

Technická univerzita v Košiciach  
Fakulta elektrotechniky a informatiky  
Katedra počítačov a informatiky

## **Opis sieťových protokolov prostredníctvom jazyka XML**

Diplomová práca

**Študijný odbor:** Výpočtová technika a informatika

Vedúci diplomovej práce:  
Ing. Ján Genči

Diplomant:  
Ľuboš Koščo

Konzultant diplomovej práce:  
Ing. Ján Genči

Košice 2005

### **Čestné prehlásenie**

Prehlasujem, že som diplomovú prácu vypracoval samostatne s využitím uvedenej odbornej literatúry.

V Košiciach dňa 5.5.2005

.....  
*vlastnoručný podpis*

Na tomto mieste bude vložené zadanie diplomovej práce

## **Pod'akovanie**

Chcel by som sa poďakovať vedúcemu diplomovej práce Ing. Jánovi Genčimu a Ing. Jurajovi Giertlovi za ich cenné pripomienky a rady a odbornú pomoc pri vypracovaní diplomovej práce. Moja vďaka patrí tiež teamu projektu QoS@Lab a celému Laboratóriu počítačových sietí na KPI FEI Technickej Univerzity v Košiciach.

Názov práce : Opis sieťových protokolov prostredníctvom jazyka XML

Katedra : Katedra počítačov a informatiky, TU FEI Košice

Autor : Ľuboš Koščo

Vedúci DP : Ing. Ján Genči

Konzultant DP : Ing. Ján Genči

Dátum : 5.5.2005

Kľúčové slová : XML, XML schema, sieťové protokoly, Java, Netflow, QoS merania

Anotácia : Táto práca sa zaoberá štandardizáciou sieťových protokolov, snaží sa o návrh zdieľaného slovníka pre vyjadrenie pravidiel daných ľuďmi, resp. opisu sieťových protokolov vykonateľného strojom. Základnými nástrojmi sú XML, XML schéma pre vytvorenie opisného jazyka pre sieťové správy, programovací jazyk Java ako jazyk praktickej implementácie. Ukážkové praktické využitie daného opisu predstavuje projekt JXColl, ktorý umožňuje zachytávať sieťové správy (UDP), ktoré sú definované opisom (pre protokoly Cisco Netflow 5 a Netflow 9) a spracovávať ich (výstup do databázy, textového súboru (log), priame pripojenie na frontend aplikáciu).

Thesis title : Description of network protocols using XML language

Department : Department of Computers and Informatics, TU FEI Košice

Author : Ľuboš Koščo

Supervisor : Ing. Ján Genči

Tutor : Ing. Ján Genči

Date : 5.5.2005

Keywords : XML, XML schema, network protocols, Java, Netflow, QoS measurements

Annotation : This thesis is dealing with standardization of network protocols. It is trying to propose a shared vocabulary for representation of rules made by people, aka network protocol description carried out by machine. Basic tools are XML, XML schema for developing a language for network packets, programming language Java in practical implementation. Example of practical use of this description is represented by project JXColl, which is able to listen to network packets (UDP), which are defined by their description (for Cisco protocols Netflow 5 and Netflow 9) and it can process them (output to database, text file (log), direct connection to frontend application).

## Obsah

Úvod .....	1
1. Formulácia úlohy .....	2
2. Analýza modelovania a štandardizácie sieťových protokolov .....	3
2.1 Sieťové protokoly.....	3
2.2 Súčasný stav opisu a štandardizácie sieťových protokolov.....	5
2.3 Vízia. ....	7
3. Implementácia .....	14
3.1 Existujúce nástroje a implementácie .....	14
3.2 Použitie XML ako opisného jazyka .....	15
3.3 Praktické využitie pri meraní kvality služieb v sieťach v rámci IPFIX. ....	17
3.4 Návrh, koncepcia a architektúra JXColl ako aplikácie využívajúcej opis pre strojové spracovanie protokolu.....	23
3.5 Príklad použitia opisu v XML pre NF5, NF9 a TCP/IP.....	35
4. Zhodnotenie riešenia .....	40
5. Zoznam použitej literatúry .....	41
6. Zoznam príloh.....	44
7. Zoznam obrázkov a tabuliek .....	44

