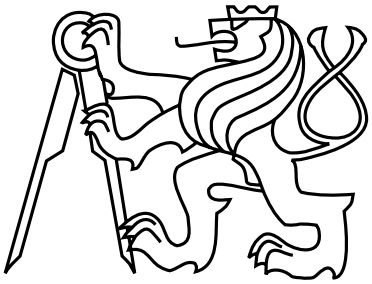
České vysoké učení technické v Praze
Fakulta elektrotechnická
Katedra počítačů



Diplomová práce

SNMP/XML brána

Bc. Tomáš Hroch

Vedoucí práce: Ing. Peter Macejko

Studijní program: Elektrotechnika a informatika, strukturovaný, Navazující magisterský

Obor: Výpočetní technika

18. února 2009

Poděkování

Zde můžete napsat své poděkování, pokud chcete a máte komu děkovat.

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 Kořenovicích nad Bečvárkou dne 15. 5. 2008 .....

Abstract

Translation of Czech abstract into English.

Abstrakt

Abstrakt práce by měl velmi stručně vystihovat její podstatu. Tedy čím se práce zabývá a co je jejím výsledkem/přínosem.
Očekávají se cca 1 – 2 odstavce, maximálně půl stránky.

Obsah

1 Úvod **1**

2 SNMP **3**
2.1 Správní struktura . . . 3
2.2 SMI, MIB standardy . . . 5
2.3 Verze SNMP protokolu . . . 6

3 XML protokol **11**
3.1 Obecný informační model . . . 11
3.1.1 Odvození typů . . . 11
3.1.2 Definice modulů . . . 12
3.1.3 Popis zařízení . . . 12
3.1.4 Oznamovací zprávy . . . 12
3.1.5 Adresace dat . . . 13
3.2 Mapování MIB do XML . . . 13
3.2.1 Importy . . . 13
3.2.2 Datové typy . . . 13
3.2.3 MIB strom . . . 13
3.3 Zprávy . . . 16

4 Návrh systému **21**
4.1 Teoretické požadavky . . . 21
4.1.1 XML . . . 22
4.1.2 SNMP . . . 27
4.2 Struktura programu . . . 27
4.2.1 SOAP vs. démon . . . 27
4.2.2 Navrhovaná aplikace . . . 28
4.2.2.1 Inicializace . . . 28
4.2.2.2 Transformace dat . . . 29
4.2.2.3 Zpracovávání požadavků . . . 29
4.3 Správa protokolové brány . . . 30
4.4 XML manažerská aplikace . . . 30

5 Implementace **31**

Literatura **33**

Seznam obrázků

2.1 Základní princip fungování SNMP spravované sítě 3
2.2 Komunikace mezi SNMP manažerem a agentem 4
2.3 Schéma datových paketů protokolu SNMPv1 a v2 ([1]) 7
2.4 Schéma datového paketu protokolu SNMPv3 ([1]) 10

3.1 Popis zařízení v XML schématu ([1]) 12
3.2 Mapování aplikačních typů SMIv1 do XML schématu ([1]) 14
3.3 Struktura XML zprávy 17

4.1 Schéma navrhovaného systému 21
4.2 Obecná struktura XML dokumentu 23
4.3 Struktura elementu device 23
4.4 Struktura elementu info 24
4.5 HTTP zprávy předávané mezi manažerem a bránou 26
4.6 HTTP komunikace mezi manažerem a agentem/bránou 26
4.7 Fáze běhu programu 28

Seznam tabulek

3.1 Mapování makra OBJECT-TYPE, jednoduchý typ (SMIv1) ([1]) 14
3.2 Mapování SEQUENCE, makro OBJECT-TYPE ([1]) 15
3.3 Mapování SEQUENCE OF, makro OBJECT-TYPE ([1]) 15
3.4 Mapování TRAP-TYPE makra ([1]) 16

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 takovéto 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, až 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 spojení jej s několika verzemi protokolu SNMP.

V kapitole 2 je podrobně popsán protokol SNMP.

Kapitola 2

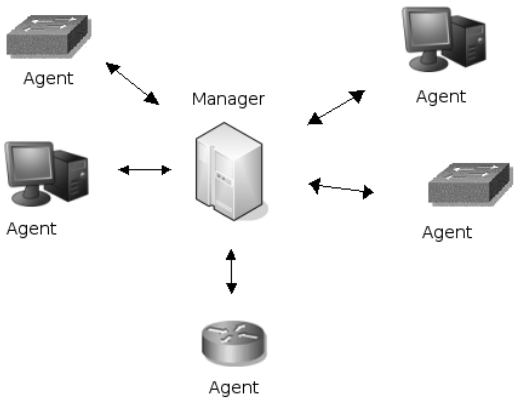
SNMP

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. Struktura spravované sítě se tak dělí na tři klíčové elementy - spravované zařízení, agenta a manažera (viz obrázek 2.1).



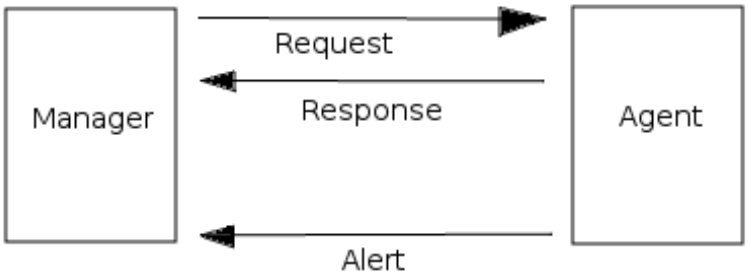
Obrázek 2.1: Základní princip fungování SNMP spravované sítě

- **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 jejích 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 jí započal. Základní schéma je vyjádřeno na obrázku 2.2.



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 zprávi Alert (v SNMP jsou to zprávy TRAP či INFORM, obě budoudiskutovány 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 ([?]) popisuje a definuje základní datové struktury a typy, které protokol využívá. Jednotlivé objekty jsou pojmenovány a organizovány, aby bylo možné 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 identifikátory jsou přiřazeny počátku 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 bylo 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:

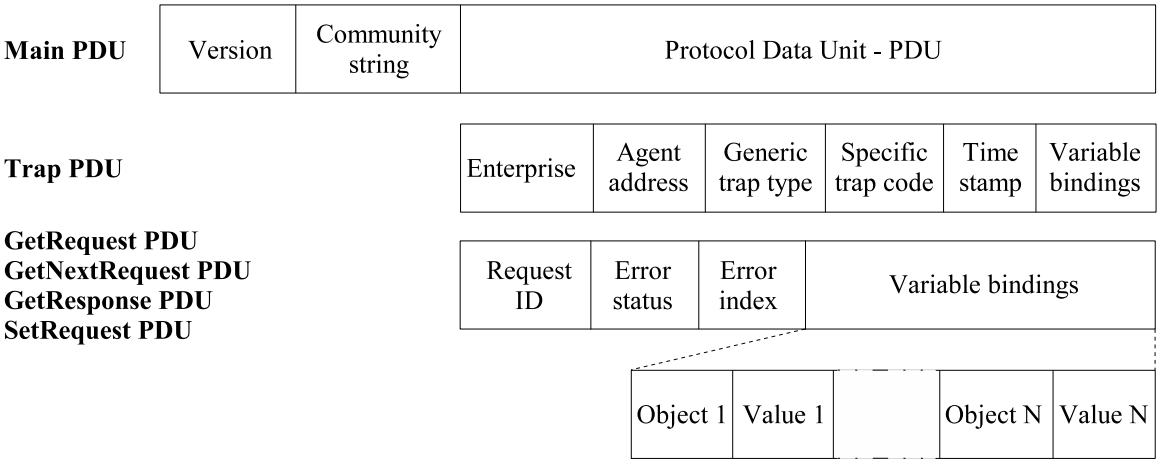
- **Network Address** - obecná síťová adresa pro podporu mnoha rodin protokolů.
- **IpAddress** - přímo definovaný typ pro IP adresu. SMIv1 podporuje pouze 32 bitovou adresu (IPv4)
- **Counter** - čítač, vyjádřen celým číslem bez znaménka; Jeho hodnota se pouze zvyšuje a to až do maxima a pak se vrací 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 nastálé 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 [?] je řečeno, že tato verze může být chápána jako SNMPv2 s dodatečnými administrativními a bezpečnostními schopnostmi.

SNMPv3 definuje tři základní služby:

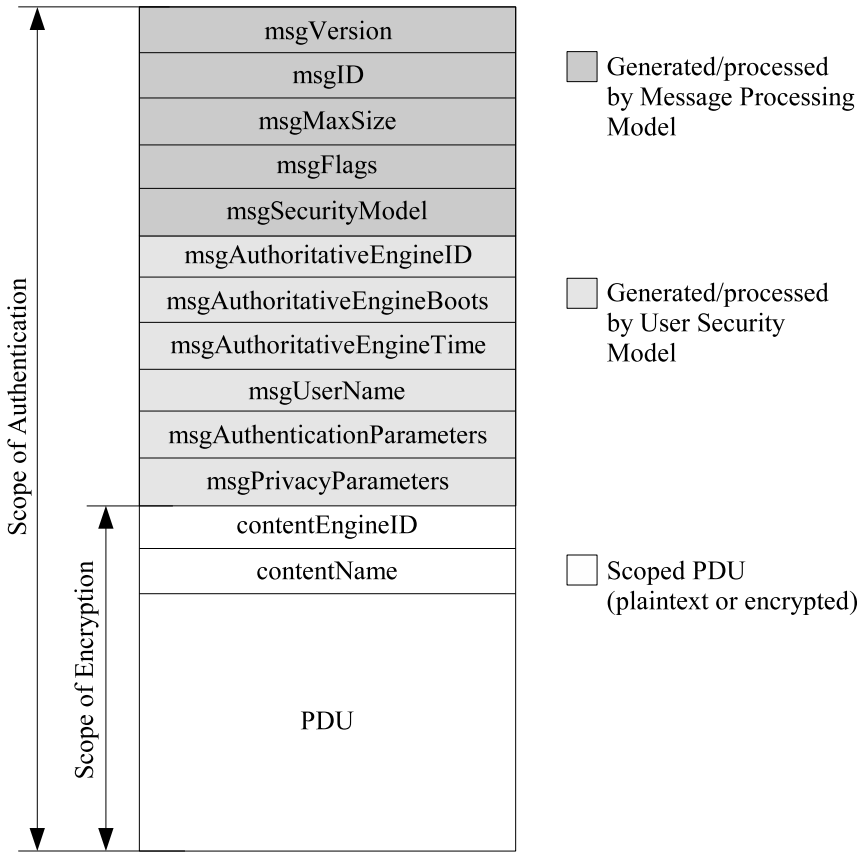
- *Autentifikaci* - datový přenos od manažera k agentovi může být autentifikován, aby se zajistilo ověření identity odesílajícího.
- *Soukromí* - šifrování přenášených zpráv.
- *Přístupová práva* - agent může definovat přístupová práva, omezovat přístup manažerům k pouze některým akcím a částem dat.

Základním principem SNMPv3 je modularita. Každá SNMP entita je tvořena SNMP řídícím systémem a vlastní aplikací. Řídící systém má za úkol přijímat, odesílat, šifrovat, 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 verzi 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 navržen 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. s dopomocí člověka.

Výsledkem analýzy problému je systém využívající kousek od obou přístupů. Na nejvyšší úrovni abstrakce je každé zařízení složeno z modulů. Každý model 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

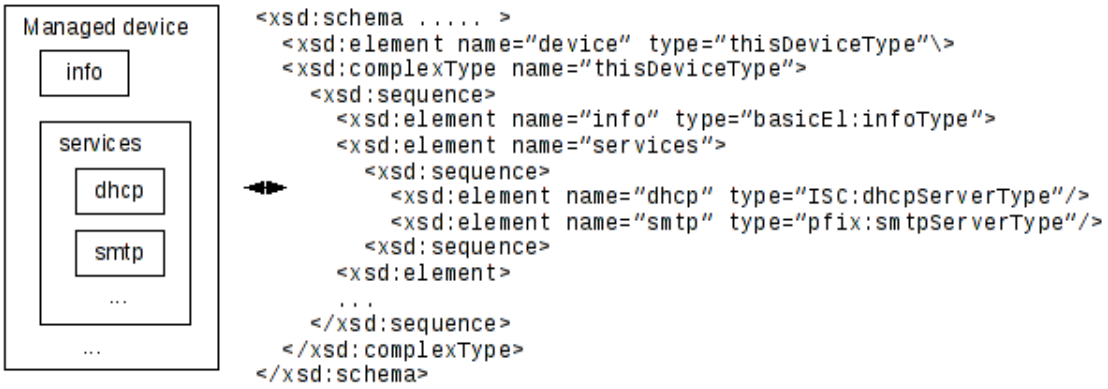
Jak již bylo řečeno, každé zařízení se skládá z modulů. Jednotlivé moduly jsou též popsány XML schématem. Každé takové schéma musí splňovat přesné požadavky na poskytované informace.

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.



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 všechny 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 vyřešit importy jednotlivých 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 příslučným 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 daném 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 splehlivý 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 probíhajícím 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 byla uzlu 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/dhcp/no
  <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

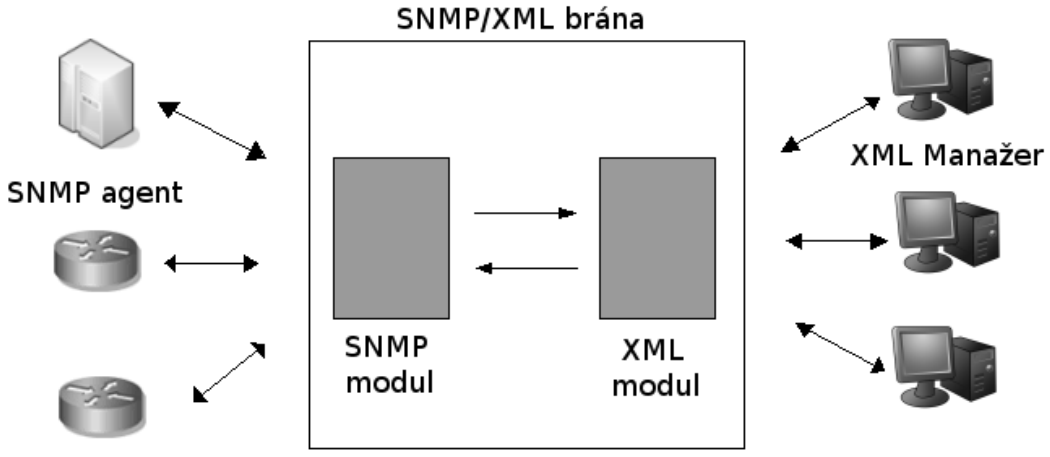
Návrh systému

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 třech 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 navrhovahého 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

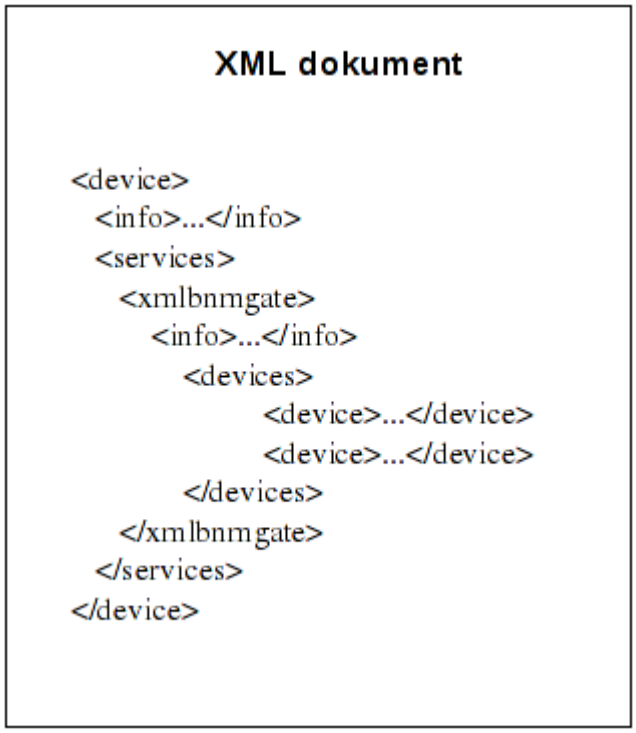
Nejprve se zaměříme na reprezentaci dat, které budou v rámci XML popisovat jak bránu, tak monitorované zařízení. Z předchozích kapitol vyplynulo, že bude použito částečně objektového přístupu a přímého mapování MIB. Strukturu dat bude popisovat XML dokument, strom, který má strukturu vyjádřenou na obrázku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

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

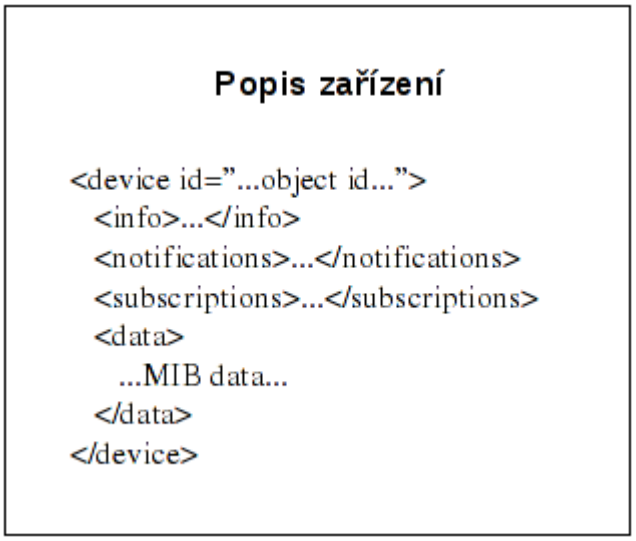
- **info** - tento element obsahuje text, kterým je popsáno dané zařízení.
- **services** - element vymezující poskytované služby (při širší implementaci může obsahovat služby DNS, DHCP, apod.)
- **xmlbnmGate** - naše služba poskytující spojení XML a SNMP protokolu
- **device** - je podelementem **xmlbnmGate**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

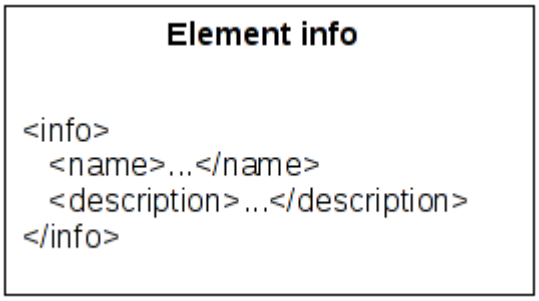


Obrázek 4.3: Struktura elementu device

- **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** - sem jsou mapována 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 mohly být data doručena a specifikovat časový interval, tj. frekvenci sledování příslušné veličiny.

Děti 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. Tudíž v rámci typu události je nutné uvést příjemce, který bude zprávy očekávat. Přesná specifikace jednotlivých uzlů dokumentu je v příloze

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

```
/device/services/xmlbnmgate/device[id=1]/data/...
```

Zprávy

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

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 pouze jednu frontu zpracování zpráv, čímž budou jednotlivé dotazy zpracovány postupně. Bude tak zaručena integrita dat a předejde se různým extrémním situacím.

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

- Get
- Sset
- 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 taky 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.

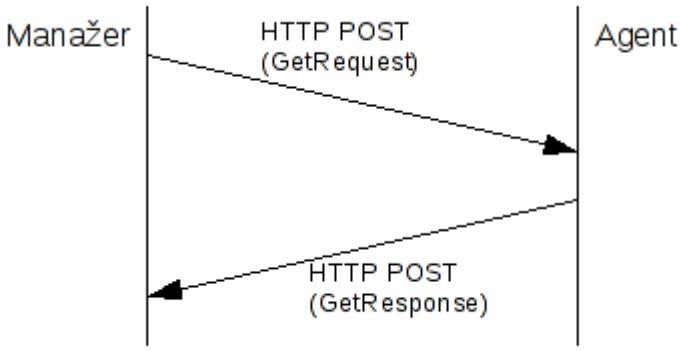
HTTP dotaz

```
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 naší 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, které je zasíláno jako součást XML zprávy a bude pouze přepsáno do SNMP paketu. Je tedy zřejmé, že ponecháváme bezpečnost takovou, jak je standardizována v SNMP protokolu.

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

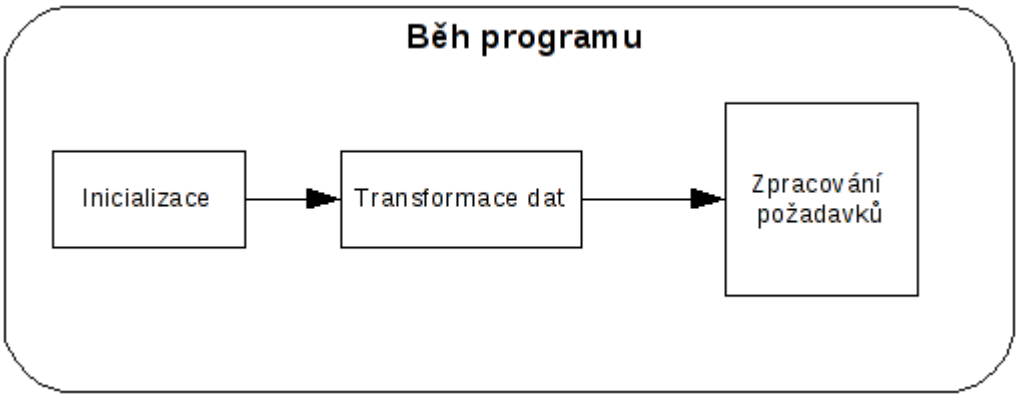
intervalech zasílý SNMP dotazy na monitorované zařízení, což je neslučitelné se základní myšlenkou webových služeb.

Asi největší nevýhodou tohoto přístupu je transformace dat z MIB do XML. Při spuštění webové služby je nutné, aby již všechna zařízení 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 přůchodu oběma přechozími fázemi se program dostává do situace, kdy vyčkává na příchozí požadavky, ať již ze strany manažera 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. Pakliže by 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 takvého 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. Tudíž 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

Literatura

[1] I. P. Macejko. *XML SNMP protocol*, volume 1. CVUT, Oval Road, London, UK, 4th edition, 2006.