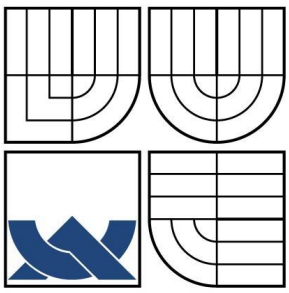
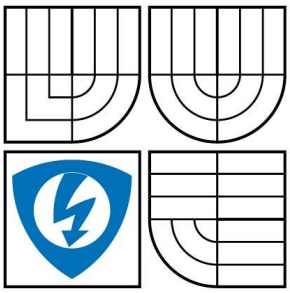


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

NÁVRH MODELOVÉ ARCHITEKTURY SYSTÉMOVÉ DATABÁZE MIB

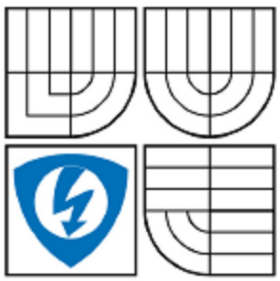
ARCHITECTURE DESIGN FOR AN EVALUATION MIB DATABASE

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — VLADIMÍR MIKULICA
AUTHOR

VEDOUCÍ PRÁCE — Ing. KAROL MOLNÁR, Ph.D
SUPERVISOR

BRNO 2008



VYSOKÉ UČENÍ
TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

Ústav telekomunikací

Bakalářská práce

bakalářský studijní obor
Teleinformatika

Student:	Mikulica Vladimír	***ID:***	78584
Ročník:	3	***Akademický rok:***	2007/2008

NÁZEV TÉMATU:

Návrh a implementace modelové architektury systémové databáze MIB

POKYNY PRO VYPRACOVÁNÍ:

Seznamte se se standardy zaměřené na popis dat v systému SNMP a na strukturu databáze MIB. Podrobně rozeberte způsob reprezentace jednotlivý datových typů. Seznamte se se způsobem správy položek v databázi MIB (čtení a zápis jednoduché hodnoty, čtení a zápis záznamů do tabulky, vkládár nových řádků do tabulky, atd.).
Na základě získaných informací navrhněte strukturu pro simulační model databáze MIB. Při návrhu zvolte vhodný kompromis mezi úrovní abstrakce daných metod a realizovatelnosti v omezeném časovém horizontu. Vypracovaný model podrobně zdokumentujte. Na základě zvoleného modelu proveďte implementaci databáze MIB zaměřené na řízení procesů spojených se zpracováním provozu modelu aktivního síťového uzlu.

DOPORUČENÁ LITERATURA:

[1] BERNET, Y., BLAKE, S., GROSSMAN, D., SMITH, A.: An Informal Management Model for Diffser Routers, RFC 3290, Internet Engineering Task Force, 2002
[2] BAKER, F., CHAN, K., SMITH, A.: Management Information Base for the Differentiated Services Architecture, RFC 3289, Internet Engineering Task Force, 2002

Termín zadání:	11.2.2008	***Termín odevzdání:***	4.6.2008

Vedoucí práce: Ing. Karol Molnár, Ph.D.

prof. Ing. Kamil Vrba, CSc.
předseda oborové rady

UPOZORNĚNÍ:

LICENČNÍ SMLOUVA
POSKYTOVANÁ K VÝKONU PRÁVA UŽÍT ŠKOLNÍ DÍLO

uzavřená mezi smluvními stranami:

1. Pan/paní

Jméno a příjmení: Vladimír Mikulica

Bytem:

Narozen/a (datum a místo): 11.8.1986

(dále jen „autor“)

a

2. Vysoké učení technické v Brně

Fakulta elektrotechniky a komunikačních technologií

se sídlem Údolní 244/53, 602 00, Brno

jejímž jménem jedná na základě písemného pověření děkanem fakulty:

prof. Ing. Kamil Vrba, CSc.

(dále jen „nabyvatel“)

Čl. 1
Specifikace školního díla

1. Předmětem této smlouvy je vysokoškolská kvalifikační práce (VŠKP):
   - □ disertační práce
   - □ diplomová práce
   - □ bakalářská práce
   - □ jiná práce, jejíž druh je specifikován jako .....

   (dále jen VŠKP nebo dílo)

Název VŠKP: Návrh a implementace modelové architektury systémové databáze MIB

Vedoucí/ školitel VŠKP: Ing. Karol Molnár, Ph.D.

Ústav: Ústav telekomunikací

Datum obhajoby VŠKP:

VŠKP odevzdal autor nabyvateli v*:

- □ tištěné formě – počet exemplářů ……..
- □ elektronické formě – počet exemplářů ……..

* hodící se zaškrtněte

2. Autor prohlašuje, že vytvořil samostatnou vlastní tvůrčí činností dílo shora popsané a specifikované. Autor dále prohlašuje, že při zpracovávání díla se sám nedostal do rozporu s autorským zákonem a předpisy souvisejícími a že je dílo dílem původním.
3. Dílo je chráněno jako dílo dle autorského zákona v platném znění.
4. Autor potvrzuje, že listinná a elektronická verze díla je identická.

Článek 2
Udělení licenčního oprávnění

1. Autor touto smlouvou poskytuje nabyvateli oprávnění (licenci) k výkonu práva uvedené dílo nevýdělečně užít, archivovat a zpřístupnit ke studijním, výukovým a výzkumným účelům včetně pořizovaní výpisů, opisů a rozmnoženin.
2. Licence je poskytována celosvětově, pro celou dobu trvání autorských a majetkových práv k dílu.
3. Autor souhlasí se zveřejněním díla v databázi přístupné v mezinárodní síti
   - □ ihned po uzavření této smlouvy
   - □ 1 rok po uzavření této smlouvy
   - □ 3 roky po uzavření této smlouvy
   - □ 5 let po uzavření této smlouvy
   - □ 10 let po uzavření této smlouvy

   (z důvodu utajení v něm obsažených informací)
4. Nevýdělečné zveřejňování díla nabyvatelem v souladu s ustanovením § 47b zákona č. 111/ 1998 Sb., v platném znění, nevyžaduje licenci a nabyvatel je k němu povinen a oprávněn ze zákona.

Článek 3
Závěrečná ustanovení

1. Smlouva je sepsána ve třech vyhotoveních s platností originálu, přičemž po jednom vyhotovení obdrží autor a nabyvatel, další vyhotovení je vloženo do VŠKP.
2. Vztahy mezi smluvními stranami vzniklé a neupravené touto smlouvou se řídí autorským zákonem, občanským zákoníkem, vysokoškolským zákonem, zákonem o archivnictví, v platném znění a popř. dalšími právními předpisy.
3. Licenční smlouva byla uzavřena na základě svobodné a pravé vůle smluvních stran, s plným porozuměním jejímu textu i důsledkům, nikoliv v tísni a za nápadně nevýhodných podmínek.
4. Licenční smlouva nabývá platnosti a účinnosti dnem jejího podpisu oběma smluvními stranami.

V Brně dne: …….

…….. ……

Nabyvatel Autor

ANOTACE

Předložená práce se zabývá problematikou návrhu a implementace modelové databáze MIB. Součástí práce je úvod do problematiky protokolu SNMP (Simple Network Management Protocol) a databáze MIB (Management Information Base), kde je podrobně popsána větev MIB II. Právě tato větev je současným standardem pro správu objektů pomocí protokolu SNMP. Následně jsou zde vysvětleny jednotlivé datové typy, které databáze MIB užívá. Dále jsou v práci popsány postupy pro návrh databáze MIB v jazyce ASN. 1 dle RFC 2233 a následný převod této databáze MIB, pomocí dostupných kompilátorů do jazyka C. V dalších kapitolách práce jsou podrobně zdokumentovány jednotlivé části modelu, které byly vytvořeny v simulačním prostředí OPNET Modeler během realizace komplexního simulačního scénáře. Uvedené části realizují komunikaci mezi manažerem a agentem. Příslušný stavový automat a jednotlivé bloky modelu jsou popsány tak, aby jej bylo možné opět později sestavit dalšími zájemci.

.

Klíčová slova: SNMP, agent, manažer, MIB, Management Information Base, UDP, OPNET Modeler, ASN. 1.

ABSTRACT

The submetted work deals with the systems design and the implementation of the model database MIB. The work is the introduction into the protocol SNMP (Simple Network Management Protocol) and the database MIB (Management Information Base), which describes in detail branch MIB II. This branch is the current standard for the management of objects using SNMP. Consequently, there are explained different data types that the MIB database uses. Furthermore, the work describes the procedures for the design of the database MIB in the language of ASN. 1 according to RFC 2233 and the subsequent transfer of the MIB database, available through the compilers into the langure C. In the following chapters of the work, each of the models that were created in the simulated environment OPNET Modeler during the implementation of a comprehensive simulation scenario are documented more in detail. The above mentioned parts implement the communication between the manager and the agent. The state machine and the individual blocks of the model are described so that it could be later re-assembled by other candidates.

Keywords: SNMP, agent, manager, MIB, Management Information Base, UDP, OPNET Modeler, ASN. 1.

MIKULICA, V. *Návrh a implementace modelové architektury systémové databáze MIB*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2008. 52 s. Vedoucí bakalářské práce Ing. Karol Molnár, Ph.D.

Prohlášení

Prohlašuji, že svou bakalářskou práci na téma „Návrh modelové architektury systémové databáze MIB“ jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedeného semestrálního projektu dále prohlašuji, že v souvislosti s vytvořením tohoto projektu jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.

V Brně dne ....

podpis autora

Poděkování

Děkuji vedoucímu bakalářské práce Ing. Karolu Molnárovi, Ph.D za jeho vstřícný přístup, motivaci, cenné rady a velkou ochotu konzultovat všechny mé poznatky. Dále bych rád poděkoval své rodině, která mne po celou dobu podporovala a poskytla dostatek času k vypracování této práce. V neposlední řadě děkuji své přítelkyni za její trpělivost a motivaci.

V Brně dne ....

podpis autora

Abecední seznam použitých zkratek:

ARP (Address Resolution Protocol); Internetový protokol, který dynamicky mapuje internetové síťové IP adresy do fyzických (hardwarových) MAC adres.

ASCII (American Standard Code for Information Interchange); Osmibitový kód stanardizovaný organizací ANSI pro zajištění kompatibility při datových přenosech.

ASN.1 (Abstract Syntax Notation One); Notace pro abstraktní zápis dat, která není závislá na typu zařízení.

AT (Address Translation Group); Skupina překladu adres nacházející se v podstromu MIB-II.

BER (Basic Encoding Rules); Pravidla pro kódování datových jednotek podle ISO standardu ASN.1.

CCITT (Comité Consultatif International Télégraphique et Téléphonique); Mezinárodní poradní výbor pro telegrafii a telefonii v rámci UIT (Union Internationale des Télécommunications) - Mezinárodní telekomunikační unie.

CER (Canoncial Encoding Rules); Tato pravidla pro kódování jsou podmnožinou kódování BER.

DER (Distinguished Encoding Rules); Tato pravidla pro kódování jsou podmnožinou kódování BER.

EGP **(**Exterior Gateway Protocol**)**; Směrovací protokol, používaný převážně v páteřním systému Internetu.

FB (Function Block); Funkční blok.

FDDI (Fiber Distributed Data Interface); Síťová technologie s přenosovou rychlostí 100 Mb/s po optických vláknech.

HB (Header Block); Hlavičkový blok

IANA **(**Internet Assigned Numbers Authority); Organizace zodpovědná za systém doménových jmen Internetu a za používání a přidělování různých technických parametrů adres a číselných identifikátorů.

ICI (Interface Control Information); Datová struktura užívána pro přenos informací mezi procesy v OM.

ICMP (Internet Control Message Protocol); Protokol z rodiny síťových protokolů IP sloužící k přenosu chybových a řídících správ na vrstvě IP.

IP (Internet Protokol); Protokol pracující na 3 vrstvě modelu ISO/OSI, který zajišťuje přenos paketů bez navazování spojení.

ISO (International Organization of Standardization); Mezinárodní organizace pro normalizaci

ITU-T (International Telecommunication Union - Telecommunication); Mezinárodní telekomunikační unie.

MAC MAC address – Hardwarová adresa nutná pro každé zařízení nebo port připojené do sítě.

MIB (Management Information Base); Databáze specifikující vlastnosti spravovaného zařízení pomocí protokolu SNMP.

OBJID (Object Identifier); Identifikátor atributu modelu v OM.

OID (Object Identifier); Číselný identifikátor představuje administrativně přiřazená jména stromové struktury databáze MIB.

OM OPNET Modeler

OSPF (Open Shortest Path First); Protokol určený k výměně směrovacích informací v rozsáhlých strukturách propojených sítí.

PER (Packed Encoding Rules); Kódovací pravidla pro mohutné datové toky. Jedná se opět o podmnožinu kódování BER.

RFC (Request for Comment); Standardy a dokumenty např. pro popis internetových protokolů .

RIP (Routing Information Protocol); Vektorově orientovaný směrovací protokol, optimalizující cestu od zdroje k cíli podle vzdálenosti mezi zdrojem a cílem.

SNMP (Simple Network Management Protocol); Protokol pro vzdálenou správu různých zařízení.

SMI (Structure of Management Information); Popis objektů a jejich chování v databázi MIB.

SV (State Variables); Stavové proměnné.

TB (Terminal Block); Ukončovací blok.

TCP (Transmission Control Protocol); Protokol poskytující spolehlivé, plně duplexní přenosy s navazováním spojení.

TV (Temporary Variables); Slouží k deklaraci dočasných proměnných.

UDP (User Datagram Protocol); Rychlý ale nespolehlivý transportní protokol bez navazování spojení ze sady protokolů TCP/IP.

XER (XML Encoding Rules); Kódovací pravidla pro převod ASN.1 do XML.

XML (eXtensible Markup Language); Standard pro definici metod reprezentace textu v elektronické formě.

Obsah

Seznam obrázků

Úvod

V dnešní době prakticky každé zařízení pro svou vzdálenou správu používá protokol SNMP (Simple Network Management Protocol), který byl původně určen pro usnadnění správy sítě. Využití tohoto protokolu ale stále nabývá na významu i v oblastech automatizace a měřící techniky. Na stav jednotlivých objektů se dotazujeme pomocí protokolu SNMP (SNMP manažer se dotazuje na SNMP agenty). Protokol SNMP úzce spolupracuje s databází MIB (Management Information Base), která je implementována v každém zařízení spravovaném tímto protokolem. Objekty v této databázi jsou uspořádány do stromové struktury.

Tato práce je zaměřena na seznámení se s protokolem SNMP a strukturou databáze MIB. V dalším kroku bude, na základě získaných informací, proveden návrh databáze MIB a implementace této databáze v simulačním prostředí programu OPNET Modeler. Vše je podrobně zdokumentováno v následujících kapitolách.

1 Protokol SNMP

S rostoucím počtem různých zařízení s různými systémovými platformami se vytváří stále větší sítě. S velikostí se zvětšuje i nepřehlednost této sítě a porucha nebo špatné nastavení některého ze síťových prvků může způsobit kolaps části nebo i celé sítě. Z tohoto důvodu nabývala na významu možnost vzdálené správy a nepřetržitého dohledu nad jednotlivými síťovými komponentami. Proto byl v roce 1990, organizací Internet Engineering Task Force, uveden protokol SNMP (Simple Network Management Protokol) [1] pro vzdálenou správu aktivních síťových prvků v sítích IP.

Univerzálnost protokolu SNMP umožňuje jeho široké využití. Oblíbeným se stal díky své jednoduchosti pro konfiguraci, implementaci do zařízení a nenáročnosti na výkon. V současné době existují 3 verze protokolu SNMP (verze 2 je nejpoužívanější). SNMPv1 a SNMPv2 používají pro autentizaci textový řetězec (community string), což není dostačující, protože tento textový řetězec se dá jednoduše „zachytit“ protokolovým analyzátorem. Proto již v protokolu SNMPv3 je možné využít autentizaci pomocí jména, hesla a šifrování.

1.1 Komponenty pro vzdálenou správu pomocí protokolu SNMP

Systém vzdálené správy pomocí protokolu SNMP slouží k centralizovanému dohledu a správě rozsáhlých sítí. K tomuto účelu systém používá dva typy komponent: stanice určené pro správu síťových prvků, tzv. manažery a spravované komponenty, tzv. agenty. Manažer je obvykle pracovní stanice, která je vybavena uživatelským programem určeným pro správu a dohled nad sítí. Tento program je nejčastěji ovládán lidskou obsluhou. Agent je taktéž specializovaným softwarem, který běžící na konkrétním síťovém prvku.

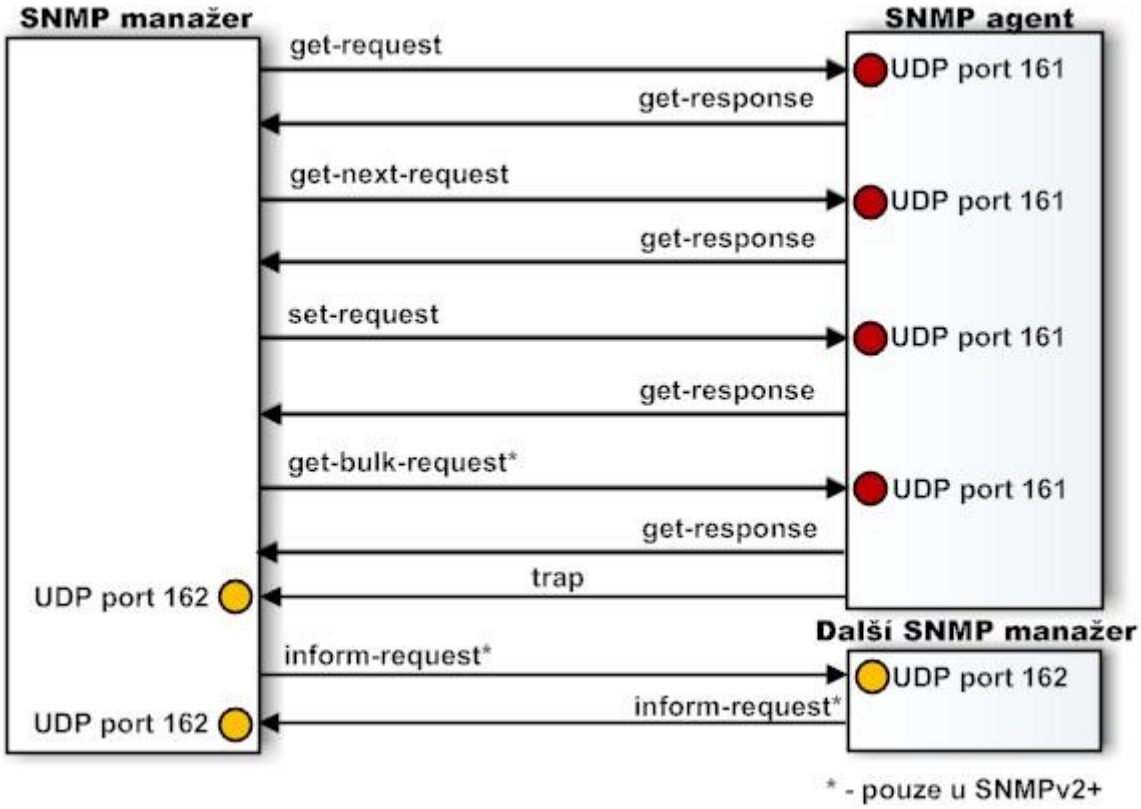
Agenti shromažďují informace o síťových událostech a na žádost SNMP manažera mu je předávají. SNMP manažer průběžně tyto informace vyhodnocuje a udržuje si přehled o celkovém stavu provozu na síti. Takto může být včas odhalen blížící se kritický stav sítě. Výsledkem činnosti SNMP manažera jsou systémové záznamy, které zaznamenávají stav sítě v určených časových periodách a varovná hlášení (tzv. trap zprávy), která informují správce sítě o výjimečných událostech na sledovaných zařízeních. Výjimečnou událostí může být například výpadek komunikace linky, porucha některého z prvků sítě nebo neoprávněný přístup.

V systému pracujícím na bázi protokolu SNMP jsou parametry (statické i dynamické) síťového prvku a zpracovávaného síťového provozu modelovány pomocí řízených objektů. Popis a chování těchto objektů definuje standard SMI (Structure of Management Information).

Všechny stavové a statistické informace, které manažer může od agenta získat, jsou ukládány do databáze řídících informací (Management Information Base – MIB). Databázi MIB je věnována kapitola 2.

1.2 Komunikace manažera a agenta

Protokol SNMP je určen pro využití v sítích založených na sadě protokolů TCP/IP. Pro výměnu zpráv využívá rychlý, ale nespolehlivý transportní protokol UDP. Komunikace na úrovni protokolu UDP, pro standardní výměnu zpráv typu žádost-odpověď, probíhá na portech 161 a pro varovné zprávy trap na portech 162. Princip komunikace mezi manažerem a agentem je znázorněn na Obr**.** 1.



Obr. 1 Princip komunikace manažer-agent

U SNMPv1 se lze setkat s pěti různými druhy operací protokolu mezi manažerem a agentem. Následující verze protokolu SNMP (2 a 3) přidaly ještě další čtyři typy. Operace protokolu SNMP jsou následující:

- get,
- get-next,
- get-bulk (pouze od SNMP verze 2 a SNMP verze 3),
- set,
- get-response,
- trap,
- notification (pouze od SNMP verze 2 a SNMP verze 3),
- inform (pouze od SNMP verze 2 a SNMP verze 3),
- report (pouze od SNMP verze 2 a SNMP verze 3).

Tyto operace jsou stručně popsány v následujících kapitolách.

1.2.1 Operace get

Operace get je základní operací, která slouží k získání hodnoty určitého objektu. Operace je vyvolána manažerem, který posílá žádost get agentovi. Dotazovaný objekt je identifikován v žádosti pomocí OID (viz kapitola 2.3.5).

1.2.2 Operace get-next

Operace get-next přečte hodnotu OID následujícího za vybraným OID a zároveň do položky OID zadá následující identifikátor, takto může například projít celou tabulku. Stejně jako u operace get je iniciátorem operace manažer, který posílá žádost get-next agentovi.

1.2.3 Operace get-bulk

Tato operace se objevila až ve druhé verzi protokolu SNMP a umožňuje získání větší části MIB pomocí jedné žádosti, čímž se mnohdy urychluje komunikace.

1.2.4 Operace set

Operace set umožňuje změnit hodnoty skalárních objektů nebo přidat řádek do tabulky. Žádost od manažera je zaslána agentovi zprávou set-request, na kterou pak agent odpoví zprávou get-response.

1.2.5 Operace get-response

Operace get-response je odpovědí agenta manažerovi. Agent vykoná tuto operaci jako reakci na předchozí příkazy. Odpověď obsahuje i dotaz, protože protokol nezajišťuje přímou souvislost mezi dotazem a odpovědí.

1.2.6 Varovná zpráva trap

Tento příkaz posílá agent manažerovi jako oznámení nějaké změny, která nastala v síti (např. výpadek linky, porucha zařízení apod.). Na tuto varovnou zprávu agent nečeká odpověď.

1.2.7 Zpráva notification

V první verzi protokolu SNMP byla struktura varovných zpráv (trap) a zprav get a set odlišná, Z toho důvodu v rámci snahy o zavedení jednotné struktury byla v dalších verzích definována zpráva SNMP notification.

1.2.8 Zpráva inform

Tato zpráva umožňuje komunikaci dvou manažerů mezi sebou s potvrzováním.

1.2.9 Zpráva report

Tato zpráva byla definována ve druhé verzi protokolu SNMP, ale nikdy nebyla implementována. Ve třetí verzi protokolu SNMP byla však ponechána pro komunikaci mezi jednotlivými komponentami tohoto protokolu.

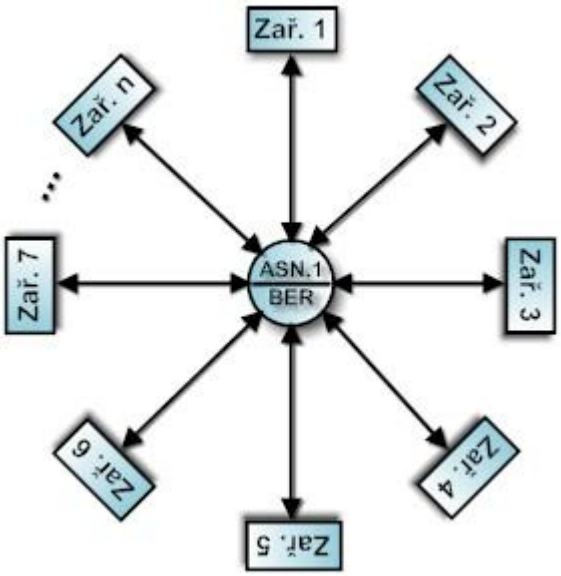
1.3 Jazyk ASN.1

Protokol SNMP je určen pro výměnu řídicích informací, které jsou uspořádány do standardizované struktury. Taková struktura je nezbytná k tomu, aby data byla interpretována na obou koncích komunikace stejně. Pro zajištění srozumitelnosti dat je třeba definovat jak tzv. abstraktní syntaxi, tak i syntaxi pro přenos. Abstraktní syntaxe specifikuje způsob zápisu dat. Syntaxe pro přenos pak definuje pro prvky, popsané pomocí abstraktní syntaxe, proces kódování do formy, která je vhodná pro přenos přes datovou síť.

ASN.1 (Abstract Syntax Notation One) je notace pro abstraktní zápis dat. ASN.1 lze též zjednodušeně považovat za „programovací jazyk“, který určuje jakým způsobem jsou data reprezentována a přenášena mezi agentem a manažerem.

V ASN.1 se objekty popisují tak, aby byl zápis jednoznačně srozumitelný a vyložitelný člověkem. Při komunikaci dvou zařízení (např. počítačů) není nutné, aby byl popis srozumitelný pro člověka, naopak je nutné zajistit takový tvar, aby byl jednoznačný a srozumitelný pro všechny platformy počítačů. Informace se převádí do kódováním BER (Basic Encoding Rules), které udává konkrétní fyzickou reprezentaci dat, určených pro přenos přes komunikační síť.

Na Obr. 2 je znázorněna slučitelnost jednotlivých zařízení pomocí datových struktur definovaných v ASN.1 [2].



Obr. 2 Vzájemná slučitelnost zařízení pomocí datových struktur definovaných v ASN.1

Výhoda oddělení zápisu v ASN.1 a kódování spočívá v tom, že pro různá technická přenosová prostředí lze definovat různá kódovací pravidla a původní zápis zůstává nezměněn. Dalšími kódovacími pravidly, kromě BER, mohou být například DER, CER, PER, XER aj. Skutečná implementace zařízení se nakonec provádí v některém programovacím jazyku (jakými jsou C nebo Java) a nikoliv v ASN.1.

Shrnutím poznatků uvedených výše můžeme říci, že vývoj zařízení, vycházejícího z definice v ASN.1, v současnosti může probíhat následujícím způsobem:

1. Zápis datové struktury v ASN.1.
2. Dále se použije kompilátor ASN.1, jenž provede kontrolu správnosti zápisu a zkonvertuje definice do cílového programovacího jazyka (např. C nebo Java). Této problematice je podrobněji věnována kapitola 5.

2 Management Information Base (MIB)

Management Information Base (MIB) je databáze spravovaná protokolem SNMP a popisuje vlastnosti a stav spravovaných objektů [6]. Tyto objekty jsou setříděny do stromové struktury, která tak dělí databázi do snadno přehledných částí. Každý z uzlů ve stromové struktuře má své označení, jak číselné tak i slovní. Lze tedy přistupovat pomocí cesty od kořene stromu až k danému uzlu a tato cesta je vždy jednoznačně určena (číselně nebo slovně).

Zařízení, které je spravováno, může implementovat jednu nebo více databází MIB. Tyto databáze jsou naspány podle pravidel SMI (Structure of Management Information), která specifikují způsob zacházení s objekty databáze MIB a orientaci v této databázi. MIB má hierarchicky uspořádanou stromovou strukturu a můžeme ji rozdělit do pěti oblastí:

Configuration Management

Configuration Management shromažďuje jména všech zařízení, která se vyskytují na síti, jejich specifikace a aktuální stav. Díky těmto vlastnostem si administrátor sítě udělá představu o celkovém fyzickém uspořádání sítě.

Performance Management

Performance Management sleduje využití sítě a poskytuje vyhodnocení nasbíraných dat, zátěžové charakteristiky apod. Administrátor si tak udělá obrázek o dostupnosti, průchodnosti, času odezev a využití jednotlivých zařízení.

Fault Management

Fault Management detekuje a opravuje vzniklé problémy. Problémem může být například porucha rozhraní některého síťového prvku signalizovaný varovnou zprávou trap (viz kapitola 1.2.6).

Security Management

Security Management se zabývá bezpečnostní politikou, řídí a chrání dostupnost informací na síti.

Accounting Management

Pomocí Accouting Managementu můžeme snadno změřit vytížení jednotlivých síťových prvků, jako je například množství přenešených dat.

2.1 SNMP Global Naming Tree

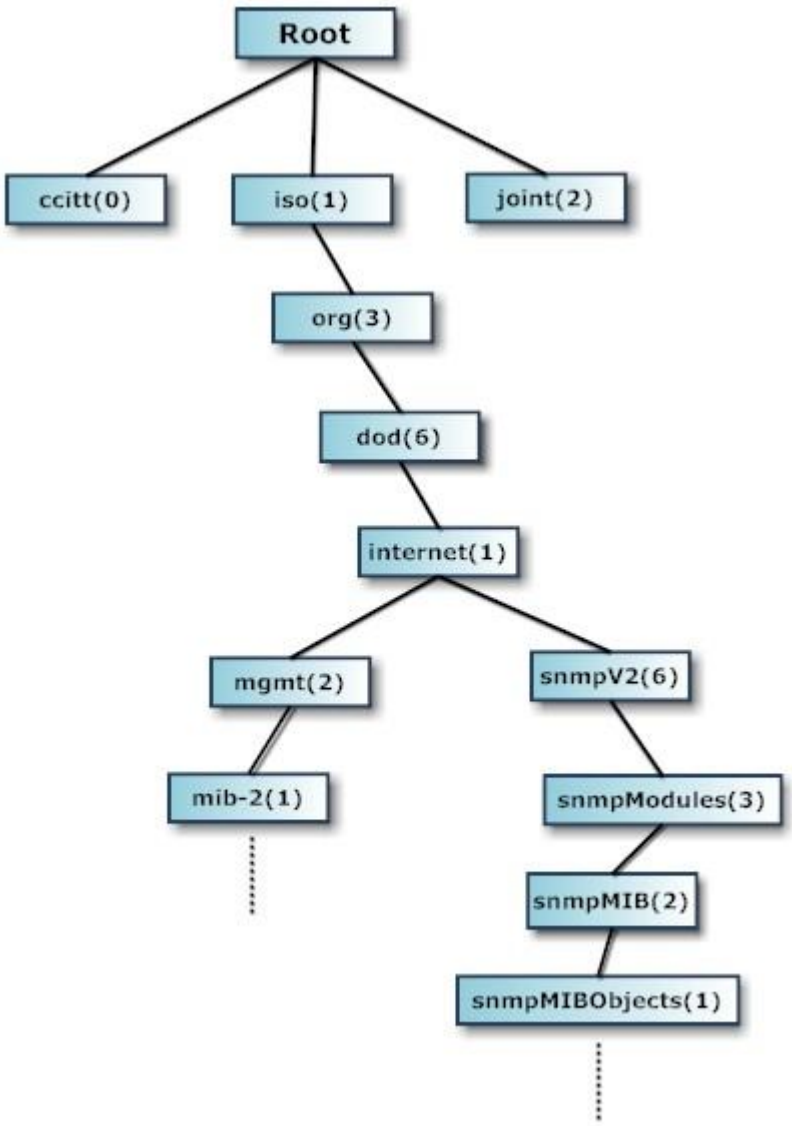
Abychom se při různých SNMP operácích (tyto operace jsou popsány v kapitole 1.2) mohli odkazovat na jednotlivé SNMP objekty, které představují konkrétní zařízení, musí mít tyto objekty jedinečný název. Může se například stát, že jednotlivá zařízení od různých výrobců obsahují stejné objekty. Proto byla vytvořena koncepce hierarchické stromové struktury SNMP Global Naming Tree (vyvinut organizací ISO), aby nemohlo dojít k záměně těchto objektů.

Struktura SNMP Global Naming Tree je členěna na objekty root, subtree a leaf. Každá část stromu má své označení - pomocí krátkého textového popisu a celého čísla. Stromová struktura vychází z kořenového uzlu, který označujeme jako root. Tento uzel root nemá číselné označení, ale pod ním se nacházejí tři uzly:

- *iso(1),*
- *ccitt(0),*
- *joint(2).*

Uzel *iso(1)* je spravován organizací ISO, uzel *ccitt(0)* organizací ITU-T a uzel *joint(2)* společně organizacemi ISO a ITU-T. Výrobcům konkrétních zařízení jsou pak přiřazovány jednotlivé podstromy (subtree) a mohou si dále vytvářet svou vlastní strukturu.

Na konkrétní uzly ve stromové struktuře se můžeme odkázat pomocí OBJECT IDENTIFIER (OID), které je tvořeno sekvencí celých čísel oddělených tečkou na cestě od kořenového uzlu (root) až k danému objektu. Příkladem může být například OID 1.3.6.1.2 větve *mgmt(2).* Na stejnou větev se můžeme odkázat i slovně např. *iso.org.dod.internet.mgmt.* Na Obr. 3 je schematicky znázorněna část této struktury, kde je názorně vidět postavení jednotlivých objektů a cesta k jejich dosažení.



Obr. 3 Ukázka struktury stromu MIB (SNMPv2)

2.2 Typy objektů

V následujících kapitolách budou popsány jednotlivé typy SNMP objektů, které mohou být dvou typů – skalární hodnoty a tabulky [1].

2.2.1 INTEGER

INTEGER je jednoduchý datový typ, který může nabývat hodnot nulových, kladných nebo záporných celých čísel.

2.2.2 Gauge32

Gauge32 představuje nezáporné celé číslo, které se může zvýšit nebo snížit, ale nikdy nepřekročí maximální hodnotu. Maximální hodnota nemůže být větší než 2^{32} - 1 (4294967295 dekadicky). Hodnota Gauge32 dosáhne maxima kdykoli je modelovaná informace větší nebo rovna tomuto maximu.

2.2.3 TimeTicks

Typ TimeTicks představuje nezáporné celé číslo, reprezentující čas, modulo 2^{32} (4294967296 dekadicky), v setinách sekundy měřený mezi dvěma obdobími. TimeTicks obsahuje časovou značku, která měří dobu, která uplynula (v setinách sekundy) od nějaké události (např. dobu uplynulou od zapnutí nějakého zařízení).

2.2.4 Counter32

Typ Counter32 představuje nezáporné celé číslo, který se monotónně zvyšuje do té doby, než dosáhne maximální hodnoty 2^{32} - 1 (4294967295 dekadicky), poté začne počítat znovu od začátku. Čítače nemají pevně definovánu „počáteční“ hodnotu. Používá se například k počítání důležitých událostí v systému.

2.2.5 OBJECT IDENTIFIER

OBJECT IDENTIFIER (OID) představuje administrativně přiřazená jména. Každá instance tohoto typu může mít nanejvýš 128 sub-identifikátorů. Dále, každý sub-identifikátor nesmí přesahovat hodnotu 2^{32}- 1 (4294967295 dekadicky). Zjednodušeně řečeno, OID představuje jméno uzlu v MIB. Příkladem může být třeba OID ifSpeed 1.3.6.1.2.1.2.2.1.5.

2.2.6 OCTET STRING

OCTET STRING představuje libovolná binární data (např. MAC adresy zařízení) nebo řetězce znaků (například jméno systému).

2.2.7 IpAddress

Objekt typu IpAddress představuje 32 - bitové IP adresy.

2.2.8 Další typy

SNMP definuje ještě tři jiné typy skalárních hodnot Opaque, NULL a NetworkAddress. Tyto typy se však nepoužívají.

2.2.8.1 Opaque

Opaque je typ, který dovoluje procházení libovolné ASN.1 syntaxe. Podle základních pravidel ASN.1 se hodnoty zakódují do řetězce oktetů. Tento řetězec je postupně zakódovaný jako OCTET STRING.

2.2.8.2 NULL

Typ NULL je rezervované místo, které oznamuje absenci informace. Například, když manažer žádá hodnotu proměnné, užívá typ NULL jako rezervované místo v pozici, kde agent vyplní odpověď.

2.2.8.3 NetworkAddress

Tento typ je zastaralý a používá se typ ze SMIv2, typ IpAddress.

Rozšířením těchto objektů je, že dle standardu SNMP lze data uspořádat do tabulek [6]. Tyto tabulky jsou uspořádány do řádků a sloupců, kde položky této tabulky jsou skalární hodnoty uvedené výše. Tabulky nelze do sebe vnořovat. SNMP operace lze provádět pouze nad jednotlivými skalárními objekty (hodnotami v tabulce), nikoli nad tabulkou jako celkem.

Jednotlivá pole v SNMP tabulkách identifikujeme pomocí indexů. K těmto polím můžeme přistupovat náhodně, tedy příkazem get a bližší určení pozice nebo sekvenčně, tedy příkazem get-next, kterým procházíme celou tabulku.

2.3 Základní elementy objektů

Každý objekt má řadu atributů, jakými jsou SYNTAX, ACCESS, STATUS a DESCRIPTION.

2.3.1 SYNTAX

SYNTAX definuje datovou strukturu objektů. Příkladem těchto datových struktur můžou být typy dat jakými jsou INTEGER, OCTET STRING, nebo NULL. SYNTAX také definuje zvláštní případy jednoduchých objektů, zahrnující například DisplayString, který je omezený na tisknutelné ASCII znaky. Tabulkové objekty užívají syntaxi SEQUENCE OF.

2.3.2 ACCESS

ACCESS definuje minimální úroveň přístupu k objektu (nebo podpoře). ACCESS může mít hodnoty read-only, read-write, not-accessible, nebo write-only. SNMP nedovoluje hodnotu write-only.

2.3.3 STATUS

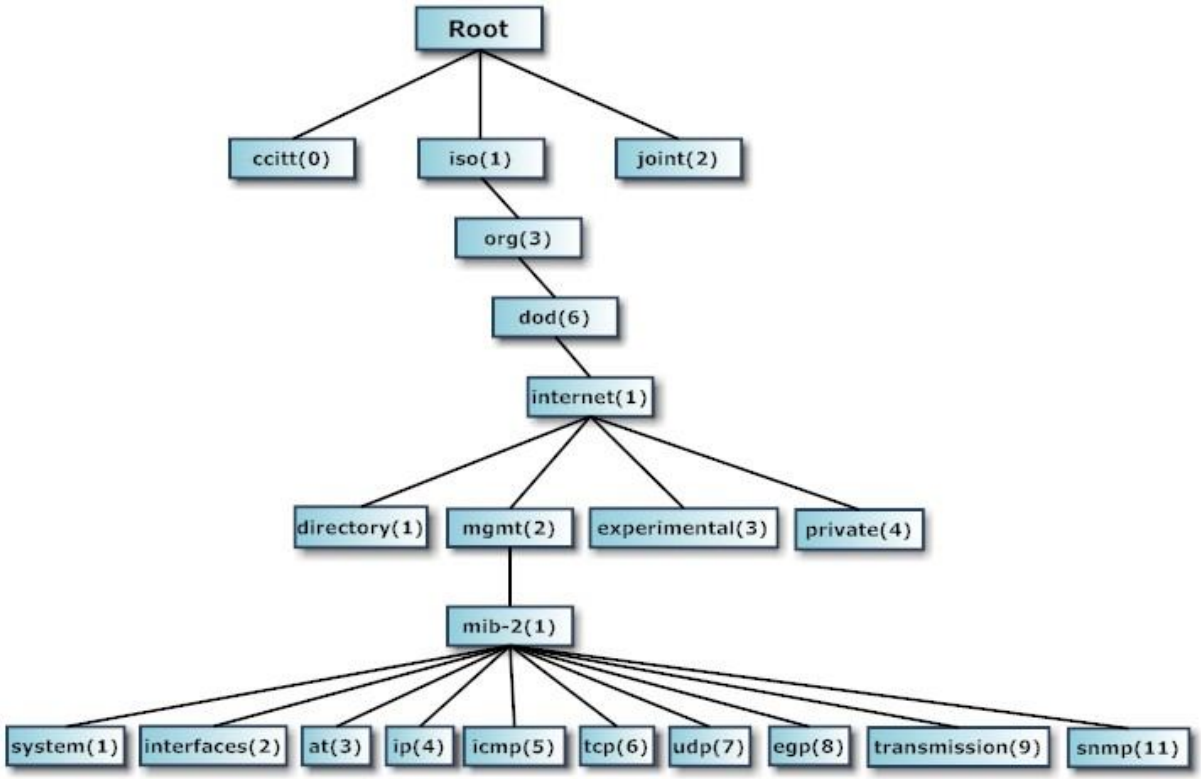
STATUS definuje podporu implementace pro objekt, který může být mandatory (povinný), optional (volitelný), deprecated (neschválený) nebo obsolete (zastaralý). Jestliže STATUS definuje úroveň podpory pro vybranou skupinu, pak tato úroveň platí pro všechny objekty uvnitř skupiny. Objekty, které byly přemístěny zpětnou kompatibilitou objektů, jsou deprecated(neschválené). Objekty, které nejsou dlouho podporované jsou obsolete (zastaralé).

2.3.4 DESCRIPTION

DESCRIPTION popisuje blíže daný objekt. Tento popis není povinný.

3 Stromová struktura MIB II

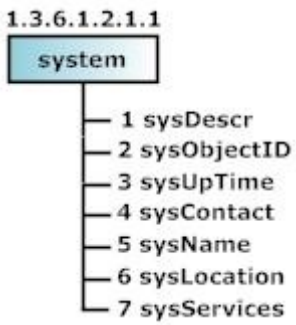
Tato kapitola bude věnována bližšímu popisu větve MIB-II [3]. Tato větev je velmi důležitá. Protože každé zařízení, které podporuje SNMP, musí také podporovat MIB- II. MIB-II je současný standard pro SNMP správu objektů. Má 10 základních skupin: system, interfaces, at, ip, icmp, tcp, udp, egp, transmission a snmp (viz Obr. 4) [4].



Obr. 4 Podstrom MIB II

3.1 Skupina System

Skupina System poskytuje textový popis zařízení v tisknutelných ASCII znacích. Tento textový popis zahrnuje popis systému, OID a objekt, který měří čas (v setinách sekundy) od posledního spuštění nebo „resetování“systému a další administrativní detaily. Implementace skupiny system je povinná. OID pro skupinu system je 1.3.6.1.2.1.1.

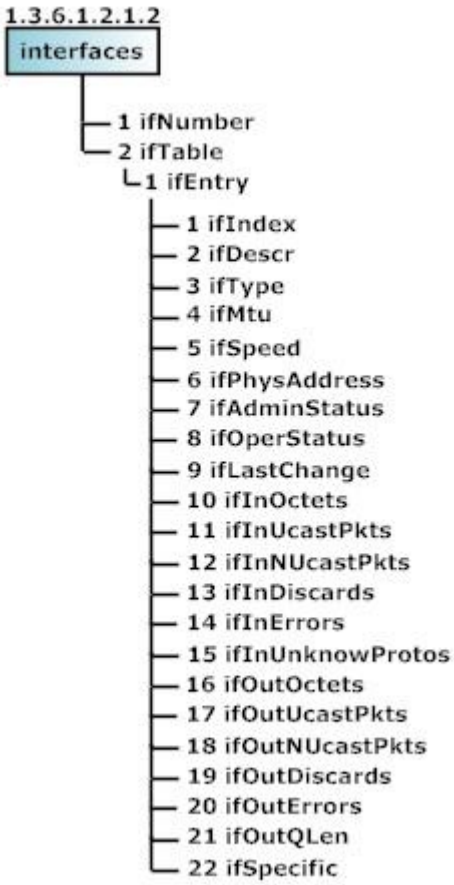


Obr. 5 Skupina System

3.2 Skupina Interfaces

Skupina Interfaces poskytuje informaci o hardwarových rozhraních zařízení. První objekt (ifNumber) udává počet rozhraní na zařízení. Pro každé rozhraní, je vytvořen jeden záznam

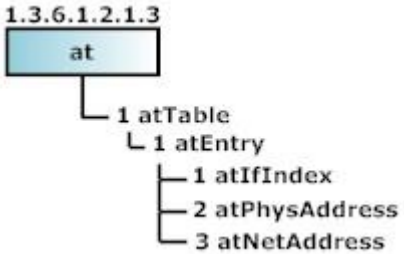
do tabulky s 22 sloupci na řádek. Sloupcové položky poskytují informaci o rozhraních, jako je rychlost rozhraní, fyzická (hardwarová) adresa, aktuální operační stav a statistiky provozu. OID pro skupinu interfaces je 1.3.6.1.2.1.2.



Obr. 6 Skupina Interfaces

3.3 Skupina Address Translation

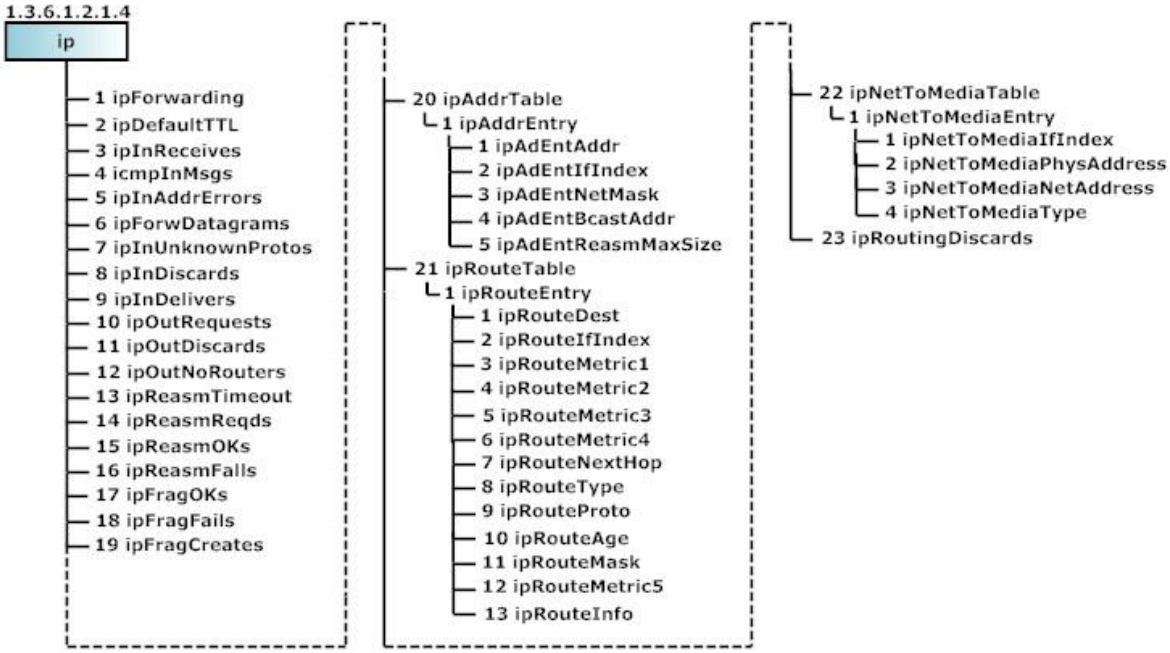
Databáze MIB-I zahrnovala skupinu Address Translation (AT), která ale v MIB-II nebyla schválena. Status "neschválený" znamená, že MIB-II sice zahrnuje tuto skupinu z důvodu zpětné kompatibility s MIB- I, ale z budoucích verzí MIB bude pravděpodobně vyloučena. Skupina Address Translation poskytovala tabulku, která překládala mezi IP adresami a fyzickými (hardwarovými) adresami. V MIB-II a budoucích verzích, každá skupina protokolů bude obsahovat svůj vlastní překlad tabulek. Tato skupina obsahuje jednu tabulku s třemi sloupci na řádek. OID pro skupinu Address Translation je 1.3.6.1.2.1.3.



Obr. 7 Skupina Address Translation

3.4 Skupina IP

Skupina IP je tvořena z individuální konfigurace, statistik provozu a tří tabulek (ipAddrTable, ipForwardTable a ipNetToMediaTable,). Tyto informace jsou potřebné pro zařízení a směrovače pracující s protokolem IP. OID pro skupinu IP je 1.3.6.1.2.1.4.



Obr. 8 Skupina IP

3.4.1 Tabulka ipAddrTable

Každému síťovému rozhraní je přiřazená IP adresa, což znamená, že je v tabulce zápis pro každou IP adresu. Někdy je počet IP adres větší než počet rozhraní. To proto, že jednotlivým rozhraním může být přiřazena víc než jedna IP adresa. Všechny informace v této tabulce jsou pouze pro čtení.

3.4.2 Tabulka ipForwardTable

Tabulka ipForwardTable je alternativa k originální tabulce ipRouteTable, která obsahuje informace potřebné k směrování datagramů. Ve skutečnosti je tabulka ipForwardTable indikována cílem směrování, užívaným protokolem a metodou doručování. Zatímco tabulka ipRouteTable je indikována jen cílem směrování.

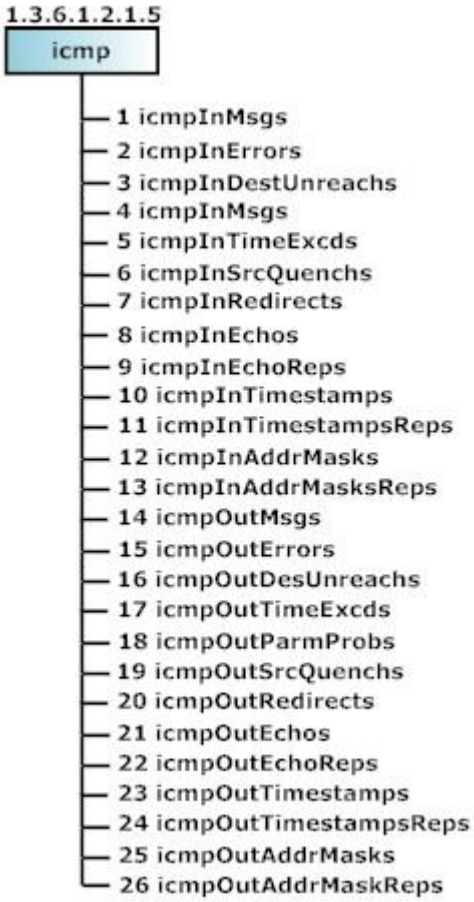
3.4.3 Tabulka ipNetToMediaTable

Cílem této tabulky je nahradit momentálně neschválenou tabulku překladu adres (atTable), která je popsána v kapitole 3.3 Skupina Address Translation. Tato nová tabulka zahrnuje proměnnou ipNeToMediaType, která signalizuje zda je záznam statického typu (ručně vložený), nebo zda byl objevený dynamickým protokolem jako je například ARP (Address Resolution Protocol).

3.5 Skupina Internet Control Message Protocol

Skupina Internet Control Message Protocol (ICMP), je povinná komponenta sady protokolů TCP/IP a je definována v RFC 792. Skupina ICMP poskytuje informace spojené s procesem komunikace v síti a o různých operacích ICMP uvnitř spravovaného zařízení. Skupina ICMP obsahuje 26 skalárních objektů, které udržují statistiky pro různé ICMP

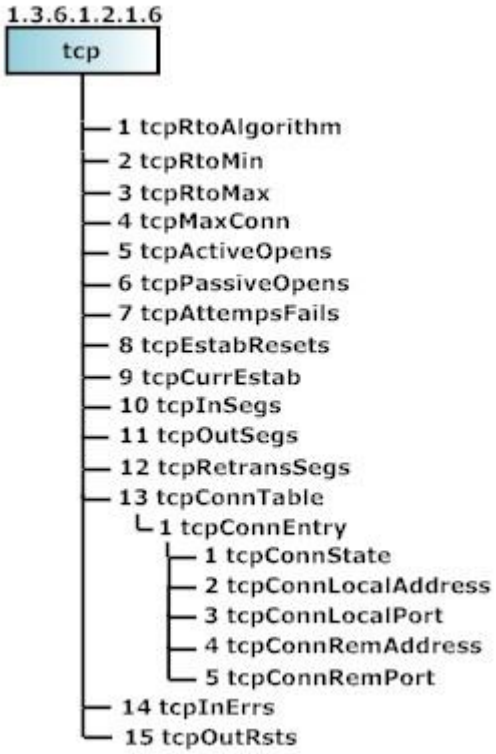
zprávy, jako je např. počet přijatých ICMP Echo Request zpráv nebo počet poslaných ICMP Redirect zpráv. OID této skupiny je 1.3.6.1.2.1.5.



Obr. 9 Skupina Internet Control Message Protocol

3.6 Skupina Transmission Control Protocol

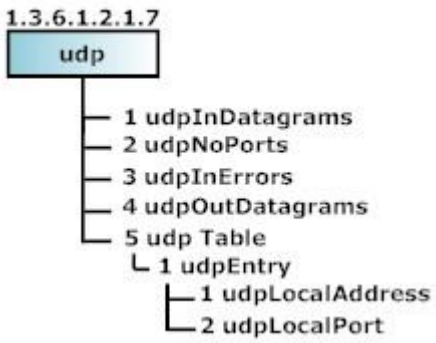
Skupina Transmission Control Protocol (TCP) je povinná a poskytuje informace týkající se TCP spojení a příslušných operací. Tato skupina obsahuje 14 skalárních objektů a jednu tabulku. Skalární objekty zaznamenávají různé parametry a statistiky protokolu TCP, jako počet TCP spojení, která zařízení jsou podporována nebo celkový počet přenesených TCP segmentů. Tabulka, tcpConnTable, obsahuje informaci vztahující se k jednotlivým TCP spojením. OID pro tuto skupinu je 1.3.6.1.2.1.6.



Obr. 10 Skupina Transmission Control Protocol

3.7 Skupina User Datagram Protocol

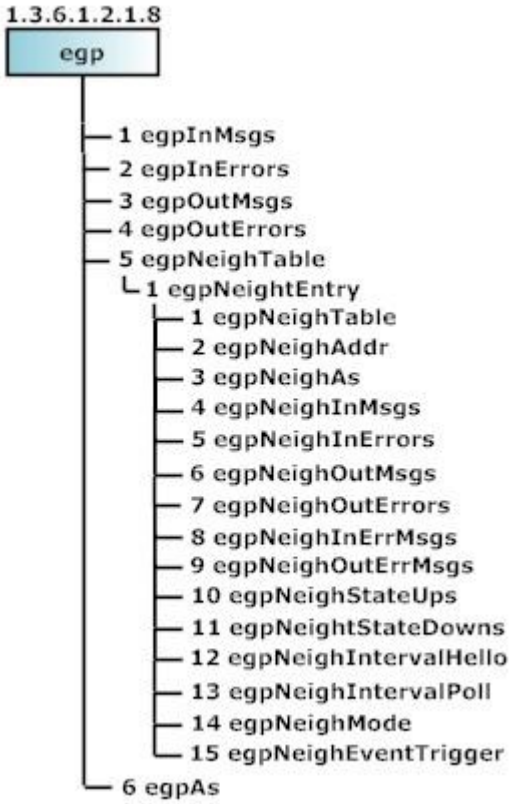
Skupina User Datagram Protocol (UDP), je povinná a poskytuje informace týkající se UDP operací. Protože protokol UDP je nespojový, je tato skupina mnohem menší než skupina pro spojově-orientovaný protokol TCP. Skupina UDP obsahuje čtyři skalární objekty a jednu tabulku. Skalární objekty udržují statistiky o datagramech UDP, jako je počet datagramů poslaných z daného zařízení. Tabulka, udpTable, obsahuje adresu a informaci o portech UDP. OID pro tuto skupinu je 1.3.6.1.2.1.7.



Obr. 11 Skupina User Datagram Protocol Group

3.8 Skupina Exterior Gateway Protocol

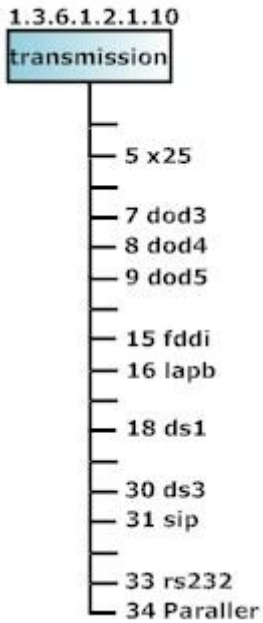
Skupina Exterior Gateway Protocol (EGP), je povinná pro všechny systémy, které implementují EGP. EGP je směrovací protokol využívaný mezi autonomními systémy, podle RFC 904. Uvnitř autonomních systémů se pro směrování používají protokoly typu Internal Gateway Protocol (IGP), např. RIP nebo OSPF. Skupina EGP zahrnuje 5 skalárních objektů a jednu tabulku. Skalární objekty udržují statistiky zpráv EGP. Tabulka, egpNeighTable, obsahuje informace o sousedních směrovačích EGP. OID pro tuto skupinu je 1.3.6.1.2.1.8.



Obr. 12 Skupina Exterior Gateway Protocol

3. 9 Skupina Transmission

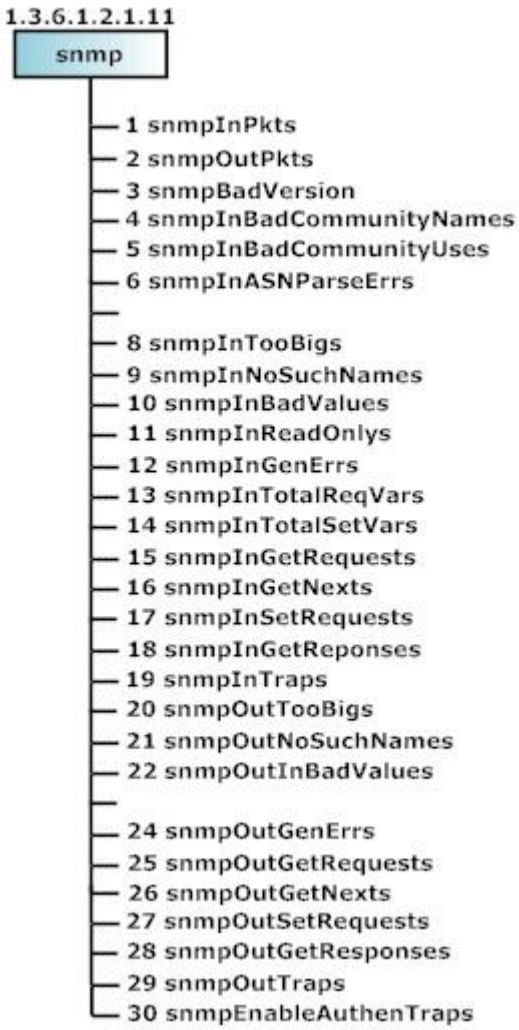
Skupina Transmission, určena 1.3.6.1.2.1.10 OID, obsahuje objekty, které se vážou k přenosu dat na úrovni linkové vrstvy.



Obr. 13 Skupina Transmission

3.10 Skupina SNMP

Skupina SNMP poskytuje informaci o SNMP objektech. V této skupině je celkem 30 skalárních objektů včetně statistik SNMP zpráv, počtu získaných MIB objektů a počtu poslaných SNMP trapů. OID pro tuto skupinu je 1.3.6.1.2.1.11.



Obr. 14 Skupina SNMP

4 Popis skupiny Interfaces MIB – SMIv2 podle RFC2233

Dokument RFC 2233 [5] popisuje skupinu interfaces podstromu MIB- II. Pro textovou reprezentaci MIB jsou využita syntaktická pravidla jazyka ASN.1 (Abstract Syntax Notation One). Kapitolu 4 věnuji popisu zjednodušené MIB, která vychází z této skupiny a kterou jsem vytvořil v rámci řešení projektu.

4.1 „Hlavička“ MIB

MIB soubory začínají definicí jména MIB (viz ukázka níže - InterfaceMIB), které musí začínat velkým písmenem. Příkaz BEGIN a END uzavírají tělo MIB souboru. Tělo obsahuje část IMPORTS, která importuje datové typy, hodnoty, makra a moduly deklarované v jiných modulech.

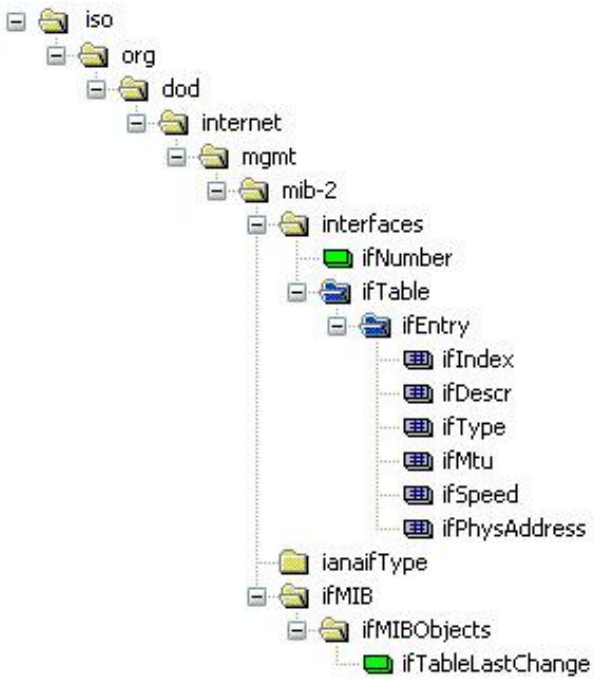
V našem zdrojovém kódu jsou importovány typy Gauge32, Integer32, TimeTicks, PhysAddress, mib-2, TEXTUAL-CONVENTION, DisplayString a IANAifType. Každá skupina položek v části IMPORTS má dodatek FROM, který ukazuje zdroj (modul), z kterého objekty musí být importovány.

Ukázka části MIB:

```
InterfaceMIB DEFINITIONS ::=

BEGIN

IMPORTS
    MODULE-IDENTITY, OBJECT-TYPE,
    Gauge32, Integer32, TimeTicks, mib-2  FROM SNMPv2-SMI
    TEXTUAL-CONVENTION, DisplayString,
    PhysAddress                            FROM SNMPv2-TC
    IANAifType                            FROM IANAifType-MIB;

        .
        .
        .
        .
END
```

4.2 InterfaceMIB objekty

Na Obr. 15 je zobrazena, pomocí prohlížeče MIB, zjednodušená struktura, kterou jsem vytvořil. V následujících kapitolách jsou podrobně popsány objekty [1], které tato struktura obsahuje.



Obr. 15 Zjednodušená stromová struktura MIB podle RFC 2233

4.2.1 Skupina Interfaces

Informace o dostupných hardwarových rozhraních na zařízení je poskytována skupinou interfaces. Tyto informace jsou prezentovány formou tabulky. OID pro tuto skupinu je 1.3.6.1.2.1.2.

4.2.1.1 ifNumber

Popis: MIB II definuje objekt, ifNumber, jehož hodnota představuje počet síťových rozhraní (bez ohledu na jejich aktuální stav) existujících v daném systému.

Object Name:	ifNumber
Object ID:	1.3.6.1.2.1.2.1
Object Syntax:	INTEGER
Object Access:	read-only
Object Status:	current

4.2.2 ifTable

Popis: Tabulka obsahuje záznam popisující jedno konkrétní rozhraní. Počet záznamů je dán hodnotou ifNumber.

Object Name:	ifTable
Object ID:	1.3.6.1.2.1.2.2
Object Syntax:	Table
Object Access:	not-accessible
Object Status:	current

4.2.2.1 ifEntry

Popis: Záznam obsahující příslušné řídicí informace spojené s rozhraním.

Object Name:	ifEntry
Object ID:	1.3.6.1.2.1.2.2.1
Object Syntax:	SEQUENCE

Object Access: not-accessible
Object Status: current

4.2.2.1.1 ifIndex

Popis: ifIndex je jedinečná hodnota, z rozsahu 1 až ifNumber, která je postupně přiřazena jednotlivým rozhraním. Hodnota pro každé rozhraní musí zůstávat konstantní přinejmenším od jedné reinicializace systému k další.

Object Name: ifIndex
Object ID: 1.3.6.1.2.1.2.2.1.1
Object Syntax: INTEGER
Object Access: read-only
Object Status: current

4.2.2.1.2 ifDescr

Popis: ifDescr je textový řetězec obsahující informaci o rozhraní. Tento řetězec by měl zahrnovat jméno výrobce, jméno produktu a verze hardwaru/softwaru daného rozhraní.

Object Name: ifDescr
Object ID: 1.3.6.1.2.1.2.2.1.2
Object Syntax: OCTET STRING
Object Access: read-only
Object Status: current

4.2.2.1.3 ifType

Popis: ifType specifikuje typ rozhraní, jako Ethernet, token ring, FDDI, frame relay, atd. Další hodnoty pro tabulku ifType jsou přiděleny organizací Internet Assigned Numbers Authority (IANA) během aktualizace modulu IANAifType (přídavný modul k souboru interfaceMIB).

Object Name: ifType
Object ID: 1.3.6.1.2.1.2.2.1.3
Object Syntax: INTEGER
Object Access: read-only
Object Status: current

4.2.2.1.4 ifMtu

Popis: ifMtu je maximální povolená velikost datové jednotky, která může být poslána/přijata na rozhraní. Velikost je vyjádřena v oktetech.

Object Name: ifMtu
Object ID: 1.3.6.1.2.1.2.2.1.4
Object Syntax: INTEGER
Object Access: read-only
Object Status: current

4.2.2.1.5 ifSpeed

Popis: Jedná se o odhad aktuální šířky pásma rozhraní v bitech za sekundu. Pro rozhraní, u kterých kolísá šířka pásma nebo pro ty, kde nemůže být žádný předběžný výsledek vypočítán, by tento objekt měl obsahovat přibližnou šířku pásma. Jestliže šířka pásma rozhraní je větší než maximální hodnota, kterou může objekt vyjádřit, pak by tento objekt měl oznámit svou maximální hodnotu (4,294,967,295) a je třeba využít objekt ifHighSpeed pro ohlášení rychlosti rozhraní.

Object Name: ifSpeed
Object ID: 1.3.6.1.2.1.2.2.1.5
Object Syntax: Gauge32
Object Access: read-only
Object Status: current

4.2.2.1.6 ifPhysAddress

Popis: ifPhysAddress je fyzická adresa rozhraní. Např. pro rozhraní 802.x by měl tento objekt za normálních okolností obsahovat MAC adresu. Pro správné vyhodnocení je důležité definovat bitové a bajtové uspořádání a formát hodnoty tohoto objektu. Pro rozhraní, která nemají takovou adresu (např. sériová linka), by měl tento objekt obsahovat řetězec typu OCTET STRING nulové délky.

Object Name: ifPhysAddress
Object ID: 1.3.6.1.2.1.2.2.1.6
Object Syntax: OCTET STRING
Object Access: read-only
Object Status: current

5 Kompilátory

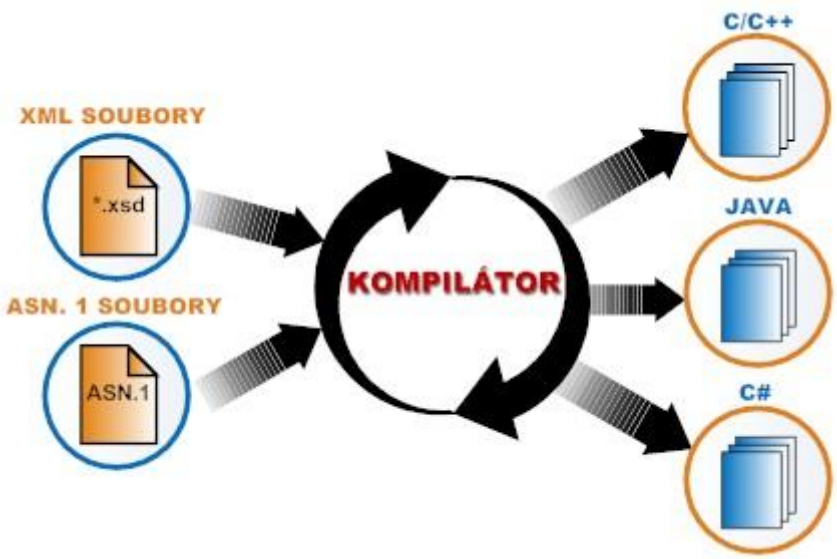
V kapitole 4 jsem popsal námi vytvořenou strukturu v jazyce ASN.1. Dalším krokem po vytvoření této struktury je její následný převod do jazyka C pomocí kódovacích pravidel BER. K tomuto účelu jsem použil kompilátor ASN.1, jenž provede kontrolu správnosti zápisu a zkonvertuje definice ASN.1 do cílového programovacího jazyka (v tomto případě do jazyka C). Definice jsou zkonvertovány do souborů dvojího typu:

- definice typů v cílovém jazyku (s nimiž se dobře pracuje v cílovém jazyku a prostředí),
- konverzní funkce, jež překládají definované typy v cílovém jazyku do přenosové syntaxe (většinou jsou ve formě knihovny).

Vyzkoušel jsem 2 volně dostupné kompilátory, které jsou popsány v následujících kapitolách.

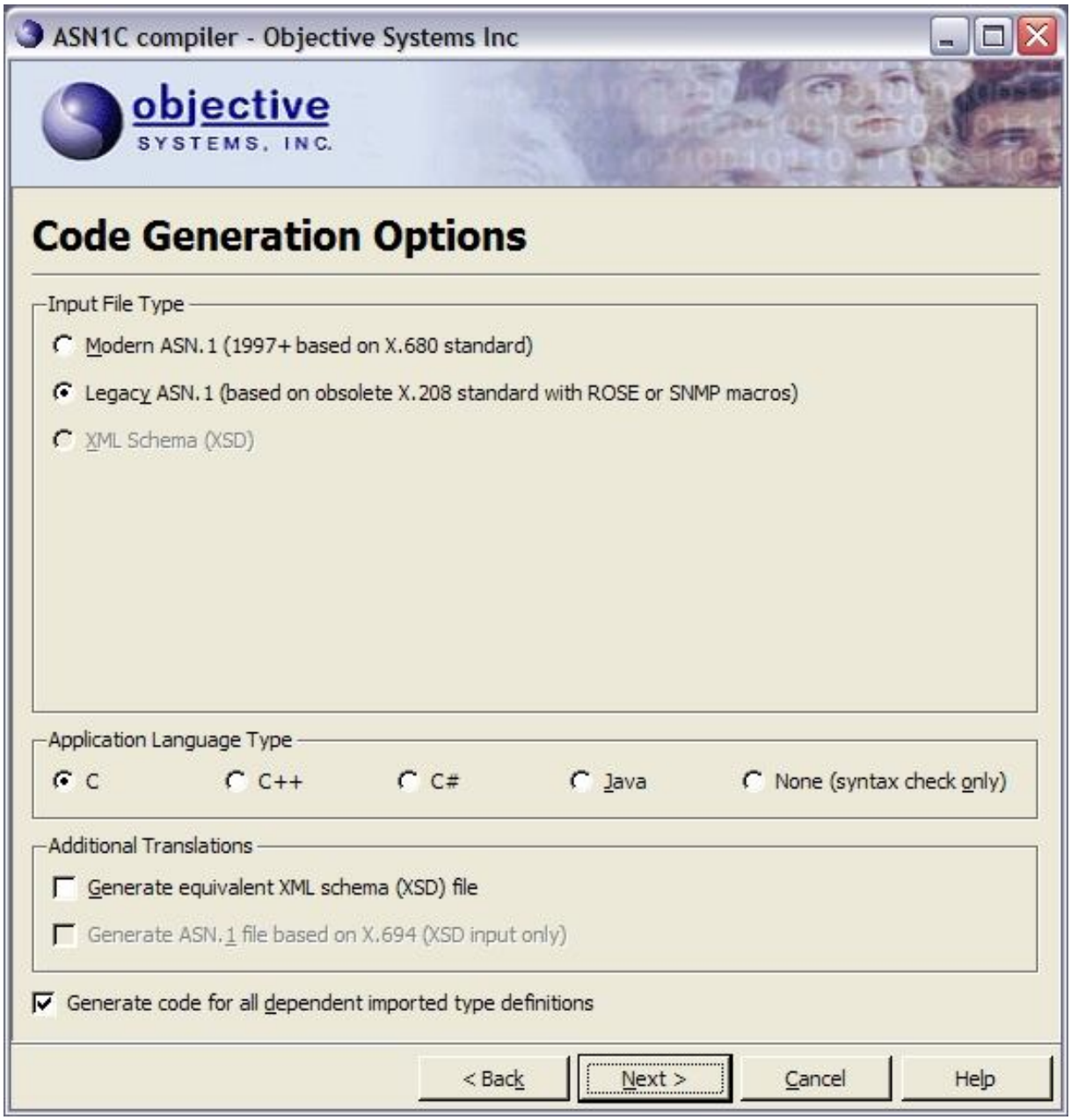
5.1 Kompilátor ASN1C

Kompilátor ASN1C od společnosti Objective System překládá ASN.1 a schémata XML do jazyků: C, C++, C# nebo do zdrojového kódu jazyku Java. Tento kompilátor umožňuje uživateli vybírat mezi několika kódovacími pravidly, jakými jsou BER, CER, DER, PER nebo XER.



Obr. 16 Blokové schéma činnosti kompilátoru ASN1C

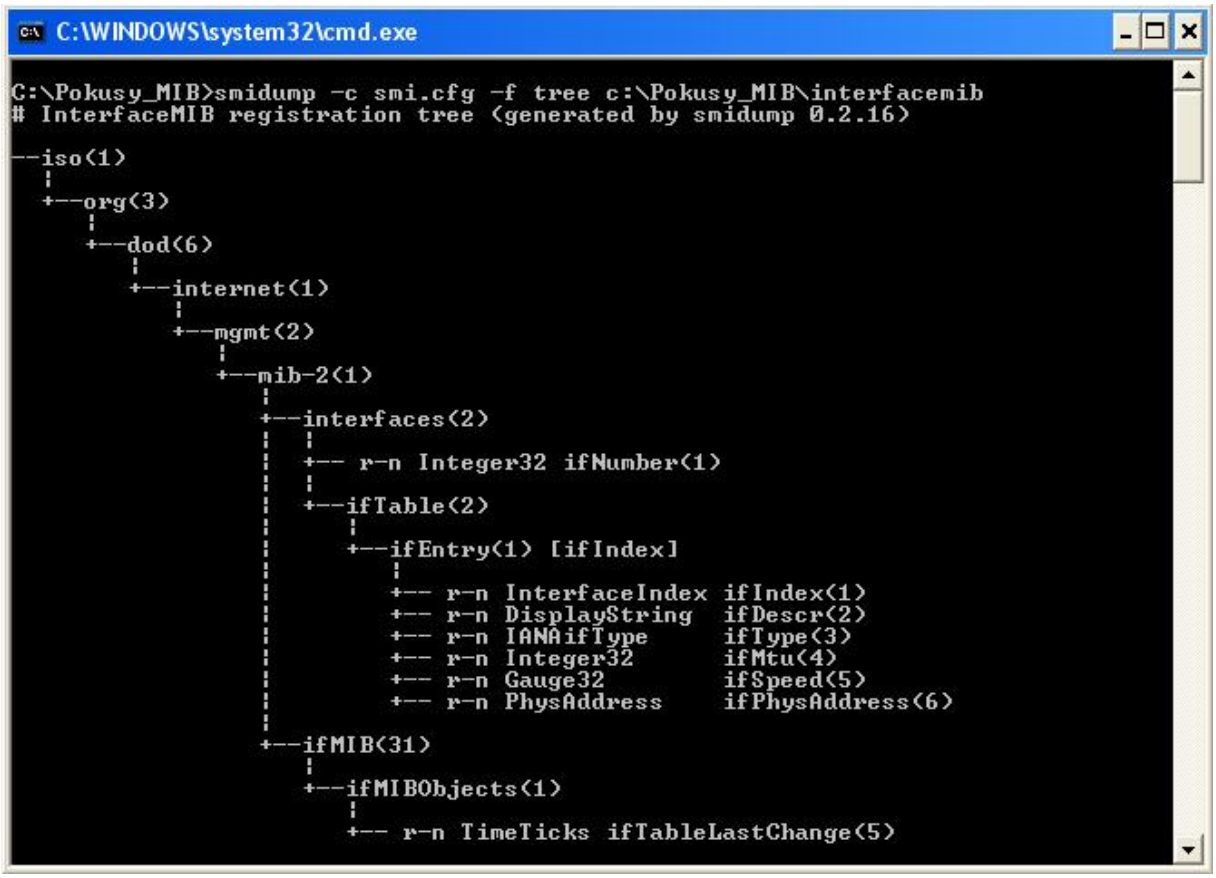
K vytvořenému souboru MIB bylo třeba přiložit moduly IANAifType-MIB, SNMPv2-SMI a SNMPv2-TC (viz kapitola 4). Zdrojový kód vygenerovaný ANS1C kompilátorem byl poněkud těžkopádný a nesnadno použitelný v programu OPNET Modeler, který si nedokázal poradit s velkým množstvím potřebných knihoven.



Obr. 17 Grafické prostředí programu ASN1C compiler

5.2 DMH SMIv2 MIB-Compiler

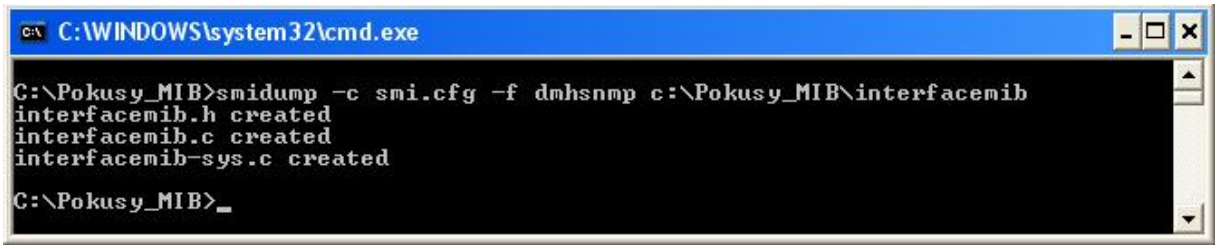
DMH SMIv2 MIB-Compiler se ovládá pomocí příkazové řádky a kromě kompilace do jazyka C a Javy umožňuje celou řadu funkcí, jakými jsou například vykreslení stromové struktury s OID (viz obr. 18), výpis importovaných modulů, výpis všech objektů ve struktuře a další.



Obr. 18 Vykreslení stromové struktury programem DMH SMIv2 MIB-Compiler

Po překladu MIB z jazyka ASN.1 jsou celkem vygenerovány 3 soubory. Dva soubory s příponou *.c a jeden s příponou *.h:

1. Soubor interfacemib.c (název souboru odpovídá zvolenému názvu MIB) je hlavním souborem v jazyce c, do kterého můžeme vkládat další obslužní funkce.
2. Soubor interfacemib.h (název souboru odpovídá zvolenému názvu MIB) je hlavičkovým souborem k výše popsanému souboru interfacemib.c. Soubor interfacemib.h zahrnuje navržené struktury pro skalární a tabulkové MIB objekty a funkce get (např. sysget_<function>, sysalloc_<ent>, sysget_interfacemib_scalars apod.)
3. Soubor interfacemib-sys.c je generován automaticky. Soubor zahrnuje funkce (sysget_<func>) našeho (hostujícího) systému k získání aktuálních hodnot objektů MIB.



Obr. 19 Kompilace do jazyka C

K vytvořenému souboru MIB nebylo třeba přikládat žádné další moduly, protože jsou již přiloženy v instalačním adresáři toho programu. Bylo pouze nutné editovat konfigurační soubor s příponou *.cfg, kde jsem nastavil cesty k těmto přídavným modulům. Vygenerovaný zdrojový kód v jazyce C byl srozumitelnější, než tomu bylo u předchozího kompilátoru, ale

zůstával problém s knihovnami, které program ke své funkci potřebuje, a jejich implementací do programu OPNET Modeler.

6 OPNET Modeler

Podrobný teoretický rozbor k programu OPNET Modeler s vysvětlením jednotlivých komponent tohoto programu (stavů, bloků, procesních modelů apod.) můžeme nalézt v práci [10] nebo [11]. Proto zde popíši pouze základní informace, které jsou nezbytné k pochopení postupů při řešení bakalářské práce v programu OPNET Modeler.

ICI (Interface Control Information)

Interface Control Information (dále jen ICI) je datová struktura, která se v prostředí OPNET Modeler používá k přenosu řídicích informací mezi procesy, přičemž využívá některého z typů přerušení, které jsou popsány níže. ICI může být například zrušeno/odstraněno (destroyed) procesem, který jej vytvořil nebo jedním z procesů, který jej přijal. ICI se skládá z jednotlivých datových částí, které nazýváme atributy. Atributy mohou být typu *integer*, *double* nebo *structure.*

Přerušení (interrupts)

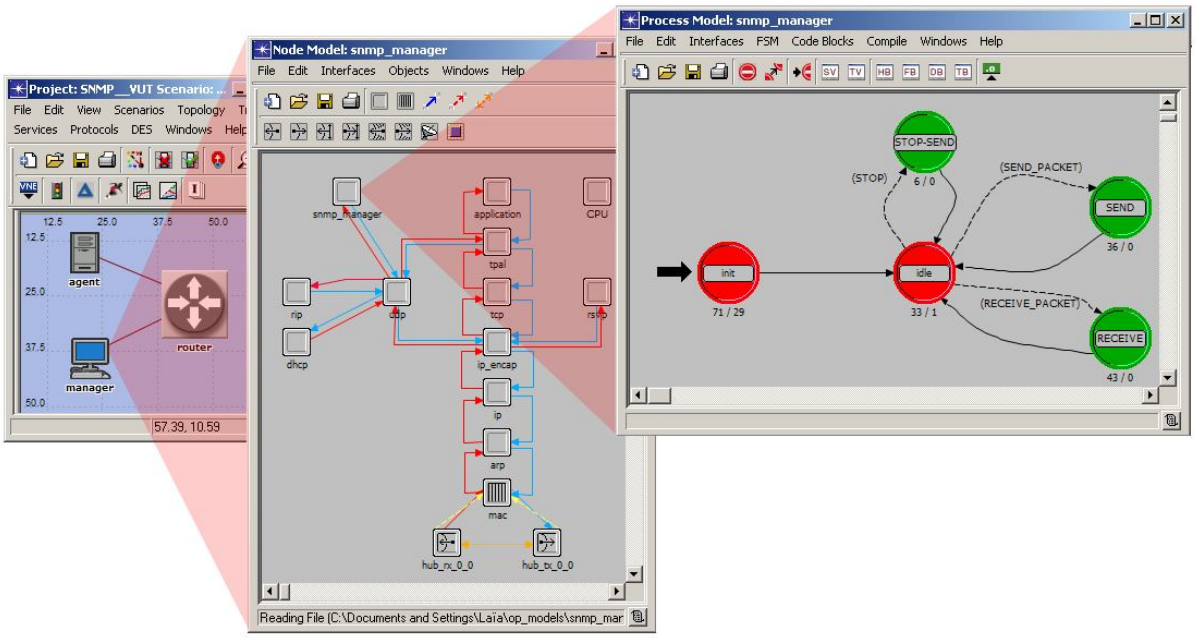
Komunikace mezi procesy je podporována přerušeními (interrupts). Procesy jsou řízeny událostmi. Událost, která je doručena k procesu, se nazývá přerušení. Proces je přerušen nebo vyvolán následkem příchozího přerušení, které signalizuje nějakou událost, jakou je například příchozí zpráva. Tento proces si může vybrat, zda přerušení ignoruje nebo vykoná vlastní kód. Příklady typických událostí vedoucích k přerušením mohou být přijetí nového paketu (stream interrupt), vypršení doby časovače (self-interrupt), přerušení, které vede k inicializaci začátku simulace (begin sim interupt - viz stav INIT manažera i agenta), přerušení vyvolávající konec simulace (end sim interupt) apod .

ICI ve spolupráci s přerušeními je v našem projektu hojně využíváno, zejména pro získání informací z ICI pomocí přerušení.

6.1 Manažer

Komunikace našeho modelu je založena na výměně zpráv (paktů) mezi agentem a manažerem. Manažer dynamicky vygeneruje paket a jeho strukturu naplní daty. Tento paket je následně odeslán agentovi, který jej příjme na UDP vrstvě. Agent si vyčte strukturu paketu a paket následně zahodí. V dalším kroku agent vygeneruje paket a odešle ho jako odpověď manažerovi. Náš model je zjednodušený, proto nepoužívá varovné zprávy trap (viz kapitola 1.2.6).

V následujícím textu budou popsány jednotlivé části procesního modelu manažera v editoru procesů [11]. Cesta jednotlivými úrovněmi modelu k tomuto editoru procesu názorně zobrazuje Obr. 20.



Obr. 20 Cesta úrovněmi modelu k editoru procesu u modelu manažera

6.1.1 Manažer – blok SV

Blok SV (State Variables) - stavové proměnné slouží pro deklaraci proměnných a jejich datových typů. V editoru procesů (Process Model) klikneme na ikonu stavové proměnné SV (State Variables) a zobrazí se nám okno editoru. Zde klineme na Edit ASCII, čímž se přepneme do textového módu.

```
/* ID číslo procesu */
Objid  \my_obj_id;

/* ID číslo rodiče */
Objid  \parent_obj_id;

/* ID číslo procesu UDP */
Objid  \udp_obj_id;

char   \status;

/* Adresa o velikosti 16 znaků */
char   \rem_addr_string [16];

/* Lokální port */
UdpT_Port    \loc_port;

/* Vzdálený port */
UdpT_Port    \rem_port;

/* IP adresa SNMP agenta (serveru, směrovače) */
IpT_Address  \rem_addr;

/* Čas, kdy manažer začne posílat pakety k agentovi */
double \start_time;

/* Čas, kdy manažer zastaví posílání paketů k agentovi */
double \stop_time;

/* Časový interval mezi generováním posledního paketu a dalšího paketu */
int    \next_intarr_time;

/* Přístup k naplánovaným událostem */
Evhandle     \next_send_evh;
```

```
/* Kód přerušení */
int    \intrpt_code;

/* Community string manažera */
char   \community_string_m;
```

6.1.2 Manažer – blok TV

Blok TV (Temporary Variables) – dočasné proměnné použijeme k deklaraci proměnných, jejichž hodnota má platnost pouze po dobu trvání jednoho procesu. Zdrojový kód bloku TV je následující:

```
Ici* ici_ptr;             \\ukazatel na ICI
Packet* send_paket;       \\datový typ OM
Packet* recv_paket;       \\ datový typ OM


SNMPPAKET* datapaket;     \\proměnná odkazující na strukturu SNMPPAKET
char* error_string;       \\proměnná pro chybové hlášky
```

6.1.3 Manažer – blok HB

Blok HB (Header Block) - hlavičkový blok slouží k deklaraci hlavičky aktuálního procesu, proměnných, maker, nových datových typů, vkládání externích knihoven apod. Zdrojový kód bloku HB je následující:

```
/* Vložení hlavičkových souborů */
#include <stdlib.h>
#include <udp_api.h>
#include <ip_addr_v4.h>

/* Definování připojení k UDP streamu */
#define UDPSTRM 0

/* Definice hodnot přerušení pro řízení plánováni */
#define MNGR_START_SEND          10     \\začátek odesílání paketů
#define MNGR_SEND                11     \\odesílání paketů
#define MNGR_STOP_SEND           12     \\odesílání paketů zastaveno
#define MNGR_DISABLED_SEND       13     \\odesílání paketů zakázáno

/* Definice pro přechody mezi stavy */
#define STOP              (op_intrpt_type() == OPC_INTRPT_SELF && \
                                       intrpt_code == MNGR_STOP_SEND)

#define SEND_PACKET       (op_intrpt_type() == OPC_INTRPT_SELF && \
                                       intrpt_code == MNGR_SEND)

#define RECEIVE_PACKET    (op_intrpt_type() == OPC_INTRPT_STRM && \
                                       intrpt_code == UDPSTRM)

/* Vlastnost manažera – odesílání paketu běží do konce simulace */
#define            MNGR_INFINITE_TIME        -1.0

char parent_obj_name[128];

/* Struktura SNMP paketu */
typedef struct variablebin{
      char *ObjectV;            \\OID žádaného objektu
      char *ValueV;             \\Vyčtená hodnota objektu
} VARIABLEBIN;

/*SNMP PDU*/
typedef struct snmppdu{
      int PDUtype;              \\Typ PDU
```

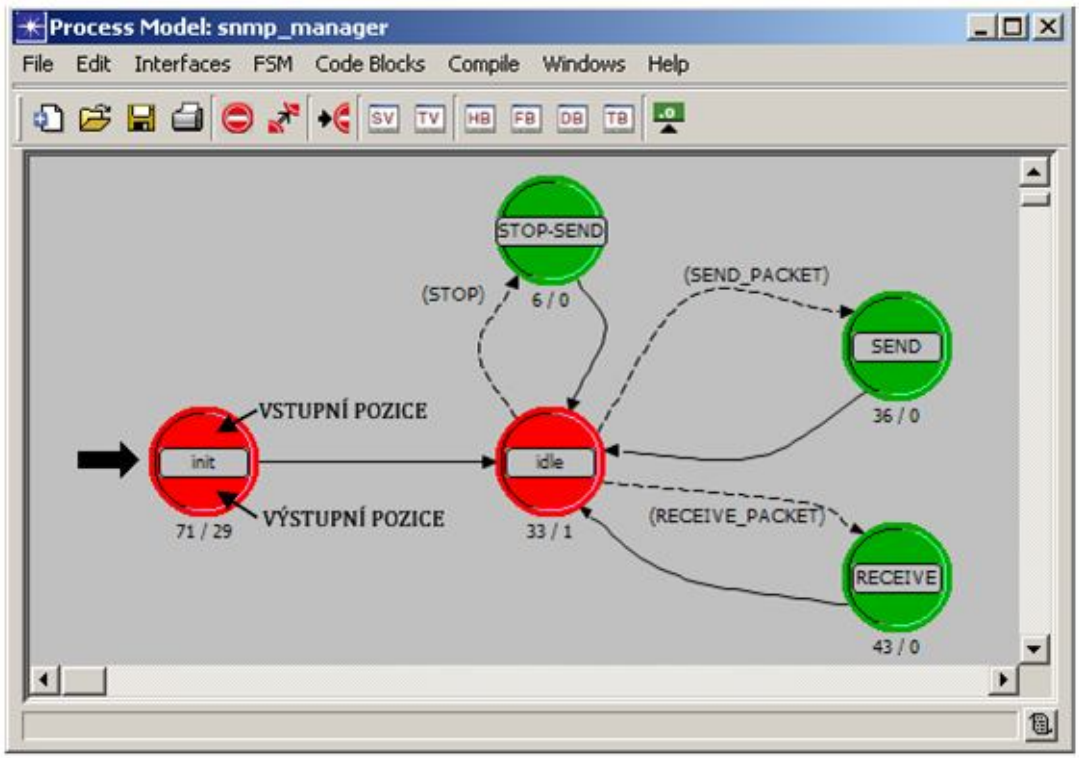
```
      int RequestId;            \\ID požadavku
      int ErrorStatus;          \\Identifikátor chyby
      int ErrorIndex;           \\Index chyby
      VARIABLEBIN *VariableBin; \\Struktura variablebin
} SNMPPDU;


/* SNMP paket*/
typedef struct snmppaket{
      int Version;              \\Verze SNMP paketu
      char *Community;          \\Komunita
      SNMPPDU *SNMPpdu;         \\Struktura PDU SNMP paketu
} SNMPPAKET;
```

Pozn.: bloky FB (Function Block), DB (Diagnostic Block) a TB (Termination Block) zůstanou prázdné.

6.1.4 Manažer – procesní model

Na Obr. 21 vidíme procesní model manažera. Stav INIT je výchozí stav (označený šipkou) a vykoná se hned při inicializaci modelu. Pomocí stavu INIT také zaregistrujeme jednotlivé proměnné a provedeme různá nastavení. Stav IDLE je stav, ve kterém model čeká na nějakou událost (např. příchozí paket nebo vypršení doby časovače), v tomto stavu se model může nacházet po ukončení určité události a nebo (jak je tomu v tomto případě) po inicializaci modelu.

Ze stavu IDLE pomocí podmínky SEND_PACKET vytvoříme přechod do stavu SEND a po vykonání kódu se proces vrátí do stavu IDLE. Tato vlastnost vyplývá z typu stavu, který je v tomto případě vynucený (zelený), tzn., že se vykoná kód, který obsahuje stav SEND a přejde do dalšího stavu. Dále ze stavu IDLE pomocí podmínky RECEIVE_PACKET proces přejde do stavu RECEIVE, kde přijmeme paket z přerušení (*op_intrpt_strm*). Dění v jednotlivých pozicích stavů popisují další kapitoly.



Obr. 21 Procesní model manažera

6.1.5 Manažer – vstupní pozice stavu INIT

Dvakrát klikneme na vstupní pozici stavu INIT (cesta jednotlivými úrovněmi modelu je zobrazena na Obr. 20, vstupní pozice stavu INIT na Obr. 21).Tato část kódu se vykoná hned při inicializaci modelu, přičemž simulátor odešle všem modelům přerušení typu *begsim intrp*. Toto přerušení musí být však správně nastaveno, více podrobností k tomuto nastavení je v práci [10] .

Pomocí funkce *op_id_self()* získáme hodnou vlastního identifikátoru (dále ID), který dále použijeme pro získání ID rodiče a to pomocí funkce *op_topo_parent()*. Dále potřebujeme zjistit ID UDP uzlu, k tomuto účelu použijeme funkci *op_id_from_name*. Následně alokujeme paměť pro chybové hlášky. V prostředí OPNET Modeler se paměť alokuje pomocí funkce *op_prg_mem_alloc()*. Nyní si načteme parametry z vlastností modelu a sice funkcí *op_ima_obj_attr_get()*.

Další část kódu se věnuje nastavení časů, tzn. čas startu odesílání paketů nesmí být později, něž je čas konce. Dále zda není konec nastaven na nekonečno (*MNGR_INFINITE_TIME*), pokud tomu tak je, nastaví se čas startu na nekonečno a bude vypsáno varovné hlášení *(op_prg_odb_print_major())*. Kompletní zdrojový kód je v příloze 8.1 stejně jako zdrojové kódy celého modelu.

6.1.6 Manažer – výstupní pozice stavu INIT

Ve výstupní pozici tohoto stavu jsou zapsány funkce pro instalaci rozhraní ICI a naplánování odesílání paketů. Nejprve prvním řádkem kódu vytvoříme ICI, které bude typu *udp_command_v3*. Další dva řádky budou, co se týká struktury zápisu, stejné. Pomocí funkce op_ici_attr_set nastavíme parametry ICI (*ici_ptr*), které potřebujeme pro registraci komunikačního portu UDP. Dále nainstalujeme ICI pomocí příkazu *op_ici_instal* (*ici_ptr*). Příkazem *op_intrp_force_remote()* [8] vyvoláme násilné přerušení poslané do uzlu UDP (*udp_obj_id*), které má za následek vytvoření komunikačního portu.

Dále získáme informace z ICI a uložíme do proměnné status. Pomocí této proměnné zjistíme, zda nedošlo k chybě při vytváření portu. Nakonec pomocí podmínky „ošetříme“ stav, kdy čas startu je nastaven na nekonečno, nebudou se odesílat žádné pakety. Když tomu tak není, manažer je připraven k odeslání paketů, které bude řízeno ve vstupní části stavu IDLE.

6.1.7 Manažer – vstupní pozice stavu IDLE

Zdrojový kód vstupní pozice stavu IDLE slouží k řízení zahájení a ukončení posílání paketů. Nejprve si pomocí podmínky zjistíme, zda se paket bude odesílat nebo ne. Pokud ano, tak musíme zkontrolovat, jestli je správně nastavena doba začátku a konce (konec odesílání nesmí být nastaven na nekonečno) odesílání paketů. Dále zjistíme, zda časový interval není menší než nula, tzn. záporný. Pokud tomu tak je, tak tento interval nastavíme na nějakou kladnou hodnotu.

Pokud je vše nastaveno v pořádku, tak pomocí funkce *op_intrpt_schedule_self()* naplánujeme přerušení. Parametry tohoto přerušení jsou čas simulace (ten zjistíme pomocí funkce *op_sim_time()*) plus interval (*next_intarr_time*) a typ přerušení, což v tomto případě je typu *MNGR_SEND*.

6.1.8 Manažer – výstupní pozice stavu IDLE

Výstupní pozice stavu IDLE obsahuje pouze uložení přerušení, typu OM, do proměnné *intrpt_code*. Toto přerušení vrací číselnou hodnotu označující stream, ze kterého přišlo.

6.1.9 Manažer – vstupní pozice stav STOP-SEND

Stav STOP-SEND je určen pro ukončení odesílání paketů, přičemž manažer je stále schopný pakety přijímat. Pomocí funkce *op_ev_valid()* zjistíme, zda je proměnná *next_send_evh* platná a pomocí funkce op_ev_cancel() zrušíme naplánované přerušení. Dále nastavíme *intrpt_code* na *MNGR_DISABLED_SEND*. Nakonec si můžeme v debuggeru [8] vypsat čas ukončení odesílání pomocí funkce *op_sim_time()*.

Pozn.: Výstupní pozice stavu STOP-SEND zůstane prázdná.

6.1.10 Manažer – vstupní pozice stavu SEND

Stav SEND slouží k odesílání paketů. Nejprve si tedy vytvoříme paket o velikosti 1024 bitů [11]. Pomocí funkce *op_prg_mem_alloc()* si dynamicky alokujeme strukturu paketu. V následujícím kroku tuto strukturu naplníme.

Naplněnou strukturu přiřadíme paketu (*send_paket*), který je deklarován v bloku TV, pomocí funkce *op_pk_fd_set()*. Parametry této funkce jsou ukazatel na paket, se kterým pracujeme (*send_paket*), index pole ve vytvořeném paketu (0), typ přiřazený poli (*OM OPC_FIELD_TYPE_STRUCT*), ukazatel na strukturu přiřazenou do paketu (*datapaket*), velikost pole v bitech (1024), funkce pro kopírování struktury (*op_prg_mem_copy_create*), uvolnění paměti (*op_prg_mem_free*) a aktuální velikost datové struktury v bytech (*sizeof (SNMPPAKET)*).

Nakonec vytvoříme nové ICI, které nastavíme a nainstalujeme. Nyní již můžeme paket (*send_paket*) pomocí funkce *op_pk_send()* poslat do streamu, který směřuje do cílového UDP uzlu.

Pozn.: Výstupní pozice stavu SEND zůstane prázdná.

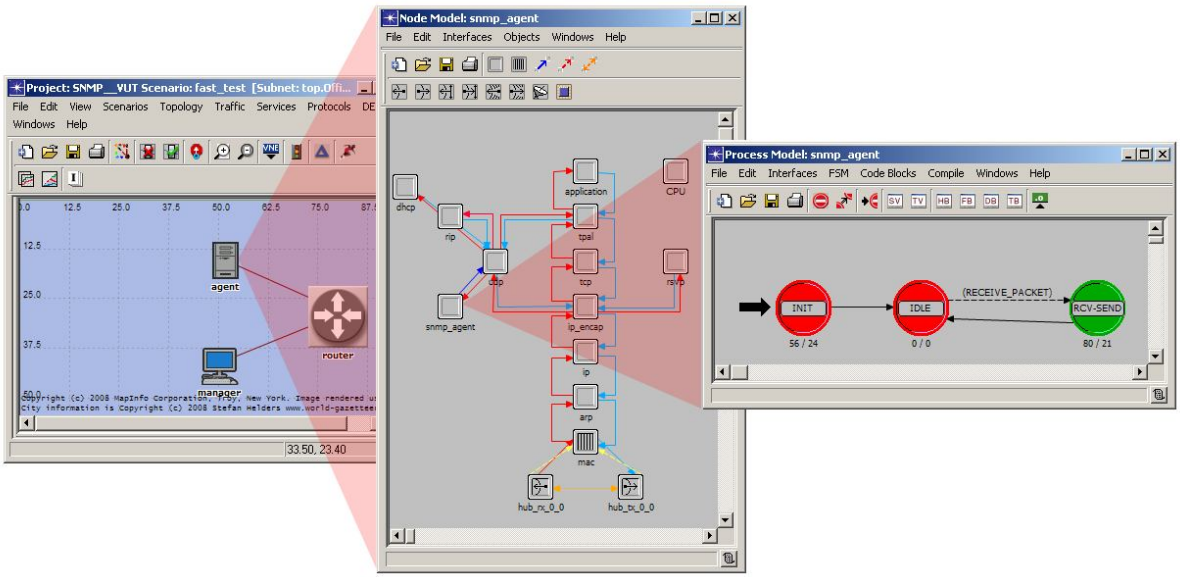
6.1.11 Manažer – vstupní pozice stavu RECEIVE

Tento stav řídí přijetí paketu. Nejprve získáme ICI z přerušení a uložíme do proměnné *ici_ptr*. Pomocí funkce *op_pk_get* získáme paket z příchozího streamu a uložíme do proměnné recv_paket. Nakonec paket (*recv_paket*) přijatý od agenta zničíme pomocí funkce *op_pkt_destroy()*.

Pozn.: výstupní pozice stavu RECEIVE zůstane prázdná.

6.2 Agent

V následujícím textu budou, stejně jako u manažera, popsány jednotlivé části editoru procesů agenta. Cesta jednotlivými úrovněmi modelu k tomuto editoru procesu názorně zobrazuje Obr. 22.



Obr. 22 Cesta úrovněmi modelu agenta k editoru procesu

6.2.1 Agent – blok SV

V editoru procesů (Process Model) klikneme opět na ikonu stavové proměnné SV (State Variables) a zobrazí se nám okno editoru. Zde klikneme na Edit ASCII, čímž se přepneme do textového módu. Blok SV modelu agenta obsahuje následující proměnné:

```
Objid \my_obj_id;          \\vlastní ID

Objid \parent_obj_id;      \\ID rodiče

Objid \udp_obj_id;         \\ID uzlu UDP

Char  \status;             \\proměnná status

char*  \error_string;      \\proměnná určena pro chybové hlášení

UdpT_Port    \loc_port;    \\místní port

UdpT_Port    \rem_port;    \\vzdálený port

/* Community string agenta - proměnná, do které budeme ukládat hodnoty vyčtené z
MIB */
char   \community_string_a;

/* ID číslo uzlu */
Objid  \ip_obj_id;

/* ID  atributu "Interface Information" */
Objid  \if_inf_attr_id;

/* ID atributu "IP host parameters" */
Objid \ip_host_attr_id;

Objid \app_obj_id;
MibTree*     \SnmpMibTree;
```

6.2.2 Agent – blok TV

V bloku dočasných proměnných si definujeme následující proměnné:

```
Packet*      pkptr;       \\ukazatel na paket typu OM
Ici*         ici_ptr;     \\vlastní ICI
```

```
/* Proměnné pro stav send-recieve */
IpT_Address remote_address;
int remote_port;

// Proměnná pro zpracování žádosti
MibTree* tmp;

/* Proměnné určené pro práce se strukturou paketu */
SNMPPAKET* datapaket;
char agentAddress[16];
```

6.2.3 Agent – blok HB

Jak již bylo zmíněno při popisu bloku HB manažera, do bloku HB vložíme externí knihovny a definici proměnných a struktur.

```
/* Vložení hlavičkových souborů */
#include <stdlib.h>
#include <udp_api.h>
#include <ip_addr_v4.h>
#include <string.h>
#include <memory.h>

/* Identifikace UDP streamu */
#define UDPSTRM 0

/* Definice přechodu */
#define RECEIVE_PACKET (op_intrpt_type() == OPC_INTRPT_STRM && \
                                        op_intrpt_code() == UDPSTRM)


/*Definice oddělovače pro dotazy na funkci QueryAttrGet() */
#define TOKEN "|"

/* Náhrada za nedefinované položky v diffServMib */
#define NOT_DEF "Není nastaveno"


/* Maximální délka OID */
const int MAX_OID_LENGTH = 1000;

char parent_obj_name[128];

/* Datové typy pro MIB tree */
typedef enum EDataTypes {TUnknown, TInteger, TString, TMibTree, TFunction}
DataTypes;
```

V další části zdrojového kódu je definována, stejně jako v bloku HB manažera, struktura *VARIABLEBIN*, *SNMPPDU* a *SNMPPAKET*. Dále si definujeme strukturu MibTree a definici jednotlivých funkcí, pro práci s MIB (funkce jsou okomentovány níže ve zdrojovém kódu).

```
/* MIB Tree */
typedef struct SMibTree{
       char *OID;
       void *Data;
       DataTypes Type;
       struct SMibTree *Next;
       struct SMibTree *Back;
} MibTree;
/* --- Definice funkcí --- */
```

```
// Získat data atributu objektu
void* QueryAttrGet(char *Query);

/*Definice MIB*/
/* Vytvoření struktury stromu MibTree */
MibTree* CreateStructure();

/* Čtení (prohledávání) struktury mib*/
MibTree* ReadStructure(MibTree *Tree, char *OID);

/* Zápis do MIB */
int WriteStructure(MibTree *Tree, char *OID, void *Value);

/* Vytvoření nové položky*/
int WriteStructureEx(MibTree *Tree, char *OID, void *Value, DataTypes Type);

/* Uvolnění MIB */
void FreeStructure(MibTree *Tree);
```

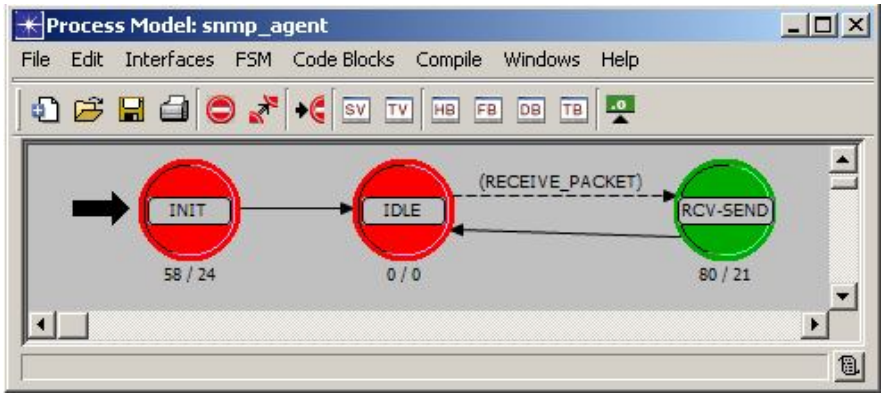
6.2.4 Agent – blok FB

V tomto bloku se zdrojový kód dá rozdělit na dvě části. V první části je vytvořena funkce pro získání hodnoty libovolného menu nebo jemu podřazenému menu (submenu) [11]. Této funkci je předán řetězec, ve kterém vyhledáváme pomocí tzv. rourové notace (znak „|"), přičemž využíváme proměnných, které jsou deklarovány v bloku SV a funkce (vstupní stav stavu INIT) popsané v kapitole 6.2.6.

Ve druhé části zdrojového kódu se nachází námi vytvořená struktura MIB v jazyce C [9]. Kompletní zdrojový kód najdeme ve vytvořeném modelu, který je obsažen v přiloženém CD.

6.2.5 Agent – procesní model

Výchozím stavem stavového automatu je stav INIT (viz model manažera). Ve stavu IDLE se nenachází žádný zdrojový kód, podmínky přechodu do dalších stavů jsou definovány v bloku HB. Obecně ve stavu IDLE model čeká na některý z typů přerušení. Stav RCV-SEND slouží nejprve k přijetí paketu pomocí přerušení ze streamu (*op_intrpt_strm*) a jeho zrušení. Následně ve výstupní pozici proběhne vytvoření nového paketu a jeho odeslání. Podrobnější vysvětlení jednotlivých stavů a jejich vstupních/výstupních pozic je uvedeno v následujících kapitolách.



Obr. 23 Procesní model agenta

6.2.6 Agent – vstupní pozice stavu INIT

Zdrojový kód vstupní pozice stavu INIT začíná stejně jako u manažera získáním ID vlastního, rodiče a uzlu UDP.

Pomocí funkce *op_ima_obj_attr_get_objid()* si zjistíme ID atributu Interface Information, který jsme si deklarovali v bloku SV a je popsán v podkapitole 6.2.1. Parametry této funkce

jsou ID uzlu, který chceme vyčíst, tzn. *Ip_obj_id*, jméno zanořeného (compound [11]) atributu a nakonec ukazatel na proměnnou, která bude naplněna ID tohoto zanořeného atributu.

Dále získáme hodnotu ID potomka ip host parameters (definován v bloku SV) a to pomocí funkce *op_topo_child().* Parametry této funkce jsou ID rodiče, datový typ potomka a index potomka.

Stejný postup aplikujeme i na atribut Interface Information. V dalším kroku vyčteme hodnotu atributu pomocí funkce *op_ima_obj_attr_get*, kde využíváme ID Interface Information *if_inf_attr_id*, definujeme jméno atributu „Address“ a ukazatel na proměnnou *agentAddress,* která je definována v bloku SV.

6.2.7 Agent – výstupní pozice stavu INIT

Zdrojový kód agenta (myšleno ve výstupní pozici stavu INIT) je stejný jako kód manažera s tou výjimkou, že ve zdrojovém kódu agenta není nastavení typu přerušení pro naplánování odesílání paketů.

6.2.8 Agent – vstupní pozice stavu RCV-SEND

V prvním kroku získáme ICI z přerušení pomocí funkce *op_intrpt_ ici()* a uložíme do *ici_ptr*. Dále zjistíme, zda ICI existuje, tzn. ici_ptr nemá nulovou hodnotu, pomocí „prázdného“ ukazatele (null pointer) *OPC_NIL*. Pokud zjistíme, že toto ICI existuje, tak pomocí funkce *op_ici_attr_get()* si zjistíme ip adresu a port odesílatele paketu a vyčteme paket ze streamu. Pak vyčteme strukturu datapaketu pomocí funkce *op_pkt_fd_get_ptr().* Dále následuje série příkazů a funkcí pro zpracování přijatého paketu.

Nakonec vstupní pozice stavu RCV-SEND, jak je to popsáno ve standardu protokolu SNMP [5], zahodíme přijatý paket.

6.2.9 Agent – výstupní pozice stavu RCV-SEND

Ve výstupní pozici stavu RCV-SEND budeme pokračovat tak, že pro odpověď agenta vytvoříme prázdný paket o velikosti 1024 bitů a do jeho pole vložíme strukturu SNMPPAKET pomocí funkce *op_pk_fd_set().* Dále pro správné odeslání nastavíme ICI, přičemž potřebujeme znát lokální port agenta, IP adresu manažera a port manažera. Pak nainstalujeme ICI a pomocí funkce *op_pkt_send()* paket odešleme do streamu, který směřuje k UDP.

Závěr

Tato práce je součástí rozsáhlejšího projektu, kde mou úlohou byl návrh modelu databáze MIB v jazyce ASN.1, její převod do jazyka C a implementace v simulačním prostředí programu OPNET Modeler.

Po teoretickém rozboru protokolu SNMP a databáze MIB jsem navrhl zjednodušenou strukturu databáze MIB v jazyce ASN.1. Tato struktura se všemi objekty, které obsahuje, je podrobně popsána v kapitole 4.

Prvotním cílem byl převod této struktury v jazyce ASN.1 do jazyka C. K tomuto účelu jsem využil dva volně dostupné kompilátory. Prvním kompilátor, ASN1C compiler, byl od společnosti Objective System, která nám poskytla studentskou licenci zdarma. Druhým kompilátorem byl volně šiřitelný program DMH SMIv2 MIB-Compiler. Bohužel jsme nemohli využít ani jednoho zdrojového kódu v jazyce C, které byly vygenerovány z těchto dvou kompilátorů. Důvodem bylo velké množství knihoven, které byly k správnému chodu programu v jazyce C vyžadovány. Z tohoto důvodu byla nemožná implementace do prostředí programu OPNET Modeler.

Proto jsme přistoupili na návrh databáze MIB přímo v jazyce C. Zdrojový kód této databáze MIB je podrobně popsán v práci [8]. V dalším kroku byla databáze MIB implemetována do simulačního prostředí OPNET Modeler.

Další část práce je věnována popisu jednotlivých bloků a stavů vytvořeného modelu v programu OPNET Modeler, zejména na komunikaci agenta a manažera.

Simulační model manažeru vygeneruje paket a naplní jej daty. Tento paket je následně odeslán agentovi, který jej příjme na UDP vrstvě. Agent si vyčte strukturu paketu a paket zahodí. V dalším kroku agent vygeneruje nový paket a odešle ho jako odpověď manažerovi.

Příslušný stavový automat a jednotlivé bloky modelu jsou popsány tak, aby jej bylo možné opět později sestavit dalšími zájemci, kteří budou pokračovat v tomto rozsáhlém projektu.

Literatura

[1] MAURO, D., SCHMIDT, Kevin. *Essential SNMP*. 2nd edition. [s.l.] : O'Reilly, 2005. 460 s. ISBN 0-596-00840-6.

[2] KMENT, V. *ASN.1 – koncept abstraktní syntaxové notace* [online]. 2005 [cit. 2007-12-16]. Dostupný z WWW: <http://www.lupa.cz/clanky/asn-1-8211-koncept-abstraktni-syntaxove-notace/>.

[3] DELCROIX, M., LUMETTA, Olivier. *Lessons about SNMP* [online]. Poznan, Poland : Faculty of Electronics and Telecommunications, 2005 [cit. 2007-12-16]. Dostupný z WWW: <http://www.et.put.poznan.pl/snmp/main/mainmenu.html>.

[4] MARK A., M. *Managing Internetworks with SNMP*. 2nd edition. [s.l.] : IDG Books Worldwide, Inc., 1997. 699 s.

[5] MCCLOGHRIE, K., KASTENHOLZ, F. *RFC 2233 - The Interfaces Group MIB using SMIv2* [online]. 1997 [cit. 2007-12-16]. Dostupný z WWW: <http://www.faqs.org/rfcs/rfc2233.html>.