## Úvod

Diplomová práca obsahuje štúdium štruktúry sieťových protokolov, rôzne spôsoby ich zápisu a štandardizácie. Navrhuje použitie XML schémy a XML ako opisného jazyka pre štruktúru sieťového protokolu. Prakticky testuje tento prístup pre rôzne verzie protokolu Netflow. Protokol Netflow [Netflow] bol vyvinutý a je používaný firmou Cisco na prenos informácií o dátových tokoch – flowoch. Práca sa teda zaoberá vývojom a praktickým využitím formálnych prístupov pri opise štruktúry dát.

V práci nájdeme koncepciu, návrh a implementáciu programového vybavenia, ktoré je schopné rozpoznávať a spracovávať dáta posielané po sieti na úrovni transportnej až prezentačnej vrstvy OSI modelu. Vstupom sú dáta, ktoré sa rozpoznávajú na základe opisu v XML schéme. Výstupom je archivácia dát do databázy, textového súboru alebo priame preposlanie nameraných dát do zobrazujúcej časti (front-end aplikácie). Tento softvér nadväzuje na architektúru IPFIX [IPFIX] a zodpovedá príslušným štandardom. Ako taký je princíp jeho funkčnosti založený na zachytávaní dát z meraní tokov na sieti a ich následné spracovanie, uchovanie, distribúciu alebo preposlanie na priame spracovanie.

Práca je štruktúrovaná do viacerých kapitol. Prvá sa venuje formulácii úlohy a jej rozdelenie na podúlohy. Druhá je teoretickou rozpravou o formálnom prístupe k modelovaniu sieťových protokolov. Naznačuje tiež riešenie problému modelovania stavov v prípade stavového protokolu, kde je iba opis štruktúry dát nepostačujúci (pre plný súlad v zmysle názvu práce). Opisuje sa momentálny stav systému štandardizácie sieťových protokolov a jeho vylepšenie použitím opisu štruktúry dát pomocou XML. Opisuje sa XML a XML schéma ako také. Ďalšia kapitola sa venuje existujúcim návrhom založeným na podobnom princípe, presnejšie opisuje samotný opis použitím XML a uvádza výhody a nevýhody takéhoto prístupu. Práca sa postupne presúva k úvodu do implementačnej časti, ktorú tvorí vysvetlenie technológií a štandardov použitých pri vývoji softvéru. Nasleduje porovnanie a ukážka existujúcich nástrojov slúžiacich podobnému účelu a následne implementácia samotného nástroja. Záver tvoria ukážky opisov štruktúr protokolov Netflow 5, Netflow 9 a TCP/IP a samotné zhodnotenie riešenia.

## 1. Formulácia úlohy

Cieľom diplomovej práce je implementovať a ukázať možné využitie formálnych metód pri opise protokolov. Ukázať, že takýto prístup je vhodný a že ho vieme využiť napr. aj pri podmienkach vysokej záťaže akým je architektúra pre pasívne merania podľa štandardov IPFIX-u, ktorá sa experimentálne implementuje v laboratóriu počítačových sietí na Katedre počítačov a informatiky Technickej Univerzity v Košiciach. [CNLTUKE]

V rámci tejto úlohy je vhodné definovať tieto podúlohy:

- analýza rôznych prístupov k opisom sieťových protokolov
- výber, resp. zhodnotenie vybraného vhodného formalizmu pre opis (XML)
- analýza a návrh nástroja schopného spracovania dát na základe takéhoto opisu, jeho využitie pri spracovaní informácií o meraniach v sieti pri využití technológií Netflow
- porovnanie s existujúcimi nástrojmi, zhodnotenie výkonