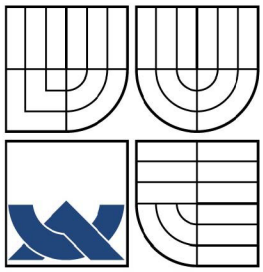
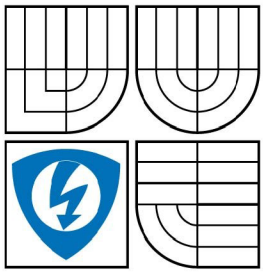


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKACNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

POKROČILÉ MODELOVÁNÍ CHOVÁNÍ PROTOKOLU SNMP V SIMULAČNÍM PROSTŘEDÍ OPNET MODELER

ADVANCED MODELLING OF SNMP PROTOCOL IN OPNET MODELER SIMULATION ENVIRONMENT

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE: Bc. JOSEF BEZCHLEBA
AUTHOR

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

BRNO 2008

LICENČNÍ SMLOUVA
POSKYTOVANÁ K VÝKONU PRÁVA UŽÍT ŠKOLNÍ DÍLO

uzavřená mezi smluvními stranami:

1. Pan/paní

Jméno a příjmení: Josef Bezchleba

Bytem: Těmice 267, 696 84, Těmice u Hodonína

Narozen/a (datum a místo): 5.10. 1982, Kyjov

(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

(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:	Pokročilé modelování chování protokolu SNMP v simulačním prostředí Opnet Modeler
Vedoucí/ školitel VŠKP:	Ing. Karol Molnár, Ph.D.
Ústav:	Ústav telekomunikací
Datum obhajoby VŠKP:	10. a 11.6 2008

VŠKP odevzdal autor nabyvateli v*:

- □ tištěné formě – počet exemplářů 2
- □ elektronické formě – počet exemplářů 2

* 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

- 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.
- Licence je poskytována celosvětově, pro celou dobu trvání autorských a majetkových práv k dílu.
- 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)
- 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

- Smlouva je sepsána ve třech vyhotoveních s platností originálu, přičemž po jednom vyhotovení obdrží autor a nabyvatel, další vyhotovení je vloženo do VŠKP.
- 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.
- 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.
- Licenční smlouva nabývá platnosti a účinnosti dnem jejího podpisu oběma smluvními stranami.

V Brně dne: ……

…….. ……

Nabyvatel Autor

ANOTACE

Cílem diplomové práce je seznámit se s protokolem pro vzdálenou správu zařízení SNMP a jeho implementací do simulačního prostředí Opnet Modeler. Pro implementaci byla vybrána databáze DiffServMIB, která popisuje komponenty technologie rozlišovaných služeb. Diplomová práce obsahuje podrobný popis možností využití databáze DiffServMIB pro řízení zpracování provozu a návrh modelu, který popisuje diferencované zpracování dat ve vstupním a výstupním směru na rozhraní směrovače. Tento model jsem následně implementoval do simulačního prostředí Opnet Modeler. Cílem takto vzniklého simulačního modelu je snadná konfigurovatelnost jednotlivých atributů databáze MIB. Ve své práci dále navrhuji způsob implementace, kterým bude provázána databáze MIB s konfigurací aktivního prvku se zaměřením na výstupní procedury mechanismu DiffServ.

Klíčová slova:

kvalita služeb, rozlišované služby, SNMP, struktura MIB, Opnet Modeler, databáze DiffServMIB, model DiffServ

ABSTRACT

The aim of my diploma is to introduce the SNMP protocol designed for remote device management to the readers and present its implementation into the Opnet Modeler simulation environment. For the implementation the DiffServMIB database has been chosen, which describes the components of differentiated services mechanism. My diploma thesis describes in detail the possibilities of application of the DiffServMIB database in traffic management. The design of a simulation model, which represents the differentiated data processing in input and output direction on a router interface, is also introduced. Next, this model is implemented into the Opnet Modeler simulation environment. The goal of this simulation model is the possibility of simple configuration of individual MIB attributes. In my work I also suggest an implementation method able to interconnect the MIB database with the configuration process of the active network element - with focus on output procedures of DiffServ mechanism.

Keywords:

quality od services, differentiated services, SNMP, structure of MIB, Opnet Modeler, database DiffServMIB, model DiffServ

PROHLÁŠENÍ

Prohlašuji, že diplomovou práci na téma „Pokročilé modelování chování protokolu SNMP v simulačním prostředí Opnet Modeler“ jsem vypracoval samostatně pod vedením vedoucího diplomové 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 uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce 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 diplomové práce Ing. K. Molnárovi, Ph.D. za velmi užitečnou metodickou pomoc a cenné rady při zpracování diplomové práce, jakožto i za jeho vstřícnost a laskavost.

V Brně dne..... .....
podpis autora

Abecední seznam použitých zkratek:

AF – (Assured Forwarding) zajištěné přesměrování

BA – (Behavior Aggregate) sdružené zacházení

CBS – (Committed Burst Size) velikost garantovaného shluku

CIR – (Committed Information Rate) garantovaná přenosová rychlost

DiffServ – (Differentiated Services) rozlišené služby

DSCP – (Differentiated Service Codepoint) značka rozlišované služby

EBS – (Excess Burst Size) velikost nadměrného shluku

EF – (Expedited Forwarding) urychlené přesměrování

FIFO – (First In First Out) paměť typu první zařazen, první ven

IP – (Internet Protocol) komunikační protokol Internetu

ISO – (International Standard Organization) mezinárodní standardizační organizace

MF – (Multi-field) více-položková klasifikace

MIB – (Management Information Base) databáze managementu informací jednoho koncového systému

NMS – (Network Management System) systém správy a dohledu nad síťovými prvky

OID – (Object Identifier) identifikátor objektu v MIB

PBS – (Peak Busrt Size) velikost maximálního shluku

PIR – (Peak Information Rate) maximální přenosová rychlost

SMI – (Structure of Management Information) pravidla způsobu orientace ve struktuře MIB a způsoba manipulace s objekty v MIB

SNMP – (Simple Network Management Protocol) jednoduchý protokol řízení sítě

srTCM – (Single Rate Three Color Marker) mechanismus měření jedna rychlost – 3 barvy

TCP – (Transmission Control Protocol) protokol řízení přenosu

TCB – (Transmission Control Block) blok úpravy provozu

trTCM – (Two Rate Three Color Marker) mechanismus měření dvě rychlosti – 3 barvy

WRR – (Weighted Round Robin) vážená cyklická obsluha fronty

Seznam obrázků

Seznam tabulek

Obsah

1 Simple Network Management Protocol

Simple Network Management Protocol (SNMP), čili v překladu jednoduchý protokol pro správu sítě je aplikační protokol, který nabízí služby správy síťových prvků nad protokolovým zásobníkem TCP/IP. SNMP je založen na modelu klient/server. V současné době existuje vlastní specifikace standardu SNMPv1 a návrhy jeho rozšíření - SNMPv2 a SNMPv3. Důvodem pro návrh nových verzí protokolu SNMP byla hlavně nedokonalost první verze z hlediska bezpečnosti.

SNMP komunikace je založena na modelu manager - agent a umožňuje přenos a komunikaci mezi správcem sítě - managerem a agenty na jednotlivých síťových zařízeních. K tomu vyžaduje, aby každé zařízení sítě poskytovalo jisté základní informace o sobě samém a stejně tak usnadnilo přidání dalších informací, specifických pro dané zařízení.

1.1 SNMP Manager

SNMP manager je program, který běží na síťové stanici. U větších systémů jde většinou o vyhrazenou výkonnou pracovní stanici s běžícím software Network Management systém (NMS). Funkce tohoto SNMP Manageru pak spočívá v dotazování jednotlivých SNMP Agentů pomocí SNMP operací.

Smyslem je získat všechny potřebné informace o daném zařízení, které agent reprezentuje. SNMP Manager poskytuje většinou grafické rozhraní, které umožňuje prezentaci získaných dat, sledování síťových alarmů a archivaci dat (např. k analýze časového vývoje).

1.2 SNMP Agent

SNMP agent je malý program, běžící na síťovém zařízení a odpovídá na dotazy SNMP Managera. Agent proto neustále monitoruje a sbírá informace o všech dostupných funkcích a stavech daného zařízení. SNMP agenti neposkytují žádný grafický interface. Slouží jen pro sběr a přenos informací.

V některých případech mohou být informace také vyslány agentem bez vyžádání managerem. Jestliže agent detekuje jisté události, jako např. hardwarovou poruchu, vyšle tuto informaci, zvanou trap, sám bez vyžádání. To je důležité pro zajištění okamžité informovanosti např. o vážnějších problémech, protože dotazování se provádí v jistých intervalech.

1.3 SNMP operace

SNMP je asynchronní protokol typu požadavek/odpověď. Slovo simple pochází z toho faktu, že protokol má jen 5 funkcí.

1.3.1 GetRequest

GetRequest reprezentuje žádost o informaci, kterou posílá manager agentovi k získání informace o stavu nebo hodnotě jistého objektu. Jde vlastně o příkaz čtení. V rámci jednoho příkazu je možné žádat informace o více objektech. To redukuje nutnou komunikaci mezi zařízeními.

1.3.2 GetNextRequest

GetRequest reprezentuje žádost o hodnotu objektu stojícího v hierarchickém systému bezprostředně za naposledy dotazovaným objektem. Protože informace jsou organizovány hierarchicky, jde o žádost o informaci na další, nižší vrstvě MIB struktury.

1.3.3 GetResponse

Tento příkaz je vyslán agentem jako odpověď na příkaz GetRequest, kterým vrací vyžádanou informaci.

1.3.4 SetRequest

Příkaz SetRequest nastavuje hodnotu proměnné v MIB agenta. Jde vlastně o příkaz pro zápis. Ne všichni výrobci SNMP zařízení jej umožňují. Pak ale uživatelé nemohou ve skutečnosti provádět plnohodnotnou správu zařízení ale pouze jeho monitorování.

1.3.5 Trap

Tento příkaz je vyslán agentem managerovi jako oznámení nějaké významné události. Na rozdíl od předchozích příkazů, není očekávaná odpověď.

2 Management Information Base

Management Information Base (MIB) popisuje sadu objektů, které jsou předmětem správy. MIB není skutečnou databází, pouze definuje databázové vlastnosti pro uložení a práci s daty. Spravované zařízení může implementovat jednu, nebo více MIB, v závislosti na funkci. Tyto MIB databáze jsou velmi podobné standardním databázím v tom smyslu, že popisují jak strukturu, tak formát dat. MIB jsou napsány podle pravidel Structure of Management Information (SMI), specifikovaných v RFC2578 [1] a určujících způsob orientace ve struktuře MIB a způsob manipulace s objekty v MIB. V současnosti již existuje i návrh (draft) standardu SMIv2, zpětně kompatibilního s předchozí verzí.

2.1 Struktura MIB

Každý SNMP objekt, reprezentující určitý parametr zařízení musí mít jedinečné jméno, aby se na něj dalo odkazovat při SNMP operacích. Protože jedno zařízení může obsahovat objekty, definované nezávisle několika různými výrobci, schéma pro pojmenování těchto objektů muselo být navrženo tak, aby nemohlo dojít k záměně. Proto byla zvolena koncepce hierarchického stromu SNMP Global Naming Tree, vyvinutého organizací International Standardization Organization (ISO).

Standardní struktura MIB tedy odpovídá tomuto systému podle SNMP Global Naming Tree, který se skládá z objektů kořen, větev a list. Každá část tohoto stromu má označení, které se skládá ze dvou částí – stručného textového popisu a z celého čísla.

Jednotlivým výrobcům zařízení jsou přidělovány větve – jsou jmenováni jeho výkonnými autoritami - a mohou si tak vytvářet do šířky a hloubky neomezenou vlastní strukturu. Takto vzniklé privátní (proprietary) MIB popisují i specifické vlastnosti konkrétního zařízení. Většinou jsou ale výrobci zveřejňovány, právě z důvodu umožnění správy těchto prvků i aplikacemi jiných výrobců.

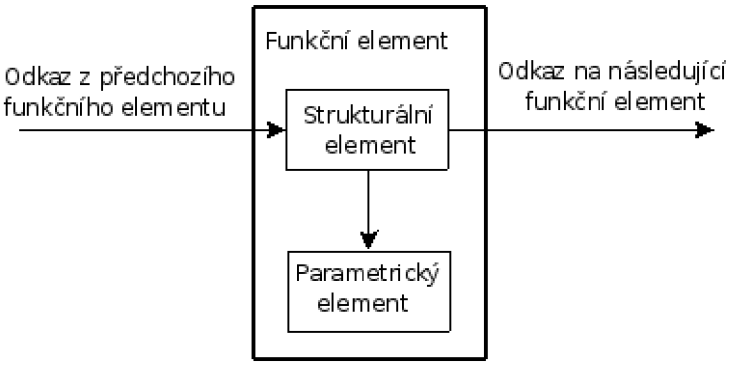
Jméno uzlu, OBJECT IDENTIFIER (OID) je tak tvořeno sekvencí těchto celých čísel od kořenového uzlu přes větev až k danému objektu typu list. Tato decimální notace reprezentuje tedy cestu ke každé z funkcí nebo schopností daného zařízení. Jde o podobný systém jako při specifikacích plných cest k souborům v systémech UNIX a DOS, přičemž nejvyšší úroveň začíná v objektu (root). Textový popis slouží jen k naší snadnější orientaci v této struktuře.

3 Popis DiffServMIB

V dokumentu RFC3289 [2] je popsána MIB pro zařízení, která implementují mechanismus diferencovaných služeb. DiffServMIB databáze definuje objekty, které jsou nezbytné k ovládání příslušných funkčních bloků v zařízení. Tyto bloky pak tvoří hierarchický systém, kde jsou jednotlivé funkční bloky spojované do bloků úpravy provozu TCB. Mechanismus diferencovaných služeb nemá striktní požadavky na to, jak má být výsledný model sestaven. Jednotlivé komponenty mohou být sestaveny různými způsoby a mohou být použity k monitorování, nebo konfiguraci směrovačů.

MIB popisuje, jakým způsobem je sestavena funkční cesta zpracování dat. Informace uložené v MIB mohou poskytnout prvotní informace o procesu zpracování dat, ale mohou být i nápomocny k obnovení poškozené cesty zpracování dat. Základními prvky MIB pro DiffServ jsou klasifikace, měření, zpracování provozu, fronta a plánovač (Classifiers, Meters, Actions, Queues and Schedulers).

Funkční blok v MIB je složen ze dvou komponent. První komponentou je strukturální a druhou parametrický element. Konkrétním příkladem je např. měřič, kde strukturální element je blok, který realizuje samotné měření a parametrickým elementem je prvek, ve kterém jsou definovány parametry pro měření, jako např. konkrétní typ algoritmu Token-Bucket, rychlost doplňování tokenů, atd. Tato struktura je zobrazena na Obr. 3.1.



Obr. 3.1: Struktura funkčního elementu

Pro odkazování na jednotlivé bloky systému byl definován speciální ukazatel označený názvem *RowPointer*. Ukazatel je koncipován tak, aby mohl odkazovat na položky v jednotlivých tabulkách. Podle struktury funkčního elementu *RowPointer* může mít dvě mírně odlišné funkce. V prvním případě ukazatel zajišťuje pospojování funkčních elementů do cesty zpracování provozu. Funguje proto jako konektor mezi sousedními funkčními elementy. Druhý způsob využití je odkazování na sadu parametrů pro daný strukturální element. V takovém případě má *RowPointer* upřesňující funkci.

3.1 Cesta zpracování dat

Každé rozhraní aktivního prvku má vstupní a výstupní směr a má zpravidla odlišnou politiku pro zpracování provozu v každém směru. Tabulka *DiffServDataPathTable* poskytuje informaci o prvním bloku cesty zpracování provozu v závislosti na tom, přes které rozhraní (ifIndex z tabulky ifTable) a kterým směrem přistupují data na rozhraní (vstupní-výstupní). Objekt *DataPathTableEntry* definuje strukturu položek pro tuto tabulku. Největší význam má položka *DiffServDataPathStart*, která obsahuje ukazatel na první blok v cestě zpracování provozu. Jednotlivé funkční bloky cesty zpracování provozu jsou pak provázány příslušným ukazatelem s názvem končícím na „...NextFree“. Takovým funkčním blokem může být např. klasifikace, měření, blok zpracování provozu, algoritmus zahození paketů, fronta, nebo plánovač.

Pokud je hodnota ukazatele *RowPointer* nastavena na zeroDotZero nebo ukazatel neodkazuje na žádný objekt, předpokládá se, že se jedná o konec cesty zpracování provozu a pakety jsou předány následujícímu logickému modulu směrovače. Tím je nejčastěji směrovací podsystém v případě vstupního rozhraní a vysílač u výstupního rozhraní.

3.1.1 DiffServDataPathTable

Tabulka cest zpracování provozu (Data path table) obsahuje ukazatele *RowPointer* ukazující na první funkční blok cesty zvlášť pro každé rozhraní a směr toku. Jednotlivé toky se mohou sloučit, nebo rozdělit na paralelní cesty.

DiffServDataPathTable (OID: 1.3.6.1.2.1.97.1.1.1)	
DiffServDataPathIfDirection [Integer]	Udává směr datového toku (vtupní, výstupní).
DiffServDataPathStart [RowPointer]	Ukazuje na první funkční element cesty zpracování provozu.

Tabulka 3.1: DiffServDataPathTable

3.2 Klasifikace

Úlohou klasifikátorů je roztřídit příchozí pakety. Tabulka *DiffServClfrElementTable* umožňuje společně využívat několik třídících pravidel stejného, nebo odlišného typu. Následující funkční blok spojený s daným výstupem klasifikátoru je určen hodnotou *DiffServClfrElementNext*. Může se jednat o zpracování paketu v rámci následujícího klasifikátoru, nebo o ukončení klasifikace a přechod do dalšího bloku zpracování. Klasifikátor identifikuje aplikaci a rozhoduje o úrovni agregace podle jednoho, nebo více DSCP. Nicméně na okrajích sítě může dojít k tomu, že datový tok přichází neoznačen, nebo se značkém nedá věřit. V těchto případech je aplikován vícepoložkový klasifikátor (Multi-field Classifier).

Informace o tom, jaké pravidlo má být aplikováno se nachází v indexu *DiffServClfrElementSpecific*.

3.2.1 DiffServClfrTable

Tabulka udává všechny funkční elementy zajišťující klasifikaci provozu. Primární úlohou této tabulky je zabezpečit, aby hodnota *DiffServClfrId* byla jedinečná, při vytvoření nového záznamu. Struktura záznamu je definována objektem *DiffServClfrElementEntry*.

DiffServClfrTable (1.3.6.1.2.1.97.1.2.2)	
DiffServClfrId [IndexInteger]	Index označující vstupní proměnou klasifikátoru.

