Cesta k souboru je uvedena v rámci spouštěcího skriptu. Proto jakékoliv změny je nutné promítnout i tam, aby program nahrával aktuální konfiguraci.

Protokolovou bránu lze spouštět při startu programu, či později manuálně. Vše je ponecháno na administrátorovi systému.

MIB soubory

Při specifikaci MIB databázových souborů v konfiguračním souboru je nutné dbát na to, aby všechny uvedené existovaly v umístění, které je součástí konfigurace zařízení brány. Jestliže některé soubory jsou nedostupné, systém nebude moci dané zařízení spravovat.

Dodatek C

Uživatelská příručka manažerské aplikace

Manažerská aplikace byla vytvořena jako dokumentační a testovací nástroj, kterým je možné ověřit rozsah funkcí protokolové brány. Aplikace je konstruována pro práci v rámci příkazové řádky. Není dostupné žádné uživatelsky příjemné grafické rozhraní.

Instalace

Pro úspěšnou kompilaci programu je nutné mít v systému nainstalovány následující knihovny ve vývojové verzi.

- **libmicrohttpd** - 0.4.0
- **libcurl** - 7.18.2
- **libargtable2** - 9

Binární zkompilovaná forma je postačující k veškeré práci. Není nutné provádět žádné další operace.

Spouštěcí parametry

Program je možné ovládat sadou vstupních parametrů. Seznam následuje:

- **-n** - určuje, že instance programu bude poslouchat příchozí asynchronní události.
- **–lport <port>** - port, na kterém je spuštěn http server pro poslouchání.
- **-v <protocol>** - verze XML komunikačního protokolu (v současnosti verze 1)
- **-a <agent ip>** - url či IP adresa brány
- **-p <port>** - port, na kterém brána poslouchá požadavky

- –**password** <**heslo**> - heslo pro danou sadu požadavků
- **-m** <**soubor s daty**> - soubor, ve kterém jsou obsaženy požadavky
- –**no-listen** - nezapne se http server

Aplikace pracuje v několika režimech - běžný dotaz/odpověď; poslouchání asynchronních událostí; poslouchání periodicky zasílaných dat.

Běžný mód se vyznačuje tím, že v rámci souboru s požadavky uvedeme několik dotazů, které chceme provést. Dále vstupními parametry specifikujeme agenta, port a heslo a odešleme žádost. Agent odpoví a systém vypíše přijatá data.

Poslouchání asynchronních událostí je specifikováno prvním a druhým přepínačem. Systém neočekává žádný vstupní soubor s daty, ale pouze spustí HTTP server a poslouchá jakákoliv příchozí data.

Distribution zprávy je možné přijímat poté, co jsme v požadavcích specifikovali nějakou zprávu Subscribe. Nyní aplikace spustí HTTP server a očekává, že jí bude agent/brána zasílat v daných frekvencích data. Zároveň systém čeká na vstup od uživatele, který zadá identifikační číslo daného zápisu, aby mohl ukončit odebírání těchto informací a zároveň zaslat ukončující zprávu na bránu.

Zde je ještě možnost upravovat daný zápis k zasílání dat. Použijeme-li přepínač `--no-listen`, systém pouze provede požadavky v souboru, ale i když je tam přítomna zpráva subscribe, nespustí HTTP server. Tato funkce demonstruje možnou změnu jednotlivých parametrů zápisu na protokolové bráně.

Struktura příkazového souboru

Požadavky v datovém souboru definují obsah jedné XML zprávy, kterou zasílá manažer agentovi. Každá řádka specifikuje jeden příkaz. Je možné specifikovat pouze čtyři požadavky - Discovery, Get, Set, Subscribe. Každý příkaz má svoji přesně definovanou strukturu, která je odvozena od navrženého protokolu. Jednotlivé elementy v rámci požadavku jsou odděleny středníkem.

Následný výčet specifikuje jednotlivé možnosti požadavků.

Discovery

```
discovery;
discovery;id_zarizeni;
```

Prvním příkazem se dotážeme na seznam všech zařízení, které brána spravuje. V druhém případě se ptáme na popisné schéma specifického zařízení.

Get

```
get;id_zarizeni;xpath_vyraz;
```

V rámci zprávy Get je možné se dotázat pouze jedním XPath výrazem na uzel či skupinu uzlů v datovém stromu.

Set

```
set;id_zarizeni;xpath_vyraz;hodnota;
```

Zprávou Set je možno nastavit hodnotu jednoho konkrétního uzlu.

Subscribe

```
subscribe;id_zarizeni;frekvence;distrid;delete;xpath;...
```

Pro zpracování zprávy Subscribe je nutné definovat, pro které zařízení je tato zpráva určena. Dále specifikujeme frekvenci zasílání dat. Pro změny či smazání zápisu slouží další dva parametry - distrid, delete - tyto mohou být nezadané a systém to interpretuje jako novou zprávu. Posledním argumentem je sekvence xpath výrazů, které definují data, jež chceme dostávat.

Argument `delete` je pouze booleovská hodnota. Pro smazání daného zápisu bude nabývat hodnoty 1.

Příklad sekvence požadavků

```
discovery;0;
get;1;/iso/org/dod/internet/mgmt/mib-2/system/sysDescr;
get;0;/iso/org/dod/internet/mgmt/mib-2/tcp;
subscribe;1;120;;;/iso/org/dod/internet/mgmt/mib-2/tcp;\
  /iso/org/dod/internet/mgmt/mib-2/system;
```

Dodatek D

Obsah přiloženého CD

Přiložené médium obsahuje následující adresáře a soubory:

- `/source` - zdrojové soubory programu
- `/manager` - zdrojové soubory manažerského programu
- `/initscript` - spouštěcí skript pro spuštění démona
- `/config_file` - ukázkový konfigurační soubor protokolové brány
- `/gate_usrguide.txt` - instalační a uživatelská příručka protokolové brány
- `/manager_usrguide.txt` - instalační a uživatelská příručka manažerské aplikace.
- `/text/source` - zdrojové kódy práce ve formátu LaTeX
- `/text/pdf` - text této práce ve formátu pdf.