[6] KLAŠKA, L. *SNMP objekty a MIB* [online]. 2000 [cit. 2007-12-11]. Dostupný z WWW: <http://www.svetsiti.cz/view.asp?rubrika=Tutorialy&temaID=23&clanekID=31>.

[7] BAKER F., CHAN K., SMITH A., *Management Information Base for the Differentiated Services Architecture - RFC3289*, [online], Internet Engineering Task Force, [cit. 2007-12-16] Dostupné z WWW: <http://www3.ietf.org/proceedings/01dec/I-D/draft-ietf-diffserv-mib-16.txt>

[8] VLČEK, M., *Implementace a optimalizace universálního aplikačního protokolu v simulačním prostředí Opnet Modeler*. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2008. Vedoucí bakalářské práce Ing. Karol Molnár, Ph.D.

[9] MACURA, M. *Řídící protokoly používané pro správu sítě a zajištění QoS*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2008. Vedoucí bakalářské práce Ing. Karol Molnár, Ph.D.

[10] BEDNÁRIK, J. *Modelování komunikace proprietárním protokolem, určeným pro výměnu informací o podporované technologii QoS, v prostředí Opnet Modeler*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2007. Vedoucí bakalářské práce Ing. Karol Molnár, Ph.D.

[11] ZEMAN, O. *Implementace simulačního modelu zjednodušené databáze Diffserv-MIB*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2008. Vedoucí bakalářské práce Ing. Karol Molnár, Ph.D.

.

7 Přílohy

7.1 Manažer – vstupní pozice stavu INIT

```
/* Získaní vlastního ID */
my_obj_id = op_id_self();

/* Získaní ID rodiče */
parent_obj_id = op_topo_parent (my_obj_id); //ID modelu

/* Získaní ID modulu UDP */
udp_obj_id = op_id_from_name (parent_obj_id, OPC_OBJTYPE_PROC, "udp");

/* Načteni názvu modelu */
op_ima_obj_attr_get_str (parent_obj_id, "name", 128, parent_obj_name);

/* Alokoace paměti pro chybové hlášky */
error_string = (char*) op_prg_mem_alloc (sizeof(char[256]));

/* Vyčtení hodnoty paketu */
op_ima_obj_attr_get (my_obj_id, "Local port", &loc_port);
op_ima_obj_attr_get (my_obj_id, "Agent Address", &rem_addr_string);
op_ima_obj_attr_get (my_obj_id, "Agent port", &rem_port);
op_ima_obj_attr_get (my_obj_id, "Packet Interarrival Time", &next_intarr_time);
op_ima_obj_attr_get (my_obj_id, "Start Time", &start_time);
op_ima_obj_attr_get (my_obj_id, "Stop Time", &stop_time);
op_ima_obj_attr_get (my_obj_id, "Community String", &community_string_m);

/* Převod IP adresy na řetězec */
rem_addr = ip_address_create (rem_addr_string);

/* Zde se ujistíme zda konec odesílání paketů není dříve než začátek */
if ((stop_time <= start_time) && (stop_time != MNGR_INFINITE_TIME))
      {
      /* Konec odesílání paketů je dříve než začátek */
      start_time = MNGR_INFINITE_TIME;

      /* Zobrazení varování */
      op_prg_odb_print_major ("Warning from SNMP manager model:",
                              "Although the generator is not disabled (start time
                               is set to a finite value),","a stop time that is not
later than the start time is specified.","Disabling the generator.", OPC_NIL);
      }

/* Výpis v debuggeru */
printf ("\n%s: Manager local port je: %d\n",parent_obj_name, loc_port);
printf ("%s: Agent remote port je: %d\n",parent_obj_name, rem_port);
printf ("%s: Agent remote address je: %s\n",parent_obj_name, rem_addr_string);
printf ("%s: Agent remote address v int je: %d\n",parent_obj_name, rem_addr);
printf ("%s: Interval time je: %d\n",parent_obj_name, next_intarr_time);
printf ("%s: Start time je: %f\n",parent_obj_name, start_time);
printf ("%s: Stop time je: %f\n\n",parent_obj_name, stop_time);

/* Pauza pre dokonceni inicializace UDP */
 op_intrpt_schedule_self (op_sim_time (), 0);
```

7.2 Manažer – výstupní pozice stavu INIT

```
/* Získání informací z ICI a registrace nového portu */
ici_ptr = op_ici_create ("udp_command_v3");
op_ici_attr_set (ici_ptr, "local_port", loc_port);
op_ici_attr_set (ici_ptr, "inet_support", 1);
op_ici_install (ici_ptr);

/* Registrace portu na UDP vrstvě */
op_intrpt_force_remote (UDPC_COMMAND_CREATE_PORT, udp_obj_id);

/* Kontrola, zdy byl port uspěšně zaregistrován na UDP vrstvě */
```

```
op_ici_attr_get (ici_ptr, "status", &status);
if (status != UDPC_IND_SUCCESS)
      {
            sprintf (error_string, "%d jako odpoved na CREATE_PORT ", status);
            printf ("%s: Chyba registrace portu! %s", parent_obj_name,
error_string);
            op_sim_end ("SNMP Manager node: Nezaregistr. port UDP!", error_string,
"", ""); // Konec simulace
      }
else printf("%s: Lokalni port (%d) uspesne zaregistrovan.\n", parent_obj_name,
loc_port);


/* Ošetření stavu, kdy čas startu odesílaní paketů je nastaven na nekonečno, nebudou
se odesílat žádné pakety. Když tomu tak není, manažer je připraven k odeslání
paketů */

if (start_time == MNGR_INFINITE_TIME)
      {
       intrpt_code = MNGR_DISABLED_SEND;
      }
else intrpt_code = MNGR_START_SEND;
```

7.3 Manažer – vstupní pozice stavu IDLE

```
if (intrpt_code == MNGR_START_SEND || intrpt_code == MNGR_SEND)
      {
      printf ("\n %s: Vse OK.Bude se generovat \n", parent_obj_name);
      /* Nastavíme začátek a konec odesílání paketů, pokud se tak již nestalo */
      if (intrpt_code == MNGR_START_SEND)
            {
            printf ("\n %s:Nastaveni preruseni startu a konce\n",parent_obj_name);
            op_intrpt_schedule_self (start_time, MNGR_SEND);

            if (stop_time != MNGR_INFINITE_TIME)
                  {
                  op_intrpt_schedule_self (stop_time, MNGR_STOP_SEND);
                  }
/* Zjistíme, zda časový interval není záporný. Pokud tomu tak je, tak tento
interval nastavíme na 30s */
            if (next_intarr_time <0)
                  {
                  next_intarr_time = 30.0;
                  }
            }

      else next_send_evh = op_intrpt_schedule_self (op_sim_time () +
next_intarr_time, MNGR_SEND);
      }
```

7.4 Manažer – výstupní pozice stavu IDLE

```
intrpt_code = op_intrpt_code();
```

7.5 Manažer – vstupní pozice satvu STOP-SEND

```
if (op_ev_valid (next_send_evh) == OPC_TRUE)
      {
      op_ev_cancel (next_send_evh);
      intrpt_code = MNGR_DISABLED_SEND;
      printf("\n %s: Questions were closed %f\n",parent_obj_name,op_sim_time());
      }
```

7.6 Manažer – vstupní pozice stavu SEND

```
// Generate new packet
send_paket = op_pk_create (1024);
```

```
/* Alokace paměti pro novou datovou strukturu */
datapaket = (SNMPPAKET *) op_prg_mem_alloc (sizeof (SNMPPAKET));
datapaket->SNMPpdu = (SNMPPDU *) op_prg_mem_alloc (sizeof (SNMPPDU));
datapaket->SNMPpdu->VariableBin   =   (VARIABLEBIN   *)   op_prg_mem_alloc   (sizeof
(VARIABLEBIN));
datapaket->Community = (char *) op_prg_mem_alloc (strlen(&community_string_m));
datapaket->SNMPpdu->VariableBin->ObjectV     =     (char     *)     op_prg_mem_alloc
(strlen(".1.3.6.1.2.1.97.1.2.6.1.3"));
datapaket->SNMPpdu->VariableBin->ValueV = (char *) op_prg_mem_alloc (strlen(" "));

/* Naplnění struktury hodnotami */
datapaket->Version=1;
datapaket->Community = &community_string_m;
datapaket->SNMPpdu->PDUtype=21;
datapaket->SNMPpdu->RequestId=22;
datapaket->SNMPpdu->ErrorStatus=33;
datapaket->SNMPpdu->ErrorIndex=44;
sprintf (datapaket->SNMPpdu->VariableBin->ObjectV, ".1.3.6.1.2.1.97.1.2.6.1.3");
sprintf (datapaket->SNMPpdu->VariableBin->ValueV, " ");

/* Přiřazení datové struktury poli v paketu */
op_pk_fd_set(send_paket,0,OPC_FIELD_TYPE_STRUCT,datapaket,1024,
op_prg_mem_copy_create, op_prg_mem_free, sizeof (SNMPPAKET));
/* Vytvoření nového ICI */
ici_ptr = op_ici_create ("udp_command_v3");

/* Nastavení hodnot ICI */
op_ici_attr_set (ici_ptr, "local_port", loc_port); // Manager port
op_ici_attr_set (ici_ptr, "rem_addr", rem_addr); // Agent IP adresa
op_ici_attr_set (ici_ptr, "rem_port", rem_port); // Agent port

/* Instalace ICI */
op_ici_install (ici_ptr);

/* Odeslání paketu */
op_pk_send (send_paket, UDPSTRM);
```

7.7 Manažer – vstupní pozice stavu RECEIVE

```
// Získaní ICI z přerušeni
ici_ptr = op_intrpt_ici();

// Načítaní paketu
recv_paket = op_pk_get (op_intrpt_strm () );

printf("%s: Receive packet from agent\n", parent_obj_name);

// Zničení paketu
op_pk_destroy (recv_paket);
```

7.8 Agent – vstupní pozice stavu INIT

```
// Získaní vlastního ID
my_obj_id = op_id_self();

// Získaní rodičovského ID
parent_obj_id = op_topo_parent(my_obj_id);

// Získaní UDP ID
udp_obj_id = op_id_from_name (parent_obj_id, OPC_OBJTYPE_PROC, "udp");

/* Získání ip uzlu id */
ip_obj_id = op_id_from_name (parent_obj_id, OPC_OBJTYPE_PROC, "ip");

/* Načtení názvu modelu */
op_ima_obj_attr_get_str (parent_obj_id, "name", 128, parent_obj_name);

/* Získání IP adresy z atributu IP uzlu */
op_ima_obj_attr_get_objid (ip_obj_id, "ip host parameters", &ip_host_attr_id);
```

```
ip_host_attr_id = op_topo_child (ip_host_attr_id, OPC_OBJTYPE_GENERIC, 0);
op_ima_obj_attr_get_objid(ip_host_attr_id,"InterfaceInformation", &if_inf_attr_id);
if_inf_attr_id = op_topo_child (if_inf_attr_id, OPC_OBJTYPE_GENERIC, 0);
op_ima_obj_attr_get (if_inf_attr_id, "Address", &agentAddress);
printf ("%s : IP Adresa: %s\n",parent_obj_name, agentAddress);

error_string = (char*) op_prg_mem_alloc (sizeof(char[128]));

/* Vyčtení atributů modelu */
op_ima_obj_attr_get (my_obj_id, "Local port", &loc_port);
op_ima_obj_attr_get (my_obj_id, "Community string", &community_string_a);

printf("%s: Local port vycteny z nastaveni je: %d\n",parent_obj_name, loc_port);

SnmpMibTree = CreateStructure();
printf ("%s : MIB Strom vygenerován\n",parent_obj_name);

/*Pauza pro dokončeni inicializace UDP a vyvolaní přerušeni pro registraci portu*/
op_intrpt_schedule_self (op_sim_time (), 0);
```

7.9 Agent – výstupní pozice stavu INIT

```
/* Získání informací z ICI a registrace nového portu */
ici_ptr = op_ici_create("udp_command_v3");
op_ici_attr_set (ici_ptr, "local_port", loc_port);
op_ici_attr_set (ici_ptr, "inet_support", 1);  // Podpora IPv4 formatu
op_ici_install (ici_ptr);

op_intrpt_force_remote(UDPC_COMMAND_CREATE_PORT, udp_obj_id);

/* Kontrola jestli bylo vytvořene spojení */
op_ici_attr_get (ici_ptr, "status", &status);


if (status != UDPC_IND_SUCCESS)
{
      sprintf(error_string, "%d jako odpoved na CREATE_PORT ", status);
      printf ("%s: Chyba! aplikace zachytila chybu %s",parent_obj_name,
               error_string);
      op_sim_end ("Chyba: aplikace zachytila chybu", error_string, "", "");
}
else printf("%s:Lokalni port ( %d ) uspesne
zaregistrovan.\n",parent_obj_name,loc_port);
```

7.10 Agent – vstupní pozice stavu RCV-SEND

```
/* Získání ICI */
ici_ptr = op_intrpt_ici();

/* Zjistíme zda ICI existuje, jestli ano, vyčteme data */
if (ici_ptr != OPC_NIL )
      {
            op_ici_attr_get (ici_ptr, "rem_port", &remote_port);
            op_ici_attr_get (ici_ptr, "rem_addr", &remote_address);
      }

// Vyčtení paketu ze streamu
pkptr = op_pk_get (op_intrpt_strm() );

printf ("%d. %s: SNMP paket dorazil na server\n", counter++, parent_obj_name);

/* Získání datové struktury pole paketu */
op_pk_fd_get_ptr (pkptr, 0, (void**)&datapaket);
printf ("***** %s: Content packet ***** \n", parent_obj_name);
printf ("Packet Version: %d\n", datapaket->Version);
printf ("Packet Community: %s\n", datapaket->Community);
printf ("Packet PDUtype: %d\n", datapaket->SNMPpdu->PDUtype);
printf ("Packet RequestId: %d\n", datapaket->SNMPpdu->RequestId);
```

```
printf ("Packet ErrorStatus: %d\n", datapaket->SNMPpdu->ErrorStatus);
printf ("Packet ErrorIndex: %d\n", datapaket->SNMPpdu->ErrorIndex);
printf ("Packet ObjectV: %s\n", datapaket->SNMPpdu->VariableBin->ObjectV);
printf ("Packet ValueV: %s\n", datapaket->SNMPpdu->VariableBin->ValueV);
printf ("***** END Content packet ***** \n");

if (datapaket->SNMPpdu->PDUtype == 11) printf ("Vycteny parametr zadany v dotazu:
%s \n", (char *) QueryAttrGet(datapaket->SNMPpdu->VariableBin->ObjectV));
      else {
            tmp   =   ReadStructure(SnmpMibTree,   datapaket->SNMPpdu->VariableBin-
            >ObjectV);
            if (tmp->Type==TString) {
                  datapaket->SNMPpdu->VariableBin->ValueV = ((char*)tmp->Data);
                  printf ("Vycteny parametr zadany v dotazu: %s \n", ((char*)
                  tmp->Data));
                  }
            }

if (strcmp (datapaket->Community, &community_string_a) == 0) {
printf ("%s: Community managers and agents are equal!!!\n", parent_obj_name);
}
if (strcmp (datapaket->Community, &community_string_a) != 0) {
printf ("%s: Community managers and agents are various!!!\n", parent_obj_name);
}
/* Zničení přijatého paketu */
op_pk_destroy (pkptr);
```

7.11 Agent – výstupní pozice stavu RCV-SEND

```
/* Generování nového paketu */
pkptr= op_pk_create (1024);
op_pk_fd_set     (pkptr,     0,     OPC_FIELD_TYPE_STRUCT,     datapaket,     1024,
op_prg_mem_copy_create, op_prg_mem_free, sizeof (SNMPPAKET));

/* Nastavení hodnoty ICI */
op_ici_attr_set (ici_ptr, "local_port", loc_port); // Manager port
op_ici_attr_set (ici_ptr, "rem_addr", remote_address); // Agent IP adresa
op_ici_attr_set (ici_ptr, "rem_port", remote_port); // Agent port

/* Instalace ICI */
op_ici_install (ici_ptr);

/* Odeslání packtu */
op_pk_send (pkptr, UDPSTRM);
```

Obsah CD

Přiložený disk obsahuje následující složky:

- Bakalářská práce – elektronická verze bakalářské práce,
- MIB - soubory se zdrojovými kódy vytvořené MIB v jazyce ASN. 1,
- OPNET Modeler – funkční model vytvořený v OM,
- Metadata – metadata v elektronické podobě.