Tabulka 3.2: DiffServClfrElementTable

3.2.2 DiffServClfrElementTable

Tabulka udává vztah mezi konkrétním klasifikačním pravidlem a následujícím prvkem cesty zpracování provozu. Pole *DiffServClfrElementSpecific* ukazuje na parametrický element definující filtrovací pravidlo. Filtrovací pravidla mohou být různého typu.

DiffServClfrElementTable (1.3.6.1.2.1.97.1.2.4)	
DiffServClfrElementId [IndexInteger]	Index položky.
DiffServClfrPrecedence [Unsigned32]	Určuje prioritu třídících pravidel.
DiffServClfrElementNext [RowPointer]	Tento atribut ukazuje na následující prvek, kam jsou vedeny pakety z výstupu třídiče.
DiffServClfrElementSpecific [RowPointer]	Je ukazatelem na instanci v tabulce filtrů, která obsahuje konkrétní konfigurační parametry pro klasifikaci. Může se jednat např. o záznam v tabulce DiffServMultiFieldClfrTable. Hodnota zeroDotZero odpovídá filtru, který vybere všechny zatím nevybrané pakety.

Tabulka 3.3: DiffServClfrElementTable

3.2.3 DiffServMultiFieldClfrTable

Tabulka vícepoložkových klasifikačních filtrů pro IP datagramy.

DiffServMultiFieldClfrTable (1.3.6.1.2.1.97.1.2.6)	
DiffServMultiFieldClfrId [IndexInteger]	Pořadové číslo filtru pro vícepoložkový klasifikátor.
DiffServMultiFieldClfrAddrType [InetAddressType]	Typ IP adresy použité pro danou položku. Hodnoty jsou omezeny na adresy IPv4, nebo Ipv6.
DiffServMultiFieldClfrDstAddr [InetAddress]	Filtr pro IP adresu cílového uzlu. Je možné specifikovat pomocí prefixu adresy IPv4 nebo IPv6, ale nelze zde využít DNS název. Počet platných bitů prefixu je uložen v položce DiffServMultiFieldClfrDstPrefixLength.

DiffServMultiFieldClfrTable (1.3.6.1.2.1.97.1.2.6)	
DiffServMultiFieldClfrDstPrefixLength [InetAddressPrefixLength]	Délka CIDR prefixu zdrojové adresy uvedená v DiffServMultiFieldClfrDstAddr. V adresách IPv4 délka 0 bitů signalizuje použití jakékoliv adresy, délka 32 bitů signalizuje konkrétní hostitelskou adresu. Délka mezi 0 a 32 bitů signalizuje použití CIDR prefixu. Použití pro IPv6 je podobné, až na to, že délka je v rozsahu od 0..128 bitů.
DiffServMultiFieldClfrSrcPrefixLength [InetAddressPrefixLength]	Délka CIDR prefixu adresy zdroje uvedené v DiffServMultiFieldClfrSrcAddr. U adres IPv4 délka 0 bitů signalizuje možnost použití jakékoliv adresy a délka 32 bitů signalizuje adresu konkrétního uzlu. Délka mezi 0 a 32 bitů odpovídá použití CIDR prefixu. Použití pro IPv6 je podobné, až na to, že délka je v rozsahu od 0..128 bitů.
DiffServMultiFieldClfrSrcAddr [InetAddress]	Filtr pro IP adresu zdroje dat. Je možné specifikovat pomocí prefixu adresy IPv4 nebo IPv6, ale nelze zde využít DNS název. Počet platných bitů prefixu je uložen v položce DiffServMultiFieldClfrSrcPrefixLength.
DiffServMultiFieldClfrDscp [DSCP]	Vstupní hodnota DSCP paketu. Pokud je hodnota tohoto indexu –1, DSCP není definována a všechny další hodnoty DSCP se uvažují jako nedefinované.
DiffServMultiFieldClfrFlowId [Unsigned32]	Identifikátor datového toku v hlavičce Ipv6.
DiffServMultiFieldClfrProtocol [Unsigned32]	Identifikátor protokolu v hlavičce IPv4 nebo IPv6 (pole Next Header). Hodnota 255, zahrnuje všechny možné protokoly. Hodnota 255 je rezervována organizací IANA a hodnota 0 je použita pro Ipv6.
DiffServMultiFieldClfrDstL4PortMin [InetPortNumber]	Minimální hodnota cílového portu na transportní vrstvě (vrstva 4), např. TCP port 80.
DiffServMultiFieldClfrDstL4PortMax [InetPortNumber]	Maximální hodnota cílového portu na transportní vrstvě. Tato hodnota musí být rovna nebo větší než hodnota specifikovaná pro tento vstup v DiffServMulti FieldClfrDstL4PorMin.
DiffServMultiFieldClfrSrcL4PortMin [InetPortNumber]	Minimální hodnota portu zdroje dat na transportní vrstvě.
DiffServMultiFieldClfrSrcL4PortMax [InetPortNumber]	Maximální hodnota portu zdroje dat na transportní vrstvě. Tato hodnota musí být rovna nebo větší než hodnota specifikovaná pro tento vstup v DiffServMultiFieldClfrSrcL4PortMin

Tabulka 3.4: DiffServMultifieldClfrTable

3.1 Měření

Po klasifikaci nejčastěji následuje měření příslušné části provozu. Měřič kontroluje dodržení sjednaných parametrů příchozího provozu. Je to důležité z hlediska ochrany směrovače nebo části sítě před zahlcením. Pro měření jsou běžně využívány algoritmy založené na mechanismu Token-Bucket. U těchto mechanismů se zpravidla měří dodržení dlouhodobé průměrné rychlosti s určitou tolerovanou mírou proměnlivostí okamžité rychlosti. Jak bylo uvedeno v předchozích kapitolách, kombinací více mechanismů Token-Bucket je možné sestavit měřiče provozu, které vyhodnocují více úrovní rychlosti.

V rámci DiffServMIB jsou konkrétní instance měřičů dostupné v tabulce *MeterTable*. Každý záznam v tabulce obsahuje tři ukazatele. První ukazatel označuje funkční element, kam bude veden provoz, který nepřekročí měřenou rychlost. Druhý ukazatel označuje funkční

element, který zpracuje provoz překračující rychlost. Třetí ukazatel označuje parametrický element, který specifikuje parametry pro měření.

3.1.1 DiffServMeterTable

Tato tabulka obsahuje seznam instancí měřičů, které jsou definovány pro sledování rychlosti datových toků. Měřený datový tok je specifikován předchozími elementy cesty zpracování provozu. Specifické parametry měřiče jsou dostupné přes ukazatel *DiffServMeterSpecific*.

DiffServMeterTable (1.3.6.1.2.1.97.1.3.2)	
DiffServMeterId [IndexInteger]	Pořadové číslo měřiče.
DiffServMeterSucceedNext [RowPointer]	Ukazatel na další funkční blok v cestě zpracování provozu kam se posílá tok, který splňuje nastavené parametry.
DiffServMeterFailNext [RowPointer]	Ukazatel na další funkční blok v cestě zpracování provozu kam se posílá tok, který nesplňuje nastavené parametry.
DiffServMeterSpecific [RowPointer]	Tento atribut ukazuje na detailní parametry konkrétního měřiče. Položky musí být nastaveny explicitně. Například, DiffServMeterSpecific může ukazovat na položku v DiffServTBParamTable, která obsahuje nastavení parametrů Token Bucket.

Tabulka 3.5: DiffServMeterTable

3.1.2 DiffServTBParamTable

Tabulka parametrů pro měřiče typu Token Bucket. Každá položka v tabulce popisuje jeden měřič Token-Bucket. Víceúrovňové vyhodnocování rychlosti je možné realizovat spojením více měřičů typu Token-Bucket.

DiffServTBParamTable (1.3.6.1.2.1.97.1.4.2)	
DiffServTBParamId [IndexInteger]	Identifikátor měřiče
DiffServTBParamType [Autonomous Type]	Objekt označující konkrétní typ Token Bucket měřícího algoritmu.
DiffServTBParamRateRate [Unsigned32]	Rychlost plnění token-bucket v kb/s. Tento atribut je používaný pro: 1. CIR podle RFC2697 pro srTCM, 2. CIR a PIR podle RFC2698 pro trTCM, 3. CTR a PTR podle RFC2859 pro TSWTCM, 4. AverageRate podel RFC3290.
DiffServTBParamRateBurstSize [BurstSize]	Maximální počet bytů ve shluku dat. Tento atribut je užívaný pro: 1. CBS a EBS podle RFC2697 (srTCM), 2. CBS a PBS podle RFC2698 (trTCM), 3. Burst Size podle RFC3290.
DiffServTBParamInterval [Unsigned32]	Časový interval pro token bucket měřič.

Tabulka 3.6: DiffServTBParamTable

3.2 Zpracování provozu

Po měření provozu následuje skupina akcí zpracování provozu. Akce které jsou aplikovány jsou uloženy do tabulky *DiffServActionTable*. Položka *DiffServActionID* spolu s identifikátorem rozhraní jednoznačně identifikují rozhraní, na kterém se provádí měření provozu. Zbývající položky této tabulky jsou tvořeny dvěma ukazateli, *DiffServActionNext* ukazuje na další cestu zpracování dat a *DiffServActionSpecific* ukazuje na konkrétní tabulku, kde jsou specifikovány tyto akce:

- Značkování DSCP,
- zahození paketu,
- počítání paketů.

3.2.1 DiffServActionTable

Tabulka definuje způsoby zpracování paketů, které mohou být aplikovány na provoz. Jednotlivé způsoby zpracování lze zřetězit do kaskády. Specifické parametry zpracování jsou dostupné přes ukazatel *DiffServActionSpecific*.

DiffServActionTable (1.3.6.1.2.1.97.1.5.2)	
DiffServActionId [IndexInteger]	Identifikátor akce.
DiffServActionInterface [InterfaceIndexOrZero]	Index rozhraní, na kterém má být daná akce využita. Je získaná z tabulky ifTable, položky ifIndex OID:1.3.6.1.2.1.2.2.1.1 což může být odvozena z hodnoty DiffServDataPathStartEntry.
DiffServActionNext [RowPointer]	Ukazatel na další funkční blok v cestě zpracování provozu.
DiffServActionSpecific [Row Pointer]	Ukazatel na přídavné parametry akce. Běžně hodnota tohoto atributu ukazuje na instanci objektu DiffServDscpMarkActEntry nebo DiffServCountActEntry, ale může ukazovat i na jiné položky databáze MIB.

Tabulka 3.7: DiffServActionTable

3.2.2 Značkování DSCP

Tabulka obsahuje značky DSCP používané pro označení nebo přeznačení paketů využitím pole DSCP v hlavičce protokolu IP. Na jednotlivé položky tabulky může ukazovat atribut *DiffServActionSpecific*.

DiffServDscpMarkActTable (1.3.6.1.2.1.97.1.5.3)	
DiffServDscpMarkActDscp [DSCP]	Značka DSCP, která je nastavená během zpracování paketu značkovačem. Pole DSCP může být nastaveno jak na příchozím, tak i na odchozím rozhraní zařízení, přičemž tyto akce mohou probíhat paralelně na jednom směrovači.

Tabulka 3.8: DiffServDscpMarkActTable

3.2.3 Počítání paketů

Veškerý provoz zpracovaný jednotlivými komponenty zpracování provozu by měl být měřen. Je to důležité pro ladění a statistiky síťového provozu. Zdali tomu tak je označuje položka *DiffServCountActNextFree*. Pak li že ano, jsou naměřené údaje uloženy v položkách tabulky *DiffServCountActEntry*. Jednotlivé položky tabulky jsou indexovány pomocí *DiffServCountActId*. Dále jsou zaznamenány počty paketů a oktetů v položce *DiffServCountActPkts* resp. *DiffServCountActsOctets*.

3.2.3.1 DiffServCountActTable

Tabulka shrnuje všechny měřiče pro celý provoz, který je zpracován komponentami zpracování paketů. Strukturu záznamu definuje objekt *DiffServCountActEntry*.

DiffServCountActTable (1.3.6.1.2.1.97.1.5.5)	
DiffServCountActId [IndexInteger]	Identifikátor čítače - elementu zpracování paketu.
DiffServCountActOctecs [Counter64]	Udává počet bajtů v elementu zpracování paketů.
DiffServCountActPkts [Counter64]	Udává počet paketů v elementu zpracování paketů.

Tabulka 3.9: DiffServCountActTable

3.2.4 Zahození paketu

V případě DiffServMIB jsou zahazovače řazeny mezi prvky zpracování paketů. Zahazovače jsou shrnuty do zvláštní tabulky *DiffServAlgDropTable*. Základní funkcí algoritmického zahazovače je identifikace algoritmu pro rozhodování o zahození, zahození paketu a počítání zahozených paketů. V případě zahazování paketů při vstupu nebo při odchodu z fronty nebo u náhodného zahazování, musí být mechanismus provázán se sledovanou frontou. Provázanost je zajištěna ukazatelem v poli *DiffServAlgDropQMeasure*. U absolutního zahazovače taková provázanost není nutná.

3.2.4.1 DiffServAlgDropTable

Tabulka algoritmických/řízených zahazovačů paketů popisuje bloky schopné zahodit pakety na základě určitého algoritmu.

DiffServAlgDropTable (1.3.6.1.2.1.97.1.6.2)	
DiffServAlgDropId [IndexInteger]	Identifikátor konkrétní instance algoritmického zahazovače paketů.
DiffServAlgDropType [Integer]	Typ použitého algoritmu zahození. Možné hodnoty jsou: 1. other - tento typ vyžaduje další specifikaci v MIB 2. tailDrop - DiffServAlgDropQThreshold reprezentuje maximální velikost fronty, kterou když překročí parametr DiffServAlgDropQMeasure, pak další přicházející pakety budou zahozeny. 3. headDrop – velikost fronty, kterou když překročí parametr DiffServAlgDropQMeasure, pak další přicházející pakety budou zahozeny. 4. randomDrop – náhodné zahazování paketů probíhající na základě nastavení parametrů přes ukazatel DiffServAlgDropSpecific. 5. alwaysDrop – tento algoritmus zahazuje všechny pakety.
DiffServAlgDropNext [RowPointer]	Ukazatel na další funkční blok v cestě zpracování provozu.
DiffServAlgDropQMeasure [RowPointer]	Ukazuje na konkrétní záznam v tabulce DiffServQTable a tím identifikuje frontu, jejíž stav má algoritmus zahození sledovat.
DiffServAlgDropQTreshold [Unsigned32]	Udává prahovou hodnotu pro velikost fronty v bajtech, po jejímž dosažení je spuštěn algoritmus zahazování. Když hodnota DiffServAlgDropType je alwaysDrop(5), hodnota objektu je ignorována. Pro typy tailDrop(2) nebo headDrop(3) hodnota označuje velikost fronty, od které bude spuštěno zahazování. Pro jiné proprietární algoritmy bude třeba způsob využití tohoto parametru definovat dodatečně.
DiffServAlgDropSpecific [RowPointer]	Ukazatel na parametrický element, který obsahuje další údaje pro vybraný algoritmus zahazování. U proprietárních řešení, kdy DiffServAlgDropType = other (1), tyto parametry jsou v jiné části MIB. U zahazovače typu randomDrop (4) musí ukazovat na jednu z položek tabulky DiffServRandomDropTable.
DiffServAlgDropOctects [Counter64]	Počet bajtů, které byly zahozeny výsledkem deterministického procesu zahazování.
DiffServAlgDropPkts [Counter64]	Počet paketů, které byly zahozeny výsledkem deterministického procesu zahazování.
DiffServAlgRandomDropOctects [Counter64]	Počet bajtů, které byly zahozeny výsledkem náhodného procesu zahazování.
DiffServAlgRandomDropPkts [Counter64]	Počet paketů, které byly zahozeny výsledkem náhodného procesu zahazování.

Tabulka 3.10: DiffServAlgDropTable

3.3 Správa front a plánování

Následující modul v cestě zpracování paketů představují bloky správy front a plánování odesílání. Podobně jako u modelu řízení mechanismu DiffServ, i v případě DiffServMIB jsou tyto bloky modelovány pomocí elementárních prvků, ze kterých je pak možné sestavit i značně komplexní funkční jednotky. Základem systému front jsou i zde fronty typu FIFO. Všechny instance FIFO front jsou řazeny do tabulky *DiffServQTable*. S touto tabulkou úzce souvisí tabulka *DiffServSchedulerTable* která obsahuje plánovače. Právě tyto plánovače

sdružují jednoduché FIFO fronty do komplexnějšího celku. Podobně jako v případě jiných funkčních elementů i fronty jsou zapojeny do cesty zpracování provozu pomocí ukazatelů.

3.3.1 DiffServQTable

Tabulka obsahuje výčet všech front v systému. V modelu DiffServMIB je systém front realizován sadou jednoduchých front, kde každá fronta je přiřazena jedné z tříd. Přitom ve skutečnosti se může jednat o vzájemně provázané třídy, obsluhované společným plánovačem odesílání.

DiffServQTable (1.3.6.1.2.1.97.1.7.2)	
DiffServQId [IndexInteger]	Identifikátor ukazující na konkrétní instanci fronty.
DiffServQNext [RowPointer]	Ukazatel na další funkční blok v cestě zpracování dat, kam se posílá tok, který splňuje nastavené parametry.
DiffServQMinRate [RowPointer]	Tento ukazatel ukazuje na položku DiffServMinRateEntry udávající minimální rychlost obsluhy pro plánovač odesílání paketů.
DiffServQMaxRate [RowPointer]	Tento ukazatel ukazuje na položku DiffServMaxRateEntry, která udává maximální rychlost obsluhy pro plánovač odesílání paketů.

Tabulka 3.11: DiffServQTable

3.3.2 DiffServSchedulerTable

Tabulka obsahující všechny instance plánovače odesílání paketů. Cesta zpracování provozu může obsahovat i více plánovačů řazených za sebou.

DiffServSchedulerTable (1.3.6.1.2.1.97.1.8.2)	
DiffServSchedulerId [IndexInteger]	Identifikátor označující konkrétní plánovač.
DiffServSchedulerNext [RowPointer]	Ukazatel na další funkční blok v cestě zpracování dat, kam se posílá tok, který splňuje nastavené parametry.
DiffServSchedulerMethod [AutonomousType]	Plánovací algoritmus použitý v tomto plánovači. Standardní hodnoty označující běžné algoritmy jako: DiffServSchedulerPriority, DiffServSchedulerWRR a DiffServSchedulerWFQ jsou přímo specifikovány v této MIB. Další hodnoty mohou být specifikovaný v jiných MIB.
DiffServSchedulerMinRate [RowPointer]	Je ukazatelem do tabulky DiffServMinRateTable a indikuje prioritu nebo minimální výstupní rychlost z plánovače. Tento atribut je použit jen v případě, kdy má plánovač více úrovní.
DiffServSchedulerMaxRate [RowPointer]	Je ukazatelem do tabulky DiffServMaxRateTable a indikuje maximální výstupní rychlost z plánovače. Je li aplikováno více maximálních rychlostí (např. v případě více-rychlostního tvarovače provozu) je brána v úvahu první hodnota. Tento atribut je použit jen v případě, kdy plánovač má více úrovní.

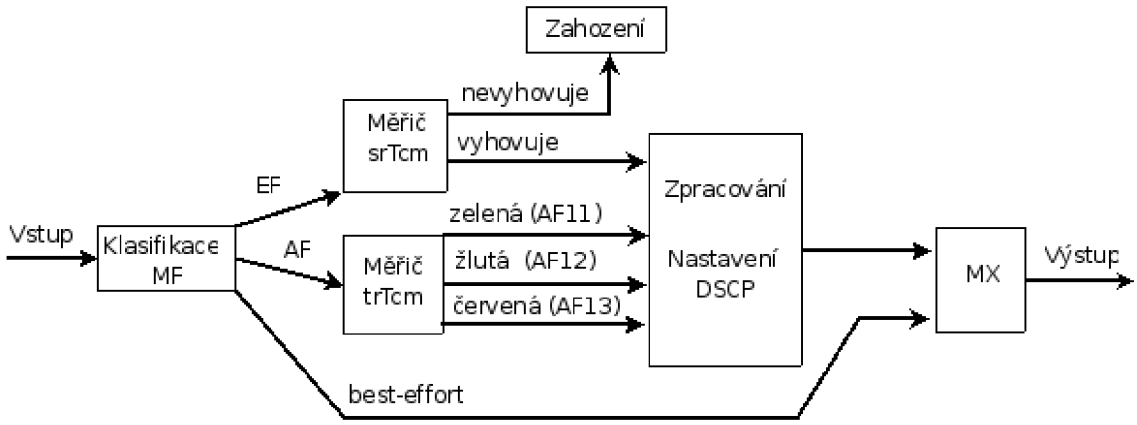
Tabulka 3.12: DiffServSchedulerTable

4 Návrh testovací struktury DiffServMIB

V rámci řešení diplomové práce jsem navrhl a realizoval část struktury MIB, která popisuje diferencované zpracování dat pomocí systému funkčních bloků. Tato struktura je rozdělena z důvodu přehlednosti na dvě samostatné části, na větev, která popisuje vstupní (inbound) směr, Obr. 4.1 a na větev pro výstupní (outbound) směr, Obr. 4.7. Schémata testovacích struktur pro jednotlivé směry jsou uvedeny v příloze.

4.1 Konfigurace vstupního směru na rozhraní

Prvním funkčním blokem cesty zpracování provozu ve vstupním směru je klasifikátor, který rozlišuje tři služby. Každé z nich je přiřazena jedna třída provozu. První službou je přenos hlasu, která je zařazena do třídy EF (Expedited Forwarding), druhou službou je přenos dat využívající protokol FTP, které přísluší třída AF a do třetí třídy označené jako best-effort se řadí ostatní provoz. Pro třídu best-effort končí klasifikací diferencované zpracování dat ve vstupním směru a data jsou předána dalšímu zpracování aktivnímu prvku. U zbylých dvou tříd následujícím blokem zpracování dat po klasifikaci je měření. Pro třídu EF se použije měřič který pracuje metodou Single Rate Three Color Marker (srTCM) [3] a pro třídu AF měřič Two Rate Three Color Marker (trTCM) [4]. Po měření následuje blok zpracování provozu, kde se přiřadí značky DSCP dle tříd provozu. Po značkování je již datový tok předán dalšímu zpracování, tj. směrovacímu podsystému.

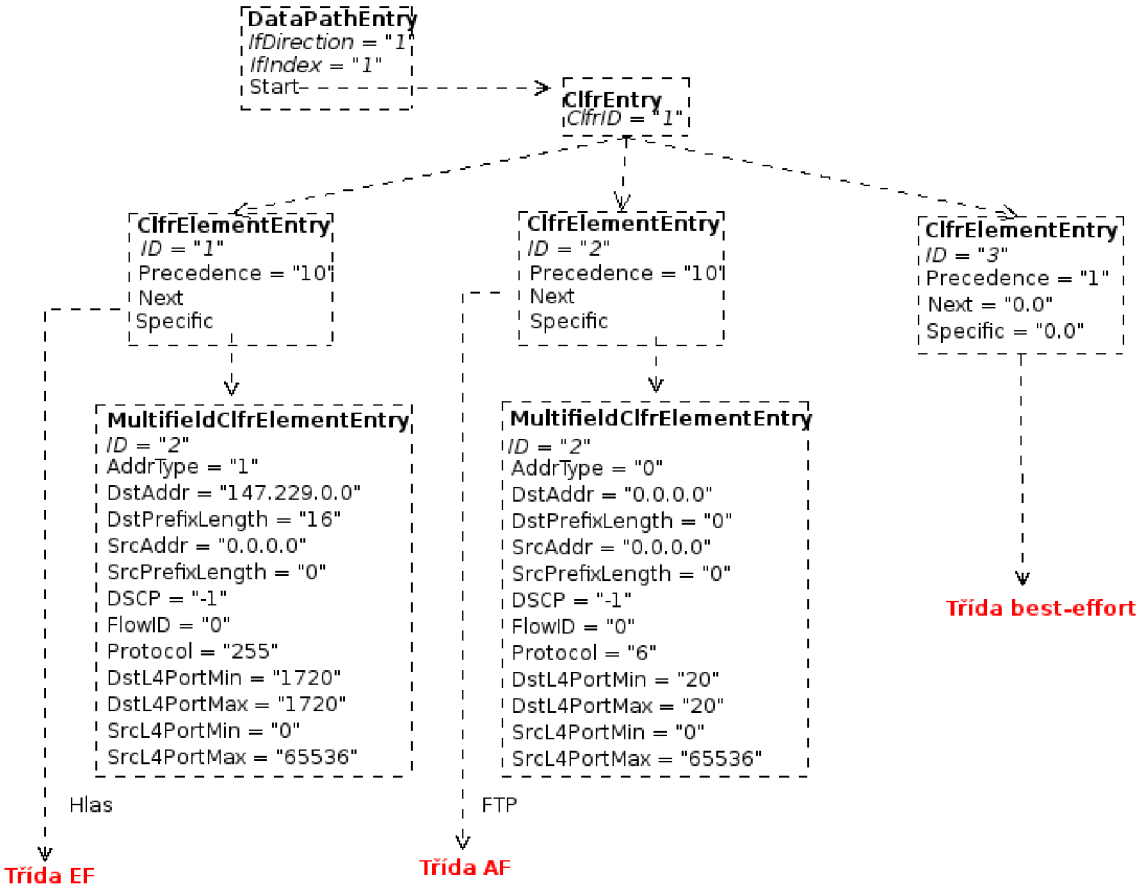


Obr. 4.1: Zpracování paketů ve vstupním směru

4.1.1 Blok klasifikace

Paket přicházející na vstupní rozhraní si z tabulky *DiffServDataPathTable* zjistí následující funkční blok, kterým je klasifikátor. Je zvolen vícepoložkový klasifikátor (Multifield Classifier), který vybírá pakety na základě jednoho, nebo více třídících pravidel definovaných v tabulce *DiffServClfrElementTable*. Jsou definovány tři pravidla. První pravidlo definuje

třídu EF (ID 1) a atribut *DiffServClfrElementSpecific* ukazuje na první záznam do tabulky *DiffServMultifieldClfrTable,* která specifikuje filtr vícepoložkového klasifikátoru a pokud je nalezena shoda, je paket zařazen do třídy EF. Stejným způsobem probíhá klasifikace pro třídu AF (ID 2). Jestliže vstupní datový tok nevyhovuje ani jednomu ze dvou pravidel, je zařazen do třídy best-effort (ID 3). Strukturu MIB klasifikátoru popisuje Obr. 4.2.



Obr. 4.2: Klasifikátor MF

4.1.2 Blok měření

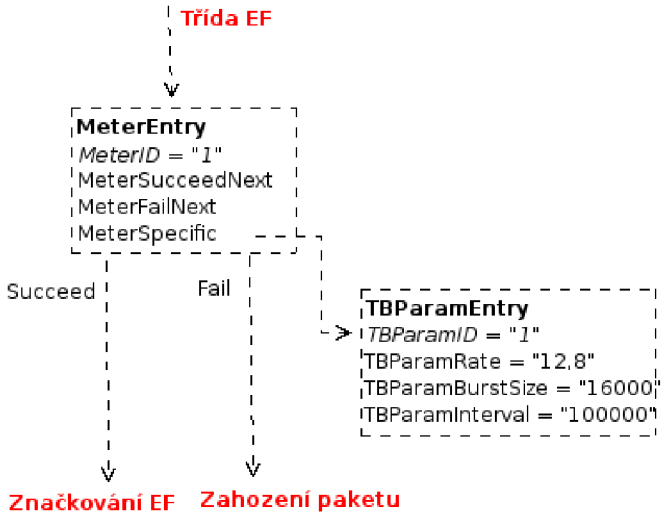
Definice měřičů se provádí v tabulce *DiffServMeterTable* a parametry měřiče Token-Bucket jsou definovány v tabulce *DiffServTBParamTable.* Pro měření jsou použity dva měřiče, každý z nich používá rozdílnou metodu měření. Pro datový tok, který je zařazen do třídy EF se použije metoda srTCM a pro třídu AF metoda trTCM.

4.1.2.1 srTCM

Měřič srTCM tvoří první záznam v tabulce *DiffServMeterTable* (Id 1) a na jeho vstup přichází neoznačený datový tok klasifikovaný do třídy EF. Metoda měření srTCM je realizována jedním mechanismem Token-Bucket, jehož parametry jsou definovány v tabulce *DiffServTBParamEntry* na kterou ukazuje atribut *DiffServMeterSpecific*. Atribut

DiffServTBParamRate specifikuje garantovanou průměrnou rychlost a atribut *DiffServTBParamBurstSize* garantovanou velikost shluku CBS.

Pokud měřený datový tok vyhovuje těmto parametrům, ukazatel *DiffServMeterSuceedNext* ukazuje do tabulky zpracování provozu *DiffServActionTable* a naopak, ukazatel *DiffServMeterFailNext* ukazuje do tabulky *DiffServAlgDropTable*, kde se provede zahození paketu.



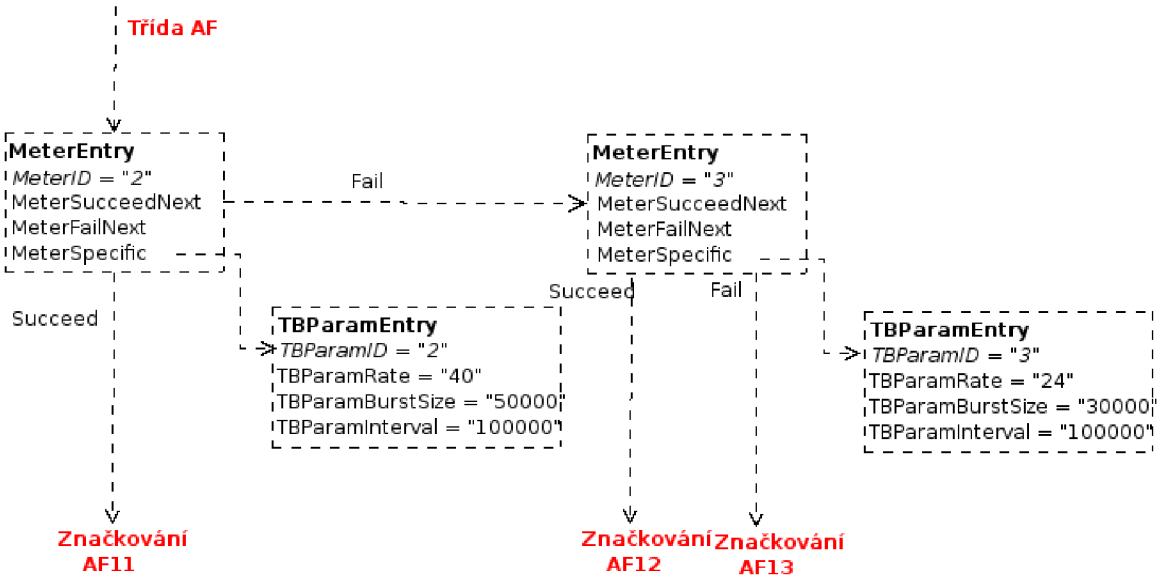
Obr. 4.3: Struktura měřiče srTCM

4.1.2.2 trTCM

Metoda trTCM, využívá víceúrovňového měření a podle výsledků měření dělí provoz do tří tříd. Metoda je realizována dvěma mechanismy Token-Bucket. Jsou tedy definovány dvě pravidla v tabulce *DiffServMeterTable*. První pravidlo definuje Token-Bucket C, jehož parametry jsou definovány v tabulce *DiffServTBParamTable*. Atribut *DiffServTBParamRate* udává garantovanou průměrnou rychlost CIR a atribut *DiffServTBParamBurstSize* garantovanou velikost shluku CBS. Druhý Token-Bucket P má definovanou maximální rychlost PIR a maximální velikost shluku PBS. Atribut *DiffServTBParamInterval* udává časový interval aktualizace shodně pro oba mechanismy Token-Bucket.

Na počátku jsou oba měřiče Token-Bucket P i C plné, počet tokenů $Tp(0) = PBS$ a počet tokenů $Tc(0) = CBS$.

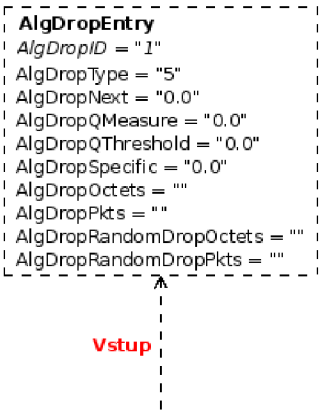
- Jestliže $Tp(t) - B < 0$, paket je zařazen do třídy AF13 (obarven červeně),
- jestliže $Tc(t) - B < 0$, je zařazen do třídy AF12 (obarven žlutě),
- paket je zařazen do třídy AF11 (obarven zeleně) a počet tokenů u obou mechanismů Token-Bucket P a C je snížen o B.



Obr. 4.4: Struktura měřiče trTCM

4.1.3 Blok zahození

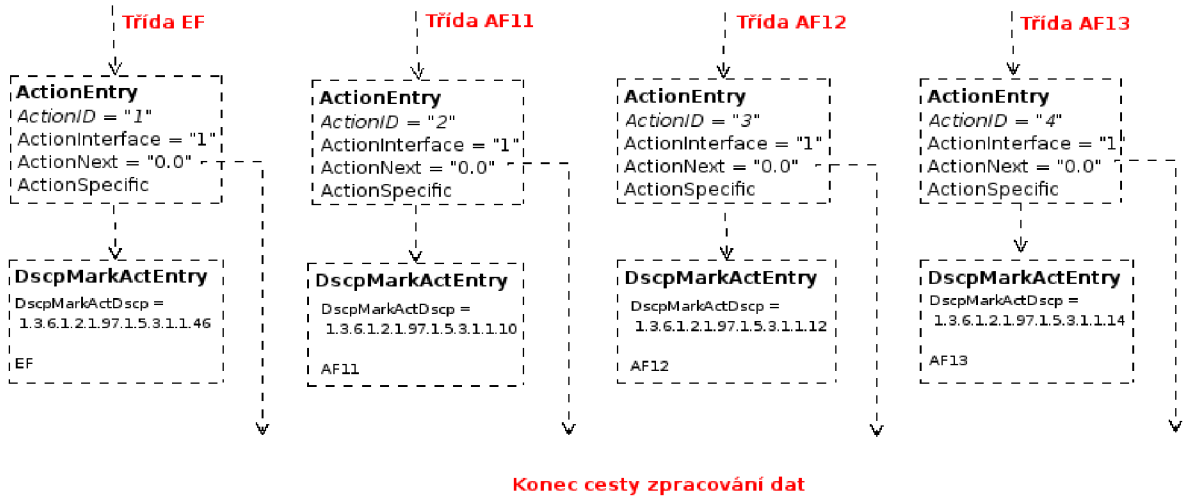
Tento blok plní jednoduchou funkci zahazování příchozích paketů, které nevyhovují měření. Důležitou položkou tabulky *DiffServAlgDropTable* je parametr *DiffServAlgdropType*, který určuje typ algoritmu. Zde je vybrán algoritmus „always drop“, který zahodí všechen provoz, který se objeví na vstupu tohoto bloku, a proto není potřeba dále specifikovat další parametry.



Obr. 4.5: Blok zahození

4.1.4 Blok zpracování

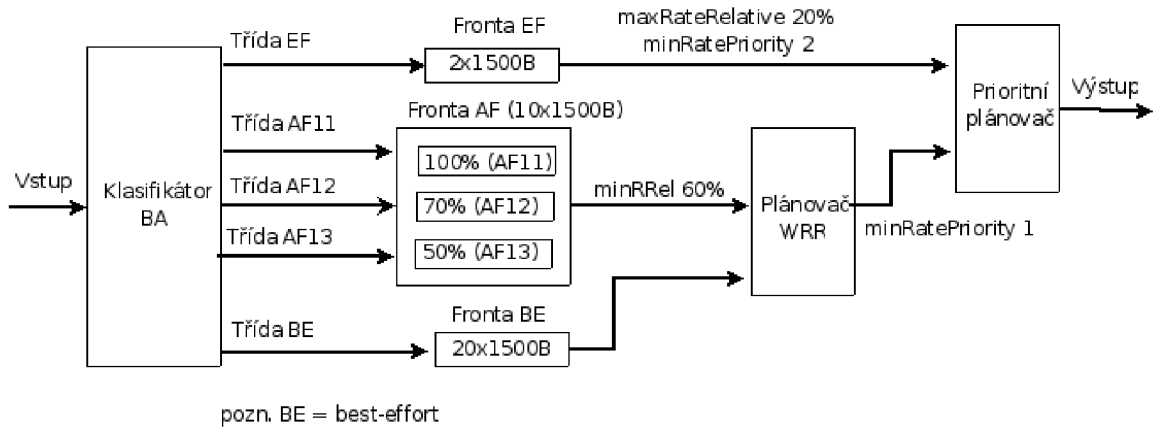
Posledním funkčním blokem zpracování dat ve vstupním směru je blok zpracování provozu, který má za úkol přidělit značky DSCP. V tabulce *DiffServActionTable* je pro každou značku DSCP definováno samostatné pravidlo. Atribut *DiffServActionSpecific* ukazuje do tabulky *DiffServDscpMarkActTable*, na atribut *DiffServDscpMarkActDscp*, kde jsou specifikovány příslušné značky DSCP, které se nastaví do pole DSCP v záhlaví IP paketu. Atribut *DiffServActionNext* obsahuje hodnotu 0.0, což znamená, že cesta zpracování dat zde končí.



Obr. 4.6: Blok zpracování provozu

4.2 Konfigurace výstupního směru na rozhraní

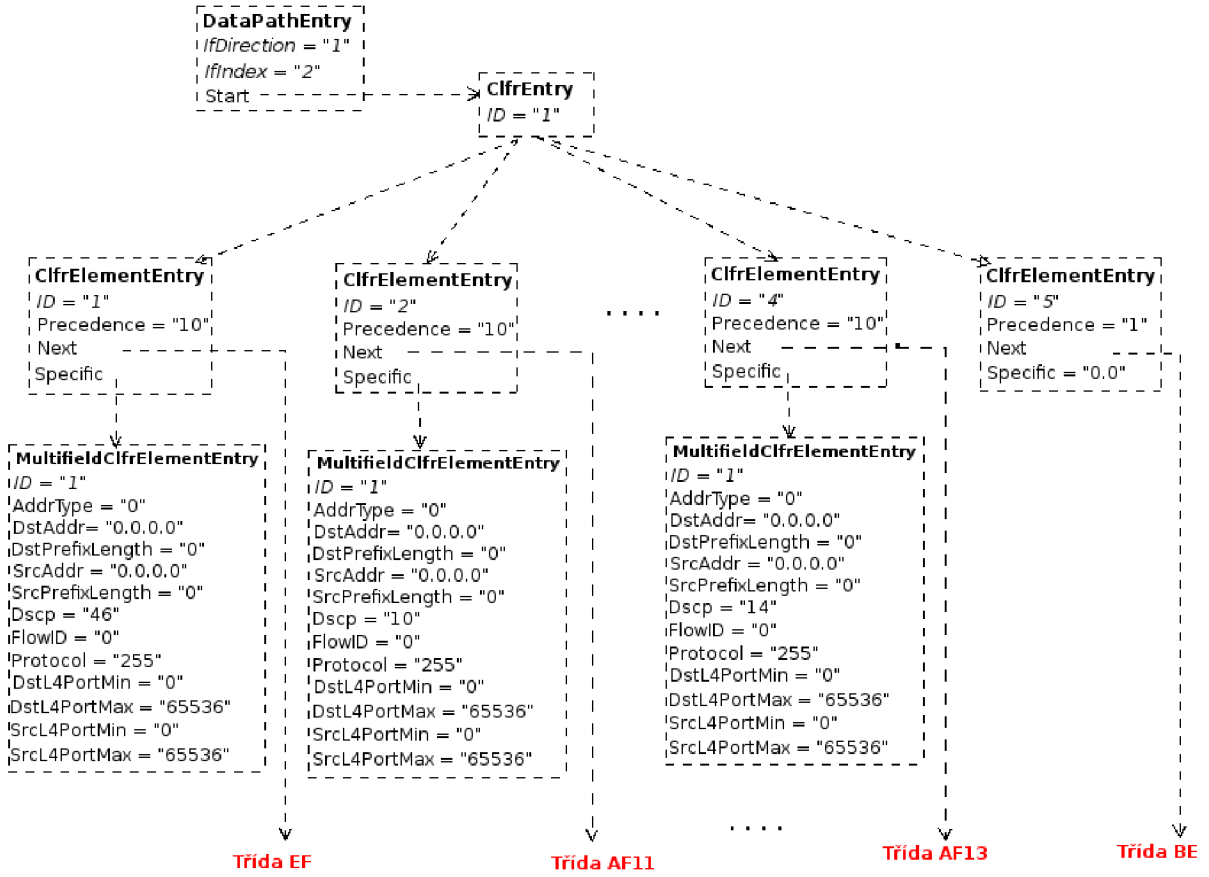
Zpracování dat ve výstupním směru rovněž začíná klasifikací. Předpokládá se, že paket byl už dříve označen a proto na základě přidělené DSCP značky vybere klasifikátor paket a zařadí jej do jedné z pěti tříd. Jsou definovány tři fronty. První fronta, která je vyhrazena pro třídu EF má velikost 3000B. Fronta AF sdružuje třídy AF11, AF12, AF 13 a má velikost 15000B a třetí fronta pro třídu best-effort má velikost 30000B. O zpracování paketů a obsluhu front se starají dva bloky plánovače. Každý z nich je řízen odlišným algoritmem. Snahou bylo, aby třída EF měla absolutní prioritu při zpracování. Proto se využívá dvoustupňová struktura zobrazena na Obr. 5.1. Algoritmus WRR (Weighted Round Robin) obsluhuje frontu AF a frontu best-effort. Fronta AF má zaručeno minimálně 60% šířky pásma plánovače WRR. Druhý, prioritní plánovač vybírá pakety k odeslání na základě priority a platí, že pakety s vyšší prioritou jsou obslouženy nejdříve. Proto je nutné nastavit prioritu oběma datovým tokům, které vstupují do prioritního plánovače. Prioritní plánovač je posledním blokem zpracování dat.



Obr. 4.7: Zpracování paketů ve výstupním směru

4.2.1 Blok klasifikace

Prvním funkčním blokem ve výstupním směru je klasifikátor, který podle přidělené značky DSCP třídí provoz do pěti tříd. V tabulce *DiffServClfrElementTable* bylo tedy třeba vytvořit pro každou třídu jedno pravidlo. Tabulka *DiffServMultifieldClfrTable* obsahuje také pět pravidel. Každé z pravidel specifikuje nastavení klasifikátoru sdruženého zacházení (Behaviour Aggregate – BA). Důležitý je atribut *DiffServMultifieldClfrDscp*, kde je specifikována značka DSCP, podle které probíhá zařazení paketu do odpovídající fronty. Atribut *DiffServClfrElementNext* ukazuje do tabulky *DiffServAlgDropTable*, kde se nastaví velikosti jednotlivých front.



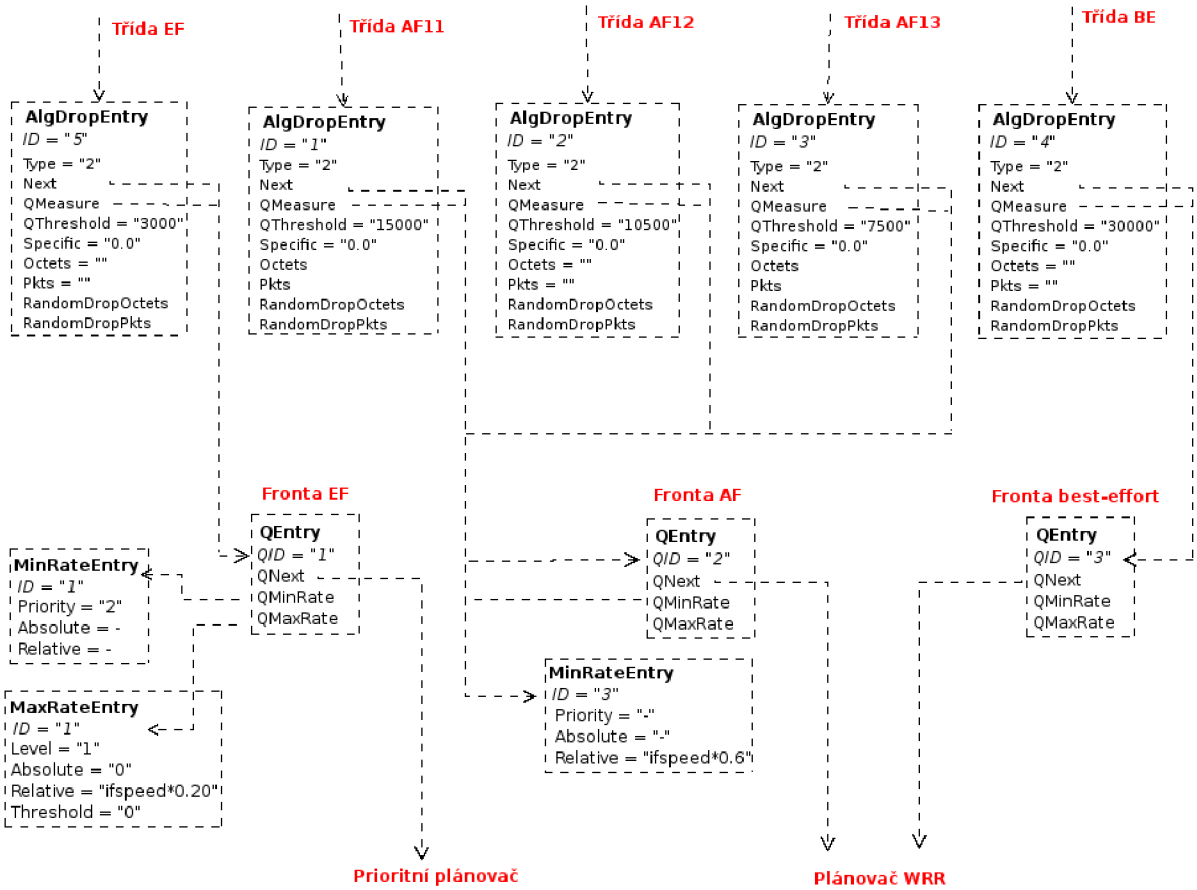
Obr. 4.8: Struktura klasifikátoru BA

4.2.2 Fronty

Fronta pro třídu AF sdružuje provoz ze všech tříd AF1x a potřebujeme tedy zajistit , aby bylo pro každou třídu rezervováno určité procento kapacity fronty. Toho dosáhneme nastavením atributu *DiffServAlgDropQThreshold,* který udává velikost fronty v bajtech, po jejíž dosažení je spuštěn zahazovací mechanizmus specifikovaný atributem *DiffServAlgDropType* (tail-drop). To, na kterou frontu jsou tyto pravidla aplikovány, ukazuje atribut *DiffServAlgDropQMeasure*. Pro třídy AF11, AF12 a AF13 bude tedy shodně ukazovat do tabulky *DiffServQTable* na frontu pro třídu AF.

V tabulce *DiffServQTable* definujeme pro každou frontu jedno pravidlo. Pokud je následujícím blokem prioritní plánovač jako v případě fronty EF (Id 1), je nutné nastavit prioritu fronty. Atribut *QminRate* ukazuje do tabulky *MinRateTable*, kde je specifikována priorita atributem *MinRatePriority*. Atribut *QmaxRate* ukazuje do tabulky *MaxRateEntry*. Nastavením atributu *MaxRateRelative* zajistíme, že fronta EF bude mít rezervováno maximálně 20% šířky pásma. Pro frontu AF (Id 2) ukazuje atribut QMinRate do tabulky *MinRateTable*, kde nastavením atributu *MinRateRelative* rezervujeme minimálně 60% šířky pásma pro frontu AF.

V případě fronty best-effort (Id 4) je specifikována pouze velikost fronty, není rezervována žádná šířka pásma ani priorita fronty a následujícím blokem zpracování dat je plánovač WRR.

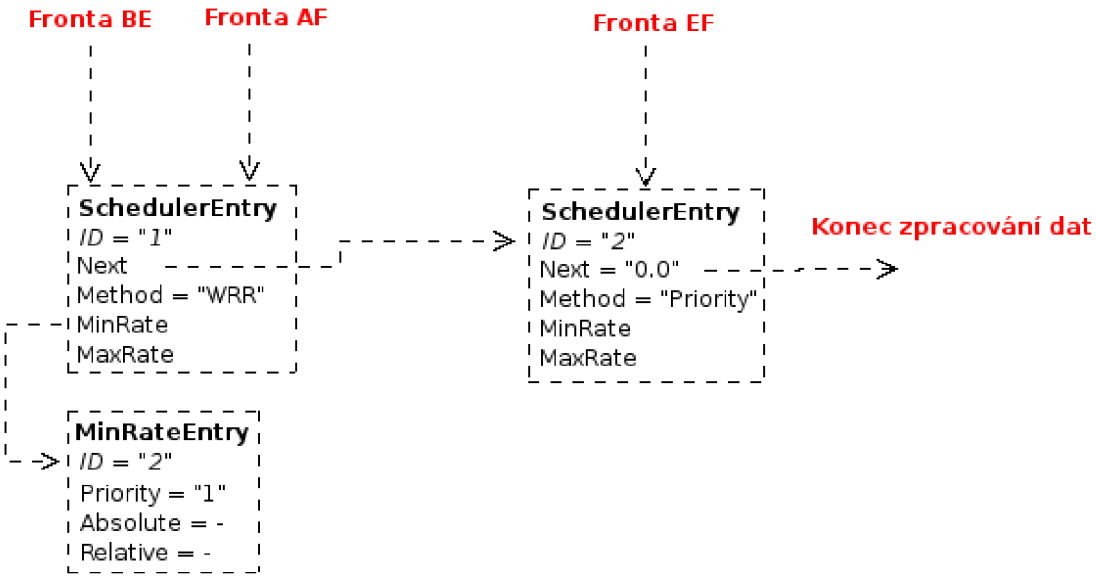


Obr. 4.9: Struktura konfigurace front

4.2.3 Plánovač

Cesta zpracování končí odesláním dat skrze výstupní rozhraní. Plánovače obsluhují fronty, řídí výběr paketů a jejich odeslání. Algoritmus obsluhy fronty nastavujeme atributem *SchedulerMethod* v tabulce *DiffServSchedulerTable*. Fronty AF a best-effort jsou obsluhovány plánovačem WRR (Id 1). Atribut *DiffServSchedulerNext* ukazuje na následující blok, kterým je prioritní plánovač, a proto musíme nastavit prioritu datovému toku. Toto provedeme nastavením atributu *DiffServMinRatePriority* v tabulce *DiffServMinRateTable*, na kterou ukazuje atribut *DiffServSchedulerMinRate*.

Pro prioritní plánovač platí, že datový tok, který má nastavenu nejvyšší prioritu je zpracován nejdříve. V tomto příkladě má třída EF zajištěnu nejvyšší prioritu zpracování. Hodnota 0.0 v atributu *SchedulerNext* prioritního plánovače znamená, že již není specifikována další cesta zpracování dat a data jsou odeslána skrze výstupní rozhraní.



Obr. 4.10: Dvoustupňová struktura plánovače

5 Implementace struktury DiffServMIB do prostředí OPNET Modeler

Prvním krokem vlastní implementace je přenesení tabulek testovací struktury DiffServMIB do OPNET Modeleru (OM). Hlavním kriteriem byla snadná dostupnost, přehlednost a konfigurovatelnost jednotlivých atributů struktury DiffServMIB. Jako nejvhodnější metodu jsem zvolil implementaci pomocí systému atributů, které umožňují každému objektu v OM definovat vlastní soubor atributů. Tyto atributy mohou mít hierarchickou strukturu a proto jsou vhodné pro reprezentaci DiffServMIB tabulek.

5.1 Funkce atributů

Atributy uchovávají data, která popisují a konfigurují model nebo objekt v OM. Dají se snadno nastavovat jako vlastnosti objektu či modelu. Poskytují informace, specifikují základní charakteristiky, modifikují interní strukturu a ovlivňují chování objektu. Každý atribut má identifikátor tj. jméno a hodnotu. Jméno je unikátní v kontextu ve kterém je atribut definován (model, objekt) a používá se jako identifikátor pro čtení, či zápis hodnoty atributu.

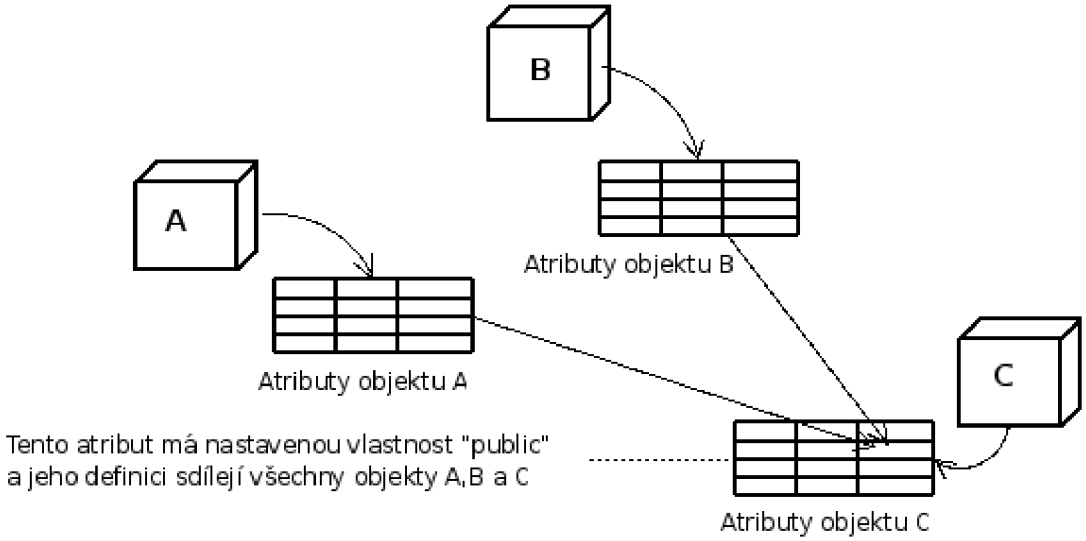
Atributy pracují s různými typy dat v závislosti na jejich určení a každý atribut nabývá hodnoty, která jsou definována datovým typem. Tyto zahrnují jak celočíselné typy, tak textové řetězce a také typ boolean. Podporu pro víceúrovňový systém atributů poskytuje datový typ „compound“. Základní typy atributů a jejich datových typů ukazuje Tabulka 5.1.

Typ atributu	Datový typ C	Význam
Integer	int	Kladné, nebo záporné celé číslo s velikostí 2^{31} – 64. Tyto proměnné typicky reprezentují identifikátory, ukazatele spojení, nebo priority.
Double	double	64-bitové číslo s dvojitou (double) přesností s plovoucí řádovou čárkou.
Compound	Objid	Speciální datový typ, umožňuje konstrukci stromového systému atributů. Atributy „compound“ se často používají k reprezentaci dat ve formě tabulek. Každý řádek tabulky může obsahovat další vnořený objekt a jednotlivé sloupce reprezentují atributy vnořeného objektu.
Toggle	int	Tento atribut nabývá pouze dvou hodnot, reprezentuje stavy „true“ nebo „false“. Obvykle tyto hodnoty mají význam „enabled“ nebo „disabled“.
String	char []	Znaková proměnná, typicky jména objektů.
Color	int	Hodnota tohoto atributu nastavuje barvu objektu, pokud to objekt umožňuje.

Tabulka 5.1: Základní typy atributů OM

5.1.1 Sdílení atributů

V mnoha případech může mít atribut stejný význam pro několik různých objektů (např atribut „data rate“, který určuje rychlost přenosu dat pro objekty typu „transmitter channels“, „receiver channels“ a také pro linky). Abychom nemuseli atribut definovat pro každý objekt samostatně, nastavíme atributu vlastnost „public“. To nám umožní jednoduše sdílet definici a vlastnosti atributu dalšími objekty.

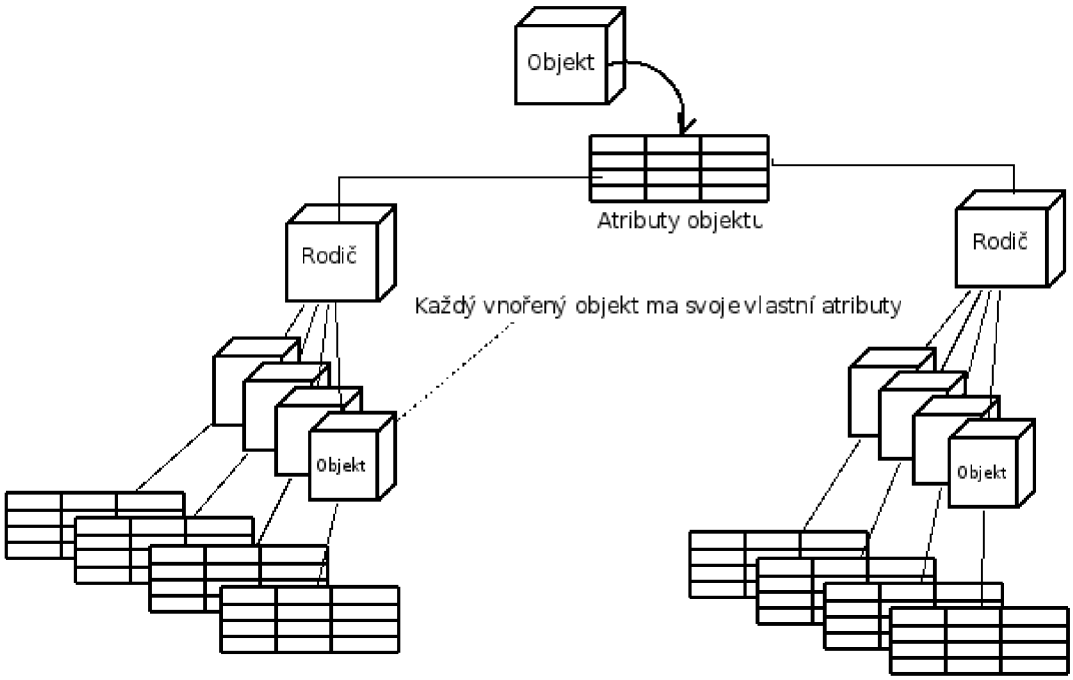


Obr. 5.1: Sdílení atributů

5.1.2 Vnořené atributy

Modeler podporuje dva druhy atributů, jednoduché - „simple“, které mají pouze název a hodotu a vnořené, jejichž datový typ je „compound“. Vnořené atributy umožňují vytvářet stromovou strukturu atributů. Každý atribut typu „compound“ může obsahovat neomezený počet vnořených atributů, které se mohou dále větvit.

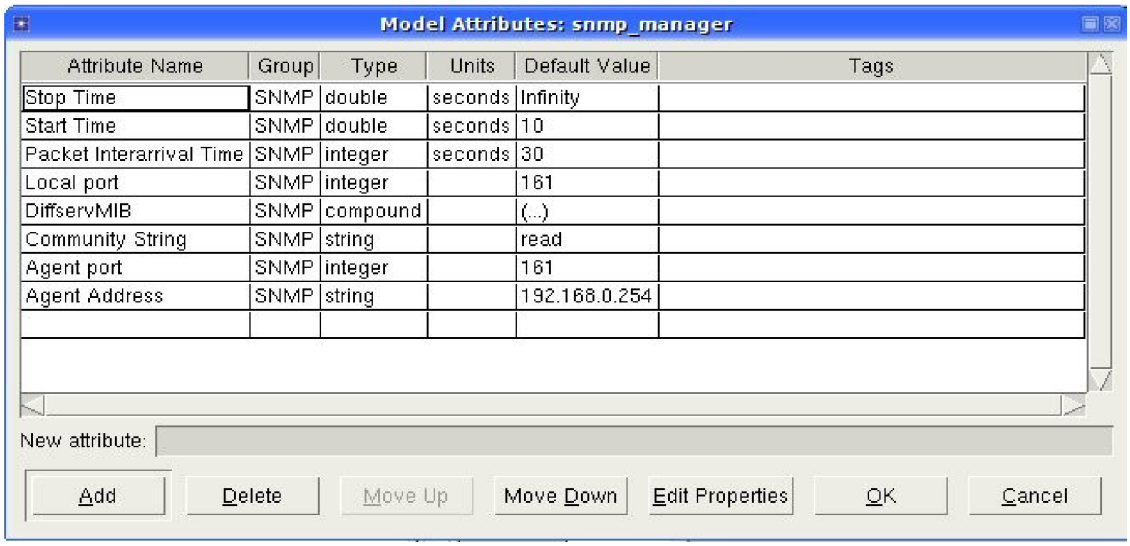
Během simulace se každému atributu přidělí jedinečné Object ID (OID). OID se použije jako identifikátor pro přístup ke vnořeným atributům. Atribut typu „compound“ je považovaný za rodiče vnořených objektů a k těmto objektům lze přistupovat pomocí funkce *op_topo_child()*. Podobně, funkce *op_topo_parent()* může být použita pro zjištění OID rodiče ze znalosti OID jednoho z vnořených objektů.



Obr. 5.2: Systém vnořených atributů

5.2 Definice atributů

Atributy lze definovat na úrovni modelu uzlu nebo procesu. Definicí se rozumí nastavení jména atributu, datového typu a dalších vlastností určujících chování atributu. My budeme pracovat na úrovni modelu procesu, a proto budeme atributy definovat v editoru procesu. Zde v menu *Interfaces* vybereme *Model Attribute's,* otevře se dialogové okno definice atributů, které je zobrazeno na Obr. 5.3.

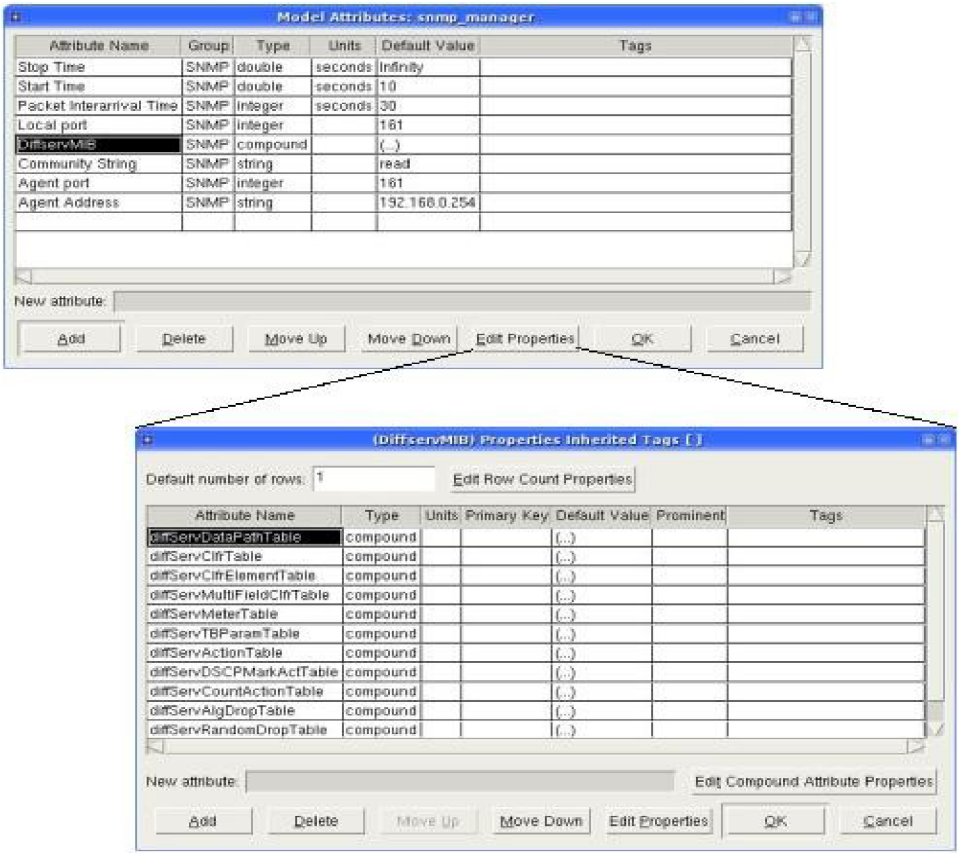


Obr. 5.3: Definice atributů OM

Položka	Význam
Attribute Name	Jméno atributu.
Group	Určuje skupinu, pod kterou bude atribut patřit.
Type	Datový typ atributu.
Units	Jednotka hodnoty atributu.
Default Value	Nastavuje implicitní hodnotu atributu.
Tags	Speciální vyhrazená slova, která identifikují atribut

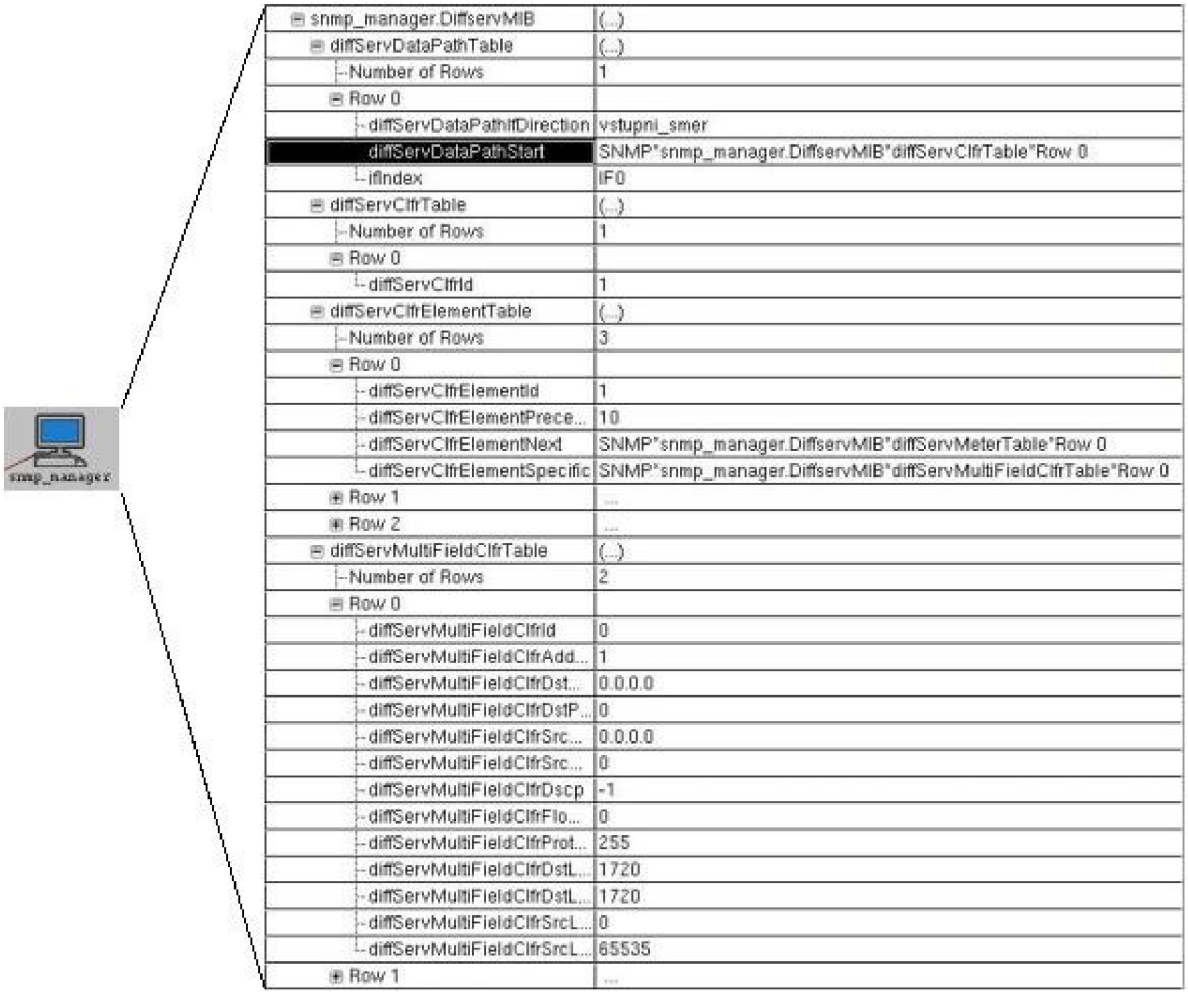
Tabulka 5.2: Význam položek - definice atributů

Všimneme si hodnoty „compound“ atributu snmp_manager.DiffServ.MIB, která má hodnotu „(...)“ a znamená že obsahuje vnořené atributy, které můžeme definovat po kliknutí na tlačítko „Edit Properties“, viz. Obr. 5.4. Takto lze nadefinovat celou strukturu DiffServMIB. První úroveň vnořených atributů obsahuje definici jednotlivých tabulek DiffServMIB. Každá tabulka obsahuje svoje vlastní atributy.



Obr. 5.4: Definice vnořených atributů

Hodnoty atributů se nastavují v menu „Edit Attributes“ na úrovni modelu. Příklad nastavení ukazuje Obr. 5.5, kde jsou nastaveny atributy tabulky *DataPathTable*, *ClfrTable*, *ClfrElementTable* a *MultifieldClfrTable* objektu snmp_manager.



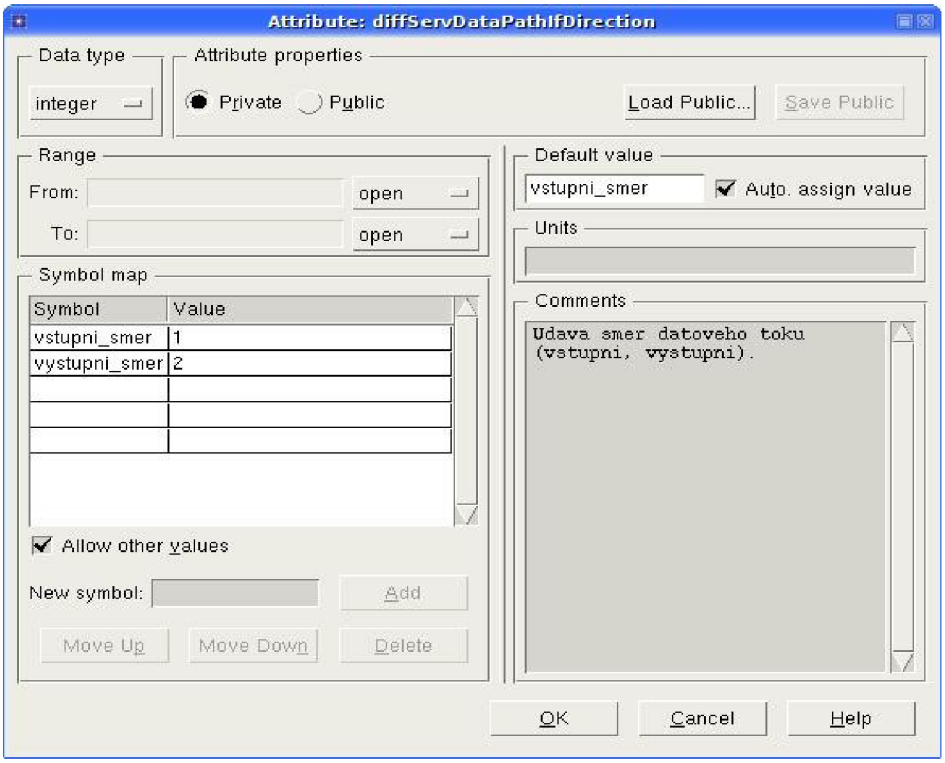
Obr. 5.5: Atributy objektu snmp_manager

5.2.1 Vlastnosti atributů

Kromě jména a hodnoty lze specifikovat další vlastnosti, které určují způsob chování atributu. Tyto vlastnosti se nastavují pro každý atribut samostatně a jsou přístupné po kliknutí na tlačítko „Edit Properties“ v okně definice atributů. Nastavení vlastností je zobrazeno na Obr. 5.6. Tabulka 5.3 uvádí význam jednotlivých vlastností atributu.

Položka	Popis
Data Type	Datový typ atributu.
Attribute Properties	Určuje, jakým způsobem jsou atributy sdíleny: • Private – definice atributu se vztahuje pouze na aktuální objekt. • Public – nastavuje sdílení atributů, viz. kapitola 5.1.1.
Range: From, To	Nastavuje rozmezí použitelných hodnot pro atributy typu integer, nebo double.
Default Value	Specifikuje výchozí hodnotu atributu.
Units	Určuje v jakých jednotkách je hodnota atributu, má pouze informativní charakter.
Symbol map	Umožňuje definovat symbolická jména s definovanou hodnotou.
Comments	Komentář.

Tabulka 5.3: Položky tabulky - vlastnosti atributu



Obr. 5.6: Nastavení vlastnosti atributu

Konfigurací těchto vlastností lze nastavit rozmezí použitelných hodnot, definovat symbolické mapy, přidávat komentáře nebo zvolit výchozí hodnotu atributu, která se nastaví při jeho vytvoření. Lze také nastavit způsob sdílení atributů. Položka „public“ nám umožní uložit definici atributu do souboru.

5.1 Funkce pro práci s atributy

Při inicializaci, nebo v průběhu simulace potřebujeme často získat hodnotu atributu, nebo ji případně změnit. Opnet používá pro práci s atributy vlastní funkce. Univerzálními funkcemi jsou **op_ima_obj_attr_get** a **op_ima_obj_attr_set**.

op_ima_obj_attr_get ()

Získá hodnotu atributu ve specifikovaném objektu.

Syntaxe:

op_ima_attr_get (objid, attr_name, value_ptr)

Argument	Typ	Popis
objid	Objid	ID objektu, ke kterému chceme přistupovat
attr_name	Const *char	název atributu
value_ptr	Void*	ukazatel, do kterého se uloží hodnota atributu

Tabulka 5.4: Parametry funkce op_ima_obj_attr_get

op_ima_obj_attr_set ()

Zapíše hodnotu atributu ve specifikovaném objektu.

Syntaxe:

op_ima_attr_set (objid, attr_name, value)

Argument	Typ	Popis
objid	Objid	ID objektu, ke kterému chceme přistupovat
attr_name	Const *char	název atributu
value	Void*	nová hodnota pro atribut.

Tabulka 5.5: Parametry funkce op_ima_obj_attr_set

Pokud známe datový typ hodnoty atributu ke kterému přistupujeme, je vhodné použít přímo funkci, která pracuje s konkrétním datovým typem, např. pokud je datový typ atributu string, použiji funkci **op_ima_obj_attr_set_str()**. Tabulka 5.6 shrnuje možné typy funkcí a jejich datové typy se kterými tyto funkce pracují.

Funkce	Datový typ atributu
op_ima_obj_attr_set_*str*	Const char*
_int32	int
_dbl	double
_toggle	int (OPC_FALSE, OPC_TRUE)
_objid	objid

Tabulka 5.6: Datové typy funkcí op_ima_obj_attr

5.2 Příklad práce s atributy

V tomto příkladě si ukážeme jak přistupovat k tabulce atributů *DiffServMultiFieldClfrTable*. Obr. 5.7 ukazuje hierarchický strom atributů. Atributy typu compound obsahují další vnořené atributy, objekty. Nadřazený atribut typu compound je vždy rodičovský objekt a jednotlivé vnořené atributy jsou potomky tohoto objektu. Sloupec Value obsahuje aktuální hodnoty atributů. Hodnota atributů typu compound obsahuje identifikátor (ID) atributu a pokud chceme přistupovat ke vnořeným atributům, potřebujeme znát tuto hodnotu.

```
            //VYCTENI PARAMETRU Z TABULKY MultiFieldClfrTable

//ziskam id compound atributu DiffServMIB
op_ima_obj_attr_get (snmp_agent_id, "DiffServMIB", &objid_DiffServMIB);

//ziskam id nulteho radku v tabulce DiffServMIB
compound_mib = op_topo_child (objid_DiffServMIB, OPC_OBJTYPE_GENERIC, 0);

//ziskam objid tabulky MultifieldClfrTable
op_ima_obj_attr_get_objid (compound_mib, "DiffServMultiFieldClfrTable",
&objid_mfClfrTable);

//ziskam pocet vnorených objektu v tabulce MultiFieldClfrTable
pocet_radku = op_topo_child_count (objid_mfClfrTable,
OPC_OBJTYPE_GENERIC);

prvni_radek = op_topo_child (objid_mfClfrTable, OPC_OBJTYPE_GENERIC, 1);

op_ima_obj_attr_get_int32 (prvni_radek, "DiffServMultiFieldClfrId",
&mf_clfr_id);

op_ima_obj_attr_get_str (prvni_radek, "DiffServMultiFieldClfrDstAddr", 15,
mf_clfr_dst_addr);

op_ima_obj_attr_get_str (prvni_radek, "DiffServMultiFieldClfrSrcAddr", 15,
mf_clfr_src_addr);

op_ima_obj_attr_get_int32 (prvni_radek, "DiffServMultiFieldClfrDscp",
&mf_clfr_dscp);

op_ima_obj_attr_get_int32 (prvni_radek, "DiffServMultiFieldClfrProtocol",
&mf_clfr_protocol);

op_ima_obj_attr_get_int32 (prvni_radek,
"DiffServMultiFieldClfrDstL4PortMin", &mf_clfr_dst_port_min);

op_ima_obj_attr_get_int32 (prvni_radek,
"DiffServMultiFieldClfrDstL4PortMax", &mf_clfr_dst_port_max);


op_ima_obj_attr_get_int32                                    (prvni_radek,
"DiffServMultiFieldClfrSrcL4PortMin", &mf_clfr_src_port_min);

op_ima_obj_attr_get_int32
(prvni_radek,"DiffServMultiFieldClfrSrcL4PortMax", &mf_clfr_src_port_max);
```

Tabulka 5.7: Program na vyčtení atributů

Nejprve zjistíme ID rodičovského atributu DiffServMIB a uložíme do proměnné. K tomu použiji standardní funkci *op_ima_obj_attr_get* a jako parametry ji předáme id objektu snmp_agent, název atributu a adresu, kam má uložit ID atributu. Abychom mohli přistupovat k jednotlivým tabulkám, potřebujeme zjistit ID prvního vnořeného objektu, který představuje nultý řádek, k tomu nám poslouží funkce *op_topo_child*.

objid **op_topo_child ()**

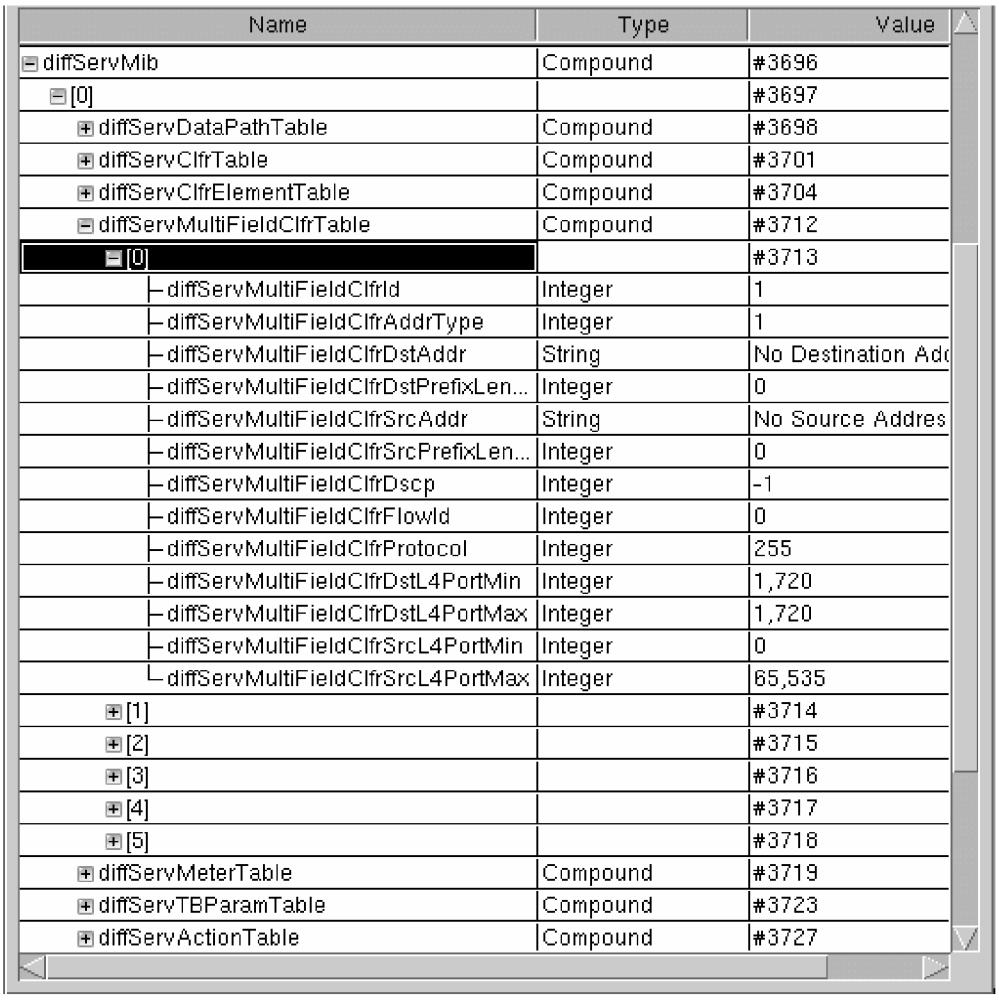
Získá ID potomka

Syntaxe:

op_topo_child (parent_objid, child_type, child_index)

Argument	Typ	Popis
parent_objid	Objid	ID rodičovského objektu
child_type	Int	představuje číselný kód, který specifikuje typ objektu
child_index	Int	index potomka

Tabulka 5.8: Parametry funkce op_topo_child



Obr. 5.7: Hierarchický strom atributů DiffServMIB

Parametr `OPC_OBJTYPE_GENERIC` značí, že objekt je uživatelsky definovaný atribut typu compound. Index značí číslo řádku, v našem případě bude tedy číslo řádku 0. V dalším kroku zjistím ID tabulky *DiffServClfrMultifieldTable*. Pokud potřebuji znát kolik vnořených objektů obsahuje tabulka *MultiFieldTable*, nebo obecně atribut typu compound, je vhodné použít funkci *op_topo_child_count*.

int **op_topo_child _count ()**

Získá počet potomků požadovaného typu, které obsahuje rodičovský objekt.

Syntaxe:

op_topo_child (parent_objid, child_type)

Argument	Type	Popis
parent_objid	Objid	ID rodičovského objektu
child_type	Int	představuje číselný kód, který specifikuje typ objektu

Tabulka 5.9: Parametry funkce op_topo_child_count

Nyní již známe všechny potřebné hodnoty a můžeme metodou get nebo set přistupovat k atributům. Podobným způsobem uvedeným v tomto příkladě můžeme pracovat se všemi atributy v Opnetu.

6 Statické provázání atributů OM <-> DiffServMIB

Při provázání tabulek DiffServMIB s funkcemi pro zpracování provozu v OM je potřeba nastavit QoS na síťových prvcích v Opnetu tak, aby toto nastavení v každém okamžiku odpovídalo konfiguraci tabulek DiffServMIB. Snahou bylo maximální využití stávajících funkčních elementů v Opnetu. V počáteční fázi realizace jsem předpokládal, že konfigurace QoS se provede nastavením příslušných atributů v OM v závislosti na nastavení parametrů DiffServMIB. Tímto nastavením se potom bude řídit uzlový model IP a aplikuje konfiguraci QoS na procházející provoz. Po testování se ukázalo, že atributy slouží jen na počáteční inicializaci a v průběhu simulace nemůžeme vytvářet nové, ani mazat stávající položky jednotlivých atributů. Pro dynamické provázání atributů se tedy tato metoda ukázala jako nevhodná Pro dynamické nastavení atributů jsem navrhl vlastní procesní model execMib. Tento model kontroluje ve stanovených intervalech tabulku DiffServMIB a provádí nastavení příslušných atributů. Tato kapitola se věnuje pouze statické konfiguraci. Na směrovači se lokálně konfiguruje podpora QoS pomocí atributů, které jsou přístupné v menu **IP >> IP Routing Parameters >> IP QoS Parameters**. Nyní si tabulky DiffServMIB rozebereme jednotlivě a zjistíme možné vazby s atributy QoS.

6.1 Blok klasifikace a třídění provozu

Nejprve definujeme klasifikační filtry, podle kterých budeme klasifikovat datové toky, následně vytvoříme třídy provozu a nakonec aplikujeme toto nastavení na konkrétní rozhraní na síťovém prvku.

6.1.1 Nastavení klasifikačních filtrů

Klasifikační filtry se nachází v tabulce *MultifieldClfrElementTable*. Je zde definována sada atributů. S těmito atributy se porovnává každý příchozí paket a v případě shody se datové jednotky řadí do jednotlivých tříd. Opnet umožňuje definovat klasifikační filtry pomocí ACL. Toto řešení není z hlediska implementace nejvhodnější, protože v ACL nejsou definovány některé atributy které se nachází v tabulce *MultifieldClfrElementTable*. Pro většinu případů je však toto nastavení dostačující. Tabulka 6.1 porovnává možnosti konfigurace ACL a tabulky klasifikačních filtrů. Význam atributů tabulky *ClfrElementTable* je uveden v kapitole 3.2.3. Postup konfigurace standardních ACL v Opnetu je podrobně popsán v [6], kap.6.1.1. Rozšířené ACL se konfigurují podobně jako standardní, v Opnetu se konfigurace nachází v menu **IP >> IP Routing Parameters >> Extended ACL Configuration**.

DiffServMIB	Extended ACL
AddrType	není
DstAddr	Destination
DstPrefixLength	není
SrcAddr	Soure
SrcPrefixLength	není
DSCP	DSCP
FlowID	není
Protocol	Protocol
DstL4PortMin	Destination Port
DstL4PortMax	
SrcL4PortMin	Source Port
SrcL4PortMax	

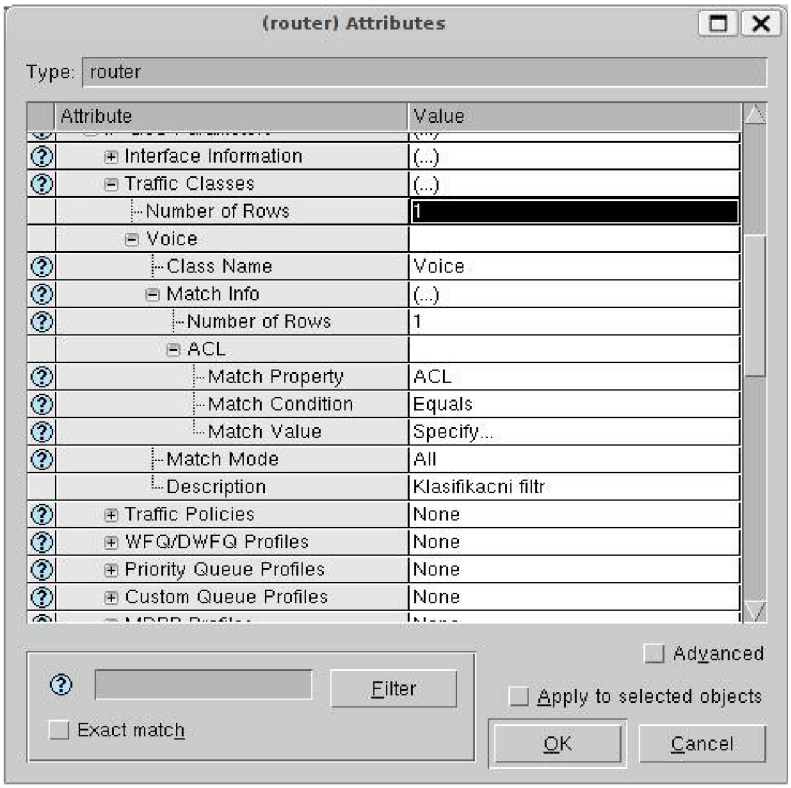
Tabulka 6.1: Porovnání atributů DiffServMIB a Extended ACL

Význam jednotlivých položek rozšířených ACL listů:

- *Destination* – IP adresa a maska podsítě cílového uzlu nebo podsítě (klienta),
- *Source* – IP adresa a maska podsítě zdroje paketů (klienta),
- *DSCP* – značka DSCP v hlavičce IP paketu,
- *Protocol* – číslo protokolu dle IANA,
- *Destination Port* – číslo portu cílového uzlu,
- *Source Port* – číslo portu zdrojového uzlu.

6.1.2 Třídy provozu

Podle tabulky *ClfrElementTable* se datový tok rozděluje do příslušných tříd provozu. V Opnetu definujeme třídy provozu v menu **IP >> IP QoS Parameters >> Traffic Classes**. Atribut **Number of Rows** odpovídá počtu definovaných tříd v tabulce *ClfrElementTable*. Každá třída by měla mít svoje jméno. Dále je potřeba specifikovat podle jakých pravidel se bude příchozí paket porovnávat. V případě klasifikace typu MF nastavíme atribut **Match Property** na ACL a vybereme již definovaný ACL list. Pokud je klasifikace typu BA, nastavíme atribut Match Property na hodnotu **DSCP** a vybereme podle jaké značky se bude porovnávat. Pro atribut precedence, který je definovaný v DiffServMIB a udává prioritu jednotlivých tříd nemá Opnet odpovídající nastavení. Příklad nastavení je na Obr. 6.1. Popis všech možností nastavení tříd provozu v Opnetu je uveden v [6], kap. 6.1.2.



Obr. 6.1: Konfigurace tříd provozu

6.2 Nastavení politiky provozu

Politika se aplikuje na vybranou třídu provozu. Nastavení politiky najdeme v menu **IP >> IP QoS Parameters >> Traffic Policies.** Nejdůležitější je položka **Set Property**, pro vstupní směr nastavíme příslušný policer profile (viz. kapitola Měření a značkování paketů), pro výstupní směr plánovač paketů. Vytvořený profil politiky pak nastavíme na příslušné rozhraní

(viz. kapitola Nastavení rozhraní). Podrobnější popis jednotlivých položek nastavení je opět uveden v [6].

6.3 Měření a značkování paketů

Pro měření se v Opnetu používá algoritmus CAR, tento algoritmus umožňuje měření a tvarování provozu. Nastavení algoritmu se nachází v menu **IP >> IP QoS Parameters >> Policer Profile**. V DiffServMIB se definice měřičů nachází v tabulce *MeterTable*. Parametry měřiče jsou specifikovány v tabulce *TBParamTable*.

Pro každý měřič definovaný v tabulce *MeterTable* vytvoříme jeden Policer profil. Počet řádků tedy odpovídá počtu definovaných měřičů. Podle metody měření se nastaví konfigurace. Výhodou policer profilu je možnost nastavení akcí, které se podle výsledku měření aplikují na pakety.

6.3.1 Metoda srTCM

Tato metoda používá k měření provozu pouze jeden TB. Provoz, který vyhovuje měření je předán dalšímu zpracování do dalšího bloku a naopak, provoz který nevyhovuje měření je zahozen. Jako parametry měření se udávají CIR, CBS a EBS. Tabulka 6.2 uvádí vzájemné vazby mezi tabulkou MeterTable a Policer profilem. Význam jednotlivých atributů tabulky MeterTable je uveden v kapitole 3.1.1

MeterTable (srTCM)	**Policer Profile**
Param Rate (CIR)	Average Rate
Burst Size (CBS/EBS)	Conform Burst Size
	Excess Burst Size
Interval	není

Tabulka 6.2: Porovnání atributů tabulky MeterTable s Policer profilem pro metodu srTCM

Atributy Policer profilu:

- *Average Rate* – průměrná dlouhodobá rychlost dat (bit/s),
- *Conform Burst Size* – velikost shluku dat (v bitech), po které už část provozu překračuje stanovený limit,
- *Excess Burst Size* – velikost shluku dat, před tím než všechen provoz překročí stanovený limit rychlosti.

6.3.2 Metoda trTCM

Tato metoda využívá k měření dva TB, každý má svoji rychlost, CIR a PIR, dále má každý TB specifikovanou maximální velikost shluku CBS a PBS. Protože v jedné instanci měřiče v tabulce MeterTable je možné specifikovat pouze jednu rychlost, je potřeba vytvořit pro tuto metodu dvě instance měřiče. Policer profil v Opnetu však pracuje s oběma rychlostma v jednom profilu, proto nám v Opnetu stačí vytvořit pouze jednu instanci Policer profilu.

Výstupem měřiče jsou tři datové toky. Datový tok, který vyhovuje měření prvního TB s rychlostí CIR, datový tok, který překročí parametr CIR, ale nepřekročí PIR a poslední datový tok, který překročí obě rychlosti.

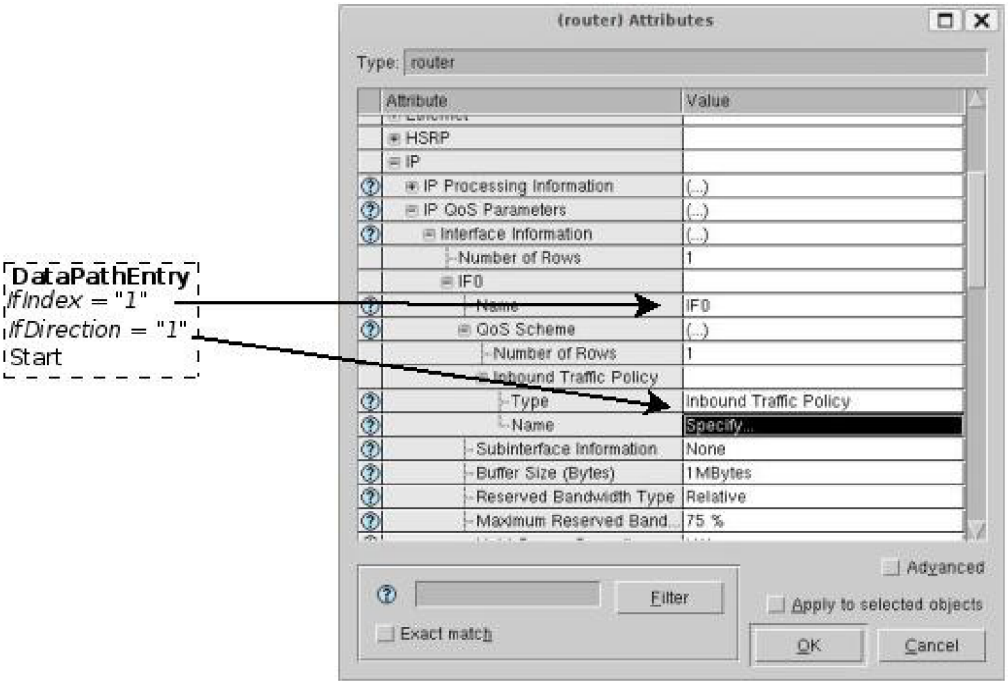
MeterTable (trTCM)	**Policer Profile**
Param Rate (CIR)	Average Rate
Param Rate (PIR)	Peak Information Rate
Burst Size (CBS/PBS)	Conform Burst Size
	Excess Burst Size
Interval	Není

Tabulka 6.3: Porovnání atributů tabulky MeterTable s Policer profilem pro metodu trTCM

Jediným rozdílem oproti konfiguraci metody srTCM je nastavení atributu Peak Information Rate, který udává maximální rychlost druhého TB (bit/s).

6.4 Nastavení rozhraní

Tabulka *DataPathTable* obsahuje informace o rozhraní a směru, ve kterém se bude aplikovat nastavení QoS. Důležité jsou atributy **IfIndex** a **IfDirection**. V Opnetu se odpovídající nastavení nachází v tabulce **Interface Information**, kterou nalezneme v menu **IP >> IP QoS Parameters**. O tom, jaký druh politiky se bude uplatňovat rozhoduje atribut **Type**. Zvolíme Inbound Traffic Policy v případě vstupního směru, nebo Outbound Traffic Policy v případě směru výstupního a vybereme příslušnou politiku (viz. kapitola 6.2 Nastavení politiky provozu).



Obr. 6.2: Nastavení rozhraní v Opnetu

6.5 Plánování a obsluha front

Opnet umí pracovat s několika algoritmy pro plánování a obsluhu front. V testovací MIB tabulce, používáme dva algoritmy, WRR a PQ. Pro realizaci těchto algoritmů, lze v Opnetu použít dva mechanismy, MDRR (Modified Deficit Round Robin) a CQ (Custom Queuing). Každý z těchto mechanismů umožňuje navíc vybrat jednu frontu, která pracuje ve striktně prioritním režimu. Tyto mechanismy můžeme konfigurovat globálně skrze objekt „QoS Attribute Config“, potom máme tento profil k dispozici na všech prvcích, nebo lokálně, jednotlivě pro každý uzel.

6.5.1 MDRR

Tento mechanismus strategie plánování front zajišťuje, že neprázdné fronty jsou obsluhovány cyklicky (round-robin). Při každé obsluze fronty, je odebráno z vrcholu fronty určité množství dat. Tento druh strategie umožňuje garanci relativní šířky pásma a umí také pracovat s jednou či více frontami jako striktně prioritními.

Každá fronta pod MDRR je definována dvěma proměnnými:

Quantum Value (QV) – hodnota udává průměrný počet bajtů, které jsou odeslány při každé obsluze fronty. Tato hodnota je vypočítána v závislosti na šířce pásma a MTU na rozhraní.

Deficit Counter (DC) – tento čítač zjišťuje, kolik bajtů bylo odesláno při každé obsluze fronty.

Funkce mechanismu je následující:

Pakety ve frontě jsou odesílány, pokud je hodnota DC větší než nula. Každý odeslaný paket zmenší proměnnou DC o hodnotu, která odpovídá délce paketu v bajtech. Pokud je hodnota DC nulová, nebo záporná, obsluha frontu přeskočí a neodesílá žádné pakety. Při každé obsluze je hodnota DC zvětšena o hodnotu QV.

V Opnetu najdeme plánovač MDRR v menu **IP >> IP QoS Parameters >> MDRR Profiles**.

Popis jednotlivých atributů:

- *Bandwidth value* – hodnota udává šířku pásma, která bude přiřazena ke třídě.
- *Bandwidth type* – určuje, zda bude proměnná Bandwidth value udávat absolutní hodnotu šířky pásma v b/s, nebo procentuální vyjádření šířky pásma.
- *Priority* – umožňuje prioritní zpracování fronty, nastavením hodnoty na enable se tato fronta stane striktně prioritní a její obsluha má přednost přede všemi ostatními.
- *Queue limit* – udává maximální množství paketů, které mohou být uloženy ve frontě.

Tabulka 6.4 popisuje nastavení plánovače MDDR s ohledem na nastavení front v testovací tabulce DiffServMIB.

MDRR					
Name	**EF**	**AF11**	**AF12**	**AF13**	**BE**
Bandwidth Type	Relative	Relative	Relative	Relative	Relative
Bandwidth Value	20	100	70	50	100
Priority	Enable	Disabled	Disabled	Disabled	Disabled
Queue Limit	2	10	10	10	20

Tabulka 6.4: Nastavení plánovače MDRR

6.5.2 Custom Queuing

Tato metoda je také založen na algoritmu WRR. Pro každou frontu definujeme určitý počet bajtů, které jsou odeslány při každé obsluze. Dalším parametrem je velikost/kapacita fronty, která se udává v paketech.

Popis jednotlivých atributů:

- *Byte count* - udává počet bajtů, které budou odeslány při každé obsluze fronty. Pokud je fronta nastavena jako LLQ, je tato hodnota ignorována.
- *Maximum Queue Size* - maximální velikost fronty v paketech. Pokud je fronta plná, všechny příchozí pakety jsou zahazovány.
- *RED/WRED* - aktivní správa front využitím mechanismů Random Early Detection nebo Weighted Random Early Detection.
- *Queue Category* – určuje, zda je fronta typu *Low Latency*, nebo *Default Queue.*
  - *Low Latency* – fronta s nejvyšší prioritou, ostatní fronty jsou obslouženy jen potom, co je tato fronta prázdná.
  - *Default Queue* – do této fronty jsou zařazeny pakety, které nesplňují klasifikační kriteria alespoň pro jednu existující frontu.

Pakety jsou zařazeny do fronty na základě klasifikace. Klasifikovat můžeme podle těchto pravidel:

- Protocol: TCP, UDP, OSPF, IGRP, EIGRP, ICMP,
- TOS: specifikuje hodnotu v poli Type of Service ip paketu,
- zdrojová IP adresa,
- cílová IP adresa,
- zdrojový TCP/UDP port,
- cílový TCP/UDP port.

Jestliže paket nevyhovuje ani jedné definované podmínce, je zařazen do vyhrazené fronty, tato fronta se nakonfiguruje jako„Default Queue“.

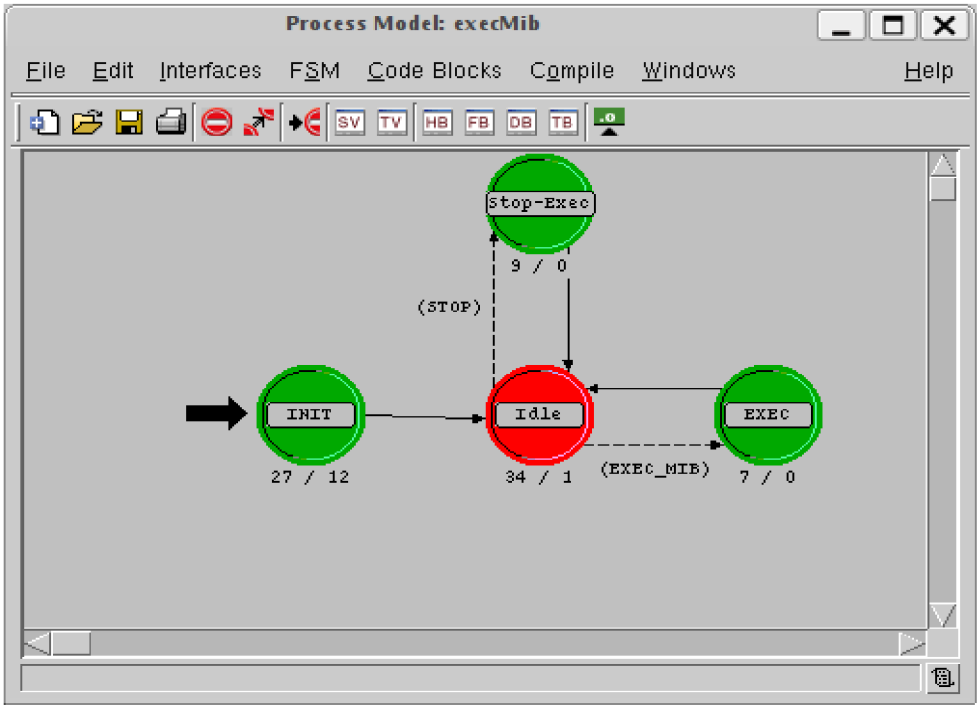
Tabulka 6.5 ukazuje nastavení profilu Custom Queuing v závislosti na konfiguraci tabulky DiffServMIB.

Custom Queuing					
Name	**EF**	**AF11**	**AF12**	**AF13**	**BE**
Byte Count	-	15000	10500	7500	30000
Maximum Queue Size	2	10	10	10	20
Queue Category	Low Latency	-	-	-	Default

Tabulka 6.5: Nastavení plánovače Custom Queuing

7 Model execMib

Tento model má za úkol ve stanovených časových intervalech kontrolovat tabulku DiffServMIB a provádí konfiguraci aktivního prvku. Stavový automat, můžeme vidět na Obr. 7.1. Tento automat vychází ze stavového automatu uzlu snmp_manager [7] kap. 5.2, proto se zde nebudu zabývat důsledným popisem implementace.



Obr. 7.1: Stavový automat modelu execMib

Počáteční stav INIT slouží pro inicializaci. Stav IDLE řídí časování mezi jednotlivými stavy. Ve stavu EXEC se bude nacházet výkonná část kódu, která se postará o načítání dat z tabulky DiffServMIB a následné nastavení politiky aktivního prvku.

Přechody mezi stavy jsou řízeny podmínkami STOP a EXEC_MIB. Tyto podmínky definujeme v Header bloku. Stavy Stop-Exec a EXEC jsou vynucené, a proto se ihned po vykonání kódu vrací zpět do stavu IDLE.

7.1 Header blok

V Header bloku definujeme podmínky přechodu a hodnoty přerušení mezi stavy (viz. Obr. 7.1). Pro podmínku STOP je definováno makro, které testuje, zda bylo vyvoláno přerušení a zda je kód přerušení GW_STOP_EXEC. Podobně také makro EXEC_MIB testuje, zda bylo vyvoláno přerušení a kód přerušení.

```
/* Definujeme hodnoty kodu preruseni            */
#define GW_START_EXEC         100
#define GW_EXEC               110
#define GW_STOP_EXEC          120
#define GW_DISABLED_EXEC      130
#define GW_INFINITE_TIME      -100

/*Definice podminek pro prechody mezi stavy */
#define STOP                  (op_intrpt_type() == OPC_INTRPT_SELF && \
                                      intrpt_code == GW_STOP_EXEC)

#define EXEC_MIB  (op_intrpt_type() == OPC_INTRPT_SELF && \
                                      intrpt_code == GW_EXEC)
```

Tabulka 7.1: Header blok

7.2 Stav INIT

Ve vstupní části stavu INIT zkontrolujeme nastavení časování. Nejdříve načteme hodnoty atributů (funkce *op_ima_obj_attr_get*) a uložíme je do příslušných proměnných. Podmínka *if ((stop_time <= start_time) && (stop_time != GW_INFINITE_TIME))* kontroluje zda čas startu není nastaven na později, než konec. Jestliže je podmínka splněna, uloží se do proměnné start_time hodnota konstanty GW_INFINITE_TIME.

```
/* Vyčteme hodnoty atributů */
op_ima_obj_attr_get (my_obj_id, "Update interval", &next_exec_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);


/* Zkontroluje zda je korektne nastaveno casovani */
if ((stop_time <= start_time) && (stop_time != GW_INFINITE_TIME))
      {
      /* Stop time je menší jak start time. Nastaví start_time na -130
      */
      start_time = GW_INFINITE_TIME;

      /* Display an appropriate warning.                    */
      op_prg_odb_print_major ("Varování: Start time nastaven na mensi
      hodnotu nez stop time", OPC_NIL);
      }
```

Tabulka 7.2: Stav INIT, vstupní část

Výstupní část stavu INIT nastaví pouze kód přerušení. Pokud je hodnota proměnné start_time nastavena na hodnotu konstanty GW_INFINITE_TIME v důsledku nekorektního nastavení doby startu, nastaví se kód přerušení na GW_DISABLED_EXEC. Tato hodnota zajistí přechod do stavu Stop-Exec, ve kterém se zruší následující naplánované přerušení a po navrácení do stavu IDLE zde proces setrvá do konce simulace.

```
if ( start_time == GW_INFINITE_TIME)
      {
       intrpt_code = GW_DISABLED_EXEC;
      }
else intrpt_code = GW_START_EXEC;
```

Tabulka 7.3: Stav INIT, výstupní část

7.3 Stav IDLE

Vstupní část stavu IDLE plní funkci časovače a řídí přechody mezi stavy. První podmínka kontroluje, zda je kód přerušení nastaven na GW_START_EXEC, nebo na GW_EXEC. Kód přerušení GW_START_EXEC značí, že se jedná o přechod ze stavu INIT, v tomto případě naplánujeme přerušení pro začátek a konec časování a zároveň zkontrolujeme, zda není časový interval next_exec_time nastaven do záporných hodnot. Plánované přerušení nastavíme funkcí **op_intrpt_schedule_self**. Parametry funkce jsou čas, kdy bude přerušení vyvoláno a kód přerušení.

```
if (intrpt_code == GW_START_EXEC || intrpt_code == GW_EXEC)
      {
      /* nastaveni kodu preruseni pro prvni prechod do stavu Exec_Mib*/
      /* v case start_time                                          */
      if (intrpt_code == GW_START_EXEC)
            {
            op_intrpt_schedule_self (start_time, GW_EXEC);

      /* nastaveni kodu preruseni v case stop_time*/
            if (stop_time != GW_INFINITE_TIME)
                  {
                  op_intrpt_schedule_self (stop_time, GW_STOP_EXEC);
                  }

      /*kontrola promenne next_exec_time, pokud je mensi nez nula, nstavi
      hodnotu 30 vterin                                            */
            if (next_exec_time <=0)
                  {
                  next_exec_time = 30.0;
                  }

            }

      else next_exec_evh = op_intrpt_schedule_self (op_sim_time () +
      next_exec_time, GW_EXEC);
      }
```

Tabulka 7.4: Stav IDLE

op_intrpt_schedule_self ()

Naplánuje přerušení pro proces

Syntaxe:

op_intrpt_schedule_self (time, code)

Argument	Typ	Popis
time	double	Čas pro přerušení
code	Int	představuje kód přerušení

Tabulka 7.5: Parametry funkce op_intrpt_schedule_self

7.4 Stav Stop-Exec

Do tohoto stavu se proces dostane, pokud je kód přerušení nastaven na GW_STOP_EXEC. Tento stav zruší následující naplánované přerušení a nastaví kód přerušení **intrpt_code** na GW_DISABLED_EXEC. Po návratu do stavu Idle zde proces zůstane do konce simulace.

```
if (op_ev_valid (next_exec_evh) == OPC_TRUE)
      {
      op_ev_cancel (next_exec_evh);
      intrpt_code = GW_DISABLED_EXEC;
      printf("\n Casování zastaveno v case: %f\n",op_sim_time());
      }
```

Tabulka 7.6: Stav Stop-Exec

7.5 Stav Exec

V tomto stavu se bude nacházet výkonná část kódu, která zajistí načtení parametrů z tabulky MIB a po provedení příslušných funkcí nakonfiguruje směrovač dle konfigurace DiffServMIB. Pro načtení parametrů je nutné přistupovat k tabulce DiffSerMIB v paměti, která se vytváří dynamicky v uzlu snmp-agent Abychom mohli k této tabulce přistupovat i z jiného uzlu, použijeme parametr *extern*. Definici ukazuje Tabulka 7.7 .

```
extern MibTree *SnmpMibTree;
```

Tabulka 7.7: definice struktury SnmpMibTree

Funkce pro ovládání QoS jsou v Opnetu již implementovány. Jejich definice se nachází v externích hlavičkových souborech. S těmito funkcemi se dále pracuje v uzlu IP, tedy na síťové vrstvě. Zde se postupně zjišťuje jaké mechanismy diferencovaných služeb jsou nastaveny a tyto jsou postupně aplikovány na jednotlivá rozhraní.

8 Závěr

Předložená diplomová práce je zaměřena na implementaci protokolu SNMP do simulačního prostředí Opnet Modeler. Pro implementaci byla zvolena tabulka DiffServMIB, která popisuje jednotlivé komponenty technologie rozlišovaných služeb. Po prozkoumání databázové struktury tabulky MIB byl navržen model, který popisuje diferencované zpracování dat ve vstupním a výstupním směru na rozhraní směrovače. Model je popsán pomocí funkčních elementů, které jsou reprezentovány jednotlivými tabulkami DiffServMIB, které jsou mezi sebou vhodně provázány. Podstatná část práce byla věnována systému atributů v prostředí Opnet Modeler, které pro svou přehlednost a snadnou konfigurovatelnost slouží pro reprezentaci tabulek DiffServMIB. Podle parametrů DiffServMIB by se měly řídit aktivní prvky v síti a nastavit příslušné politiky v obou směrech. Byly vypracovány postupy pro konfiguraci pomocí systému atributů, které lze použít pro statickou konfiguraci modelu síťového prvku. Dynamická konfigurace není možná z důvodu nemožnosti přidávat nové položky atributů během simulace. Tyto atributy proto slouží pro počáteční inicializaci a naplnění dynamické struktury v paměti počítače. S touto strukturou se dále pracuje a na základě nastavení tabulky MIB se prováže toto nastavení s konfigurací směrovače. Byl navržen model, který v určitém časovém intervalu bude kontrolovat strukturu DiffServMIB, číst parametry jednotlivých tabulek a konfigurovat aktivní prvek.

Literatura:

[1] MCCLOGHRIE K., PERKINS D., SCHOENWAELDER J., *Structure of Management Information Version 2 (SMIv2)* – RFC2578, [online], Internet Engineering Task Force. Dostupný z WWW:<http://www.ietf.org/rfc/rfc2578.txt?number=2578>

[2] BAKER F., CHAN K., SMITH A., *Management Information Base for the Differentiated Services Architecture* – RFC3289, [online], Internet Engineering Task Force. Dostupný z WWW: <http://www.ietf.org/rfc/rfc3289.txt?number=3289>

[3] HEINANEN J., GUERIN R., *A Single Rate Three Color Marker* – RFC2697 [online], Internet Engineering Task Force. Dostupný z WWW: <http://www.ietf.org/rfc/rfc2697.txt?number=2697>

[4] HEINANEN J., GUERIN R., *A Two Rate Three Color Marker* – RFC2698 [online], Internet Engineering Task Force. Dostupný z WWW: <http://www.ietf.org/rfc/rfc2698.txt?number=2698>

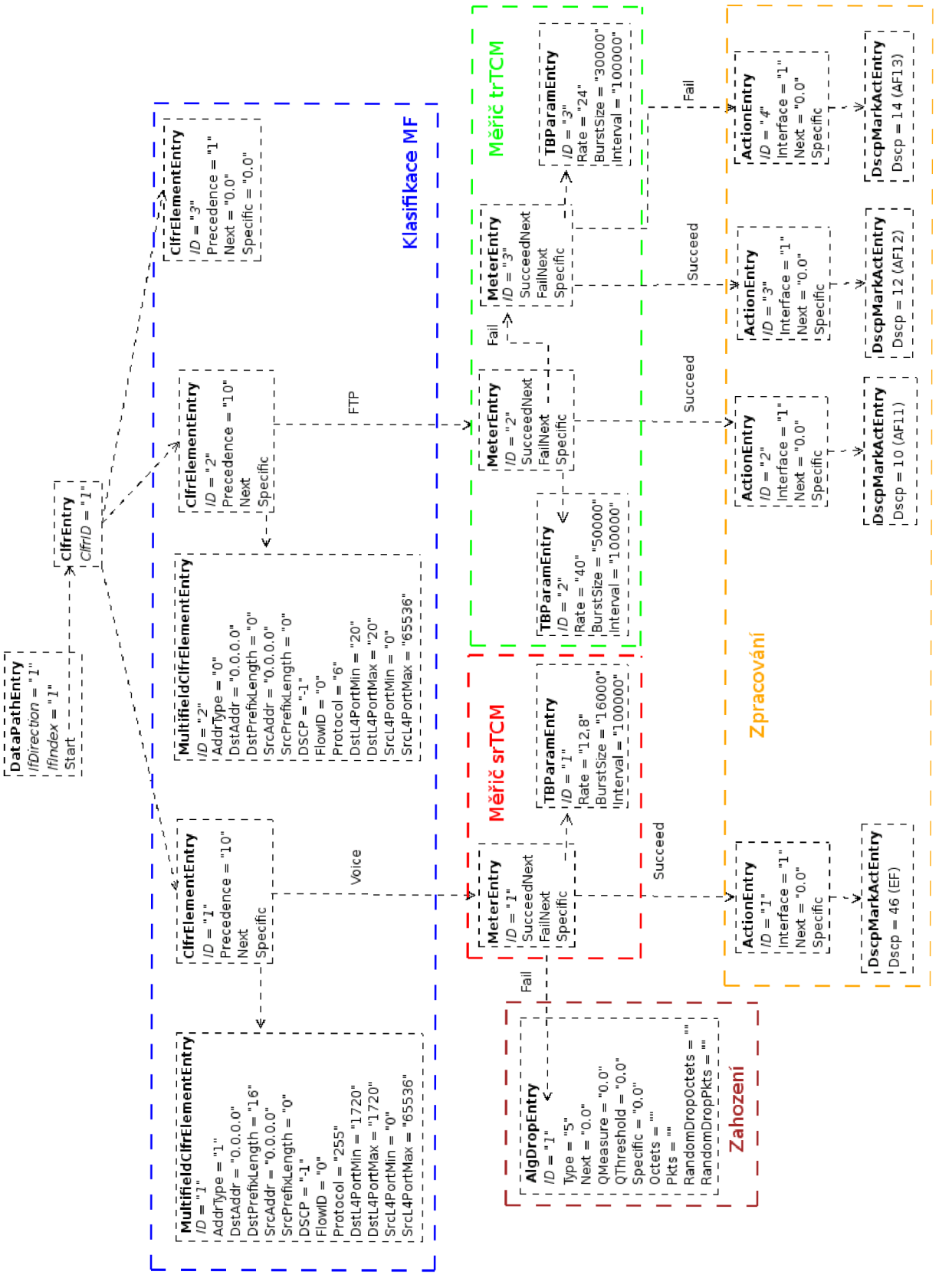
[5] OPNET TECHNOLOGIES, INC.: *OPNET Modeler Product Documentation Release 14.5*, OPNET Modeler, 2008

[6] BARTOŠ P., *Detailní analýza způsobu zacházení assured forwarding v simulačním prostředí Opnet Modeler:* diplomová práce. Brno: VUT fakulta elektrotechniky a komunikačních technologií, 2007.

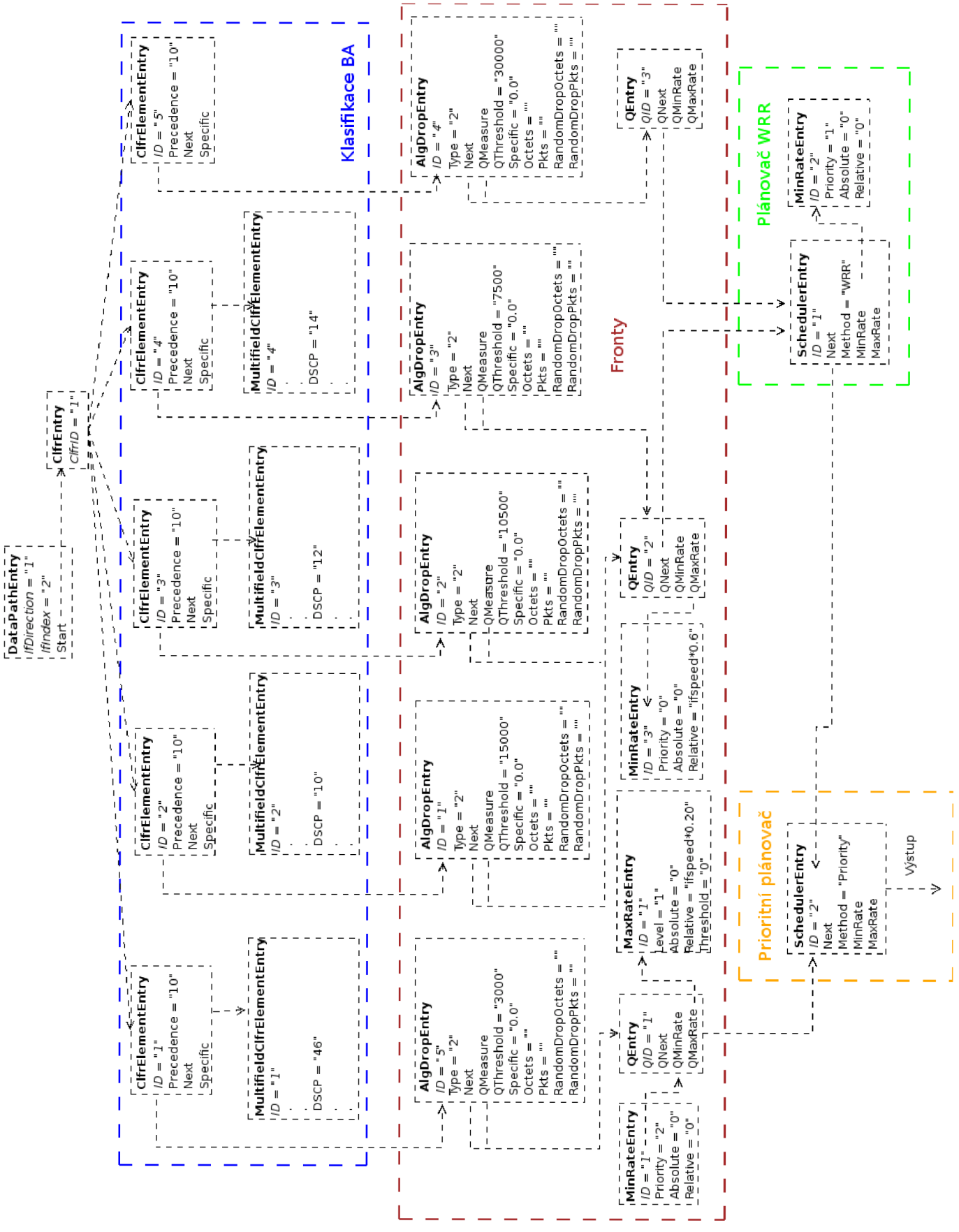
[7] VLČEK M., *Implementace a optimalizace univerzálního aplikačního protokolu v simulačním prostředí Opnet Modeler:* semestrální projekt. Brno: VUT fakulta elektrotechniky a komunikačních technologií, 2007.

Seznam příloh:

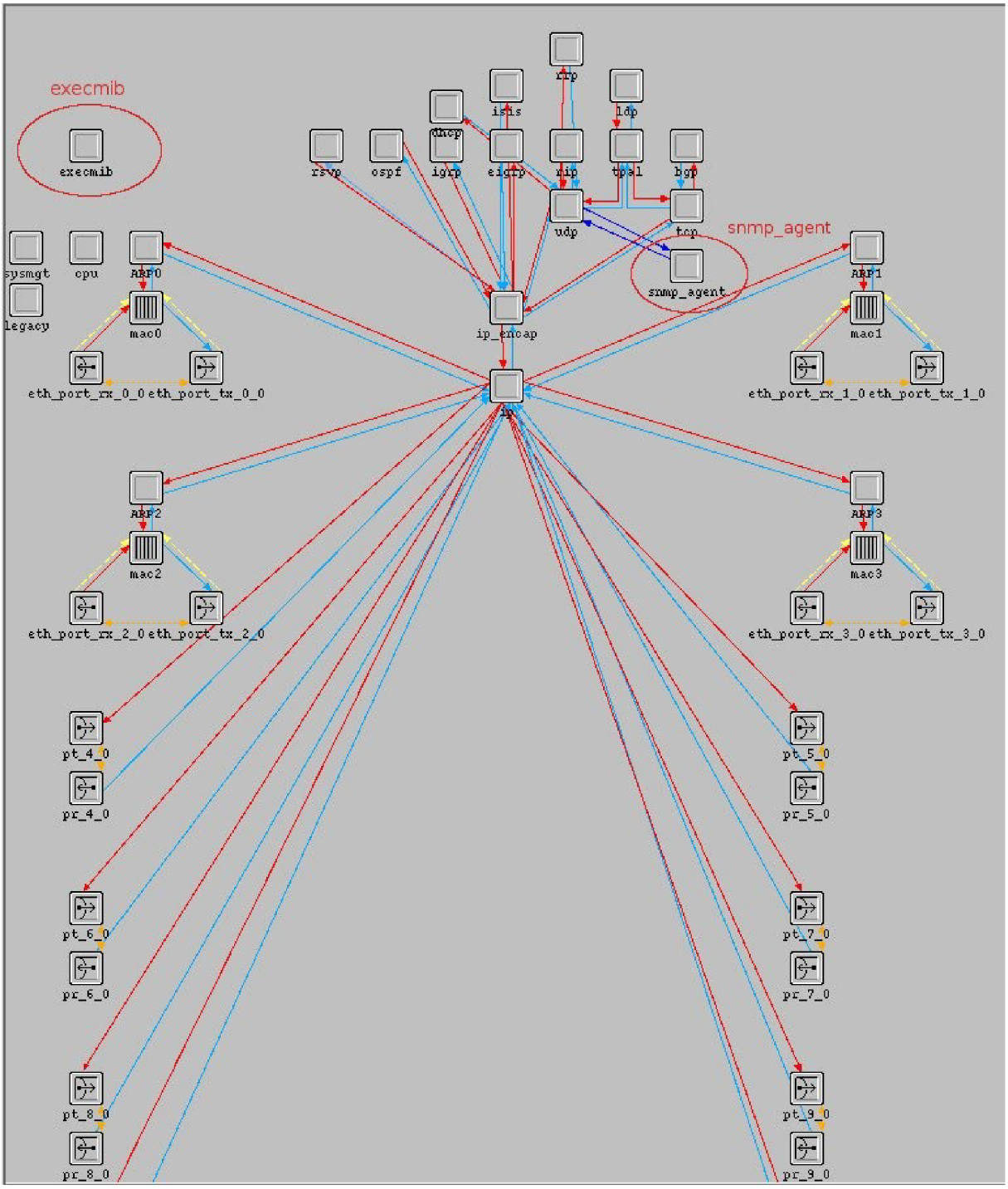
Příloha 1:


DataPathEntry
IfDirection = "1"
IfIndex = "1"
Start
ClfrEntry
ClfrID = "1"
Klasifikace MF
MultifieldClfrElementEntry
ID = "1"
AddrType = "1"
DstAddr = "0.0.0.0"
DstPrefixLength = "16"
SrcAddr = "0.0.0.0"
SrcPrefixLength = "0"
DSCP = "-1"
FlowID = "0"
Protocol = "255"
DstL4PortMin = "1720"
DstL4PortMax = "1720"
SrcL4PortMin = "0"
SrcL4PortMax = "65536"
ClfrElementEntry
ID = "1"
Precedence = "10"
Next
Specific
Voice
MultifieldClfrElementEntry
ID = "2"
AddrType = "0"
DstAddr = "0.0.0.0"
DstPrefixLength = "0"
SrcAddr = "0.0.0.0"
SrcPrefixLength = "0"
DSCP = "-1"
FlowID = "0"
Protocol = "6"
DstL4PortMin = "20"
DstL4PortMax = "20"
SrcL4PortMin = "0"
SrcL4PortMax = "65536"
ClfrElementEntry
ID = "2"
Precedence = "10"
Next
Specific
FTP
ClfrElementEntry
ID = "3"
Precedence = "1"
Next = "0.0"
Specific = "0.0"
Měřič srTCM
MeterEntry
ID = "1"
SucceedNext
FailNext
Specific
TBParamEntry
ID = "1"
Rate = "12.8"
BurstSize = "16000"
Interval = "100000"
Succeed
Fail
MeterEntry
ID = "2"
SucceedNext
FailNext
Specific
TBParamEntry
ID = "2"
Rate = "40"
BurstSize = "50000"
Interval = "100000"
Succeed
Fail
Měřič trTCM
MeterEntry
ID = "3"
SucceedNext
FailNext
Specific
TBParamEntry
ID = "3"
Rate = "24"
BurstSize = "30000"
Interval = "100000"
Succeed
Fail
Zahození
AlgDropEntry
ID = "1"
Type = "5"
Next = "0.0"
QMeasure = "0.0"
QThreshold = "0.0"
Specific = "0.0"
Octets = ""
Pkts = ""
RandomDropOctets = ""
RandomDropPkts = ""
Zpracování
ActionEntry
ID = "1"
Interface = "1"
Next = "0.0"
Specific
DscpMarkActEntry
Dscp = 46 (EF)
ActionEntry
ID = "2"
Interface = "1"
Next = "0.0"
Specific
DscpMarkActEntry
Dscp = 10 (AF11)
ActionEntry
ID = "3"
Interface = "1"
Next = "0.0"
Specific
DscpMarkActEntry
Dscp = 12 (AF12)
ActionEntry
ID = "4"
Interface = "1"
Next = "0.0"
Specific
DscpMarkActEntry
Dscp = 14 (AF13)

Příloha 2:


DataPathEntry
IfDirection = "1"
IfIndex = "2"
Start
ClfrEntry
ClfrID = "1"
Klasifikace BA
ClfrElementEntry
ID = "1"
Precedence = "10"
Next
Specific
MultifieldClfrElementEntry
ID = "1"
DSCP = "46"
ClfrElementEntry
ID = "2"
Precedence = "10"
Next
Specific
MultifieldClfrElementEntry
ID = "2"
DSCP = "10"
ClfrElementEntry
ID = "3"
Precedence = "10"
Next
Specific
MultifieldClfrElementEntry
ID = "3"
DSCP = "12"
ClfrElementEntry
ID = "4"
Precedence = "10"
Next
Specific
MultifieldClfrElementEntry
ID = "4"
DSCP = "14"
ClfrElementEntry
ID = "5"
Precedence = "10"
Next
Specific
Fronty
AlgDropEntry
ID = "5"
Type = "2"
Next
QMeasure
QThreshold = "3000"
Specific = "0.0"
Octets = ""
Pkts = ""
RandomDropOctets = ""
RandomDropPkts = ""
AlgDropEntry
ID = "1"
Type = "2"
Next
QMeasure
QThreshold = "15000"
Specific = "0.0"
Octets = ""
Pkts = ""
RandomDropOctets = ""
RandomDropPkts = ""
AlgDropEntry
ID = "2"
Type = "2"
Next
QMeasure
QThreshold = "10500"
Specific = "0.0"
Octets = ""
Pkts = ""
RandomDropOctets = ""
RandomDropPkts = ""
AlgDropEntry
ID = "3"
Type = "2"
Next
QMeasure
QThreshold = "7500"
Specific = "0.0"
Octets = ""
Pkts = ""
RandomDropOctets = ""
RandomDropPkts = ""
AlgDropEntry
ID = "4"
Type = "2"
Next
QMeasure
QThreshold = "30000"
Specific = "0.0"
Octets = ""
Pkts = ""
RandomDropOctets = ""
RandomDropPkts = ""
MinRateEntry
ID = "1"
Priority = "2"
Absolute = "0"
Relative = "0"
MaxRateEntry
ID = "1"
Level = "1"
Absolute = "0"
Relative = "ifspeed*0.20"
Threshold = "0"
MinRateEntry
ID = "3"
Priority = "0"
Absolute = "0"
Relative = "ifspeed*0.6"
QEntry
QID = "1"
QNext
QMinRate
QMaxRate
QEntry
QID = "2"
QNext
QMinRate
QMaxRate
QEntry
QID = "3"
QNext
QMinRate
QMaxRate
Plánovač WRR
SchedulerEntry
ID = "1"
Next
Method = "WRR"
MinRate
MaxRate
MinRateEntry
ID = "2"
Priority = "1"
Absolute = "0"
Relative = "0"
Prioritní plánovač
SchedulerEntry
ID = "2"
Next
Method = "Priority"
MinRate
MaxRate
Výstup

Příloha 3:


execmib
execmib
rtp
isis
ldp
dhcp
rsvp
ospf
igrp
eigrp
mip
tpal
bgp
udp
tcp
snmp_agent
snmp_agent
sysmgt
cpu
ARP0
legacy
mac0
ip_encap
ARP1
mac1
eth_port_rx_0_0
eth_port_tx_0_0
eth_port_rx_1_0
eth_port_tx_1_0
ip
ARP2
ARP3
mac2
mac3
eth_port_rx_2_0
eth_port_tx_2_0
eth_port_rx_3_0
eth_port_tx_3_0
pt_4_0
pr_4_0
pt_5_0
pr_5_0
pt_6_0
pr_6_0
pt_7_0
pr_7_0
pt_8_0
pr_8_0
pt_9_0
pr_9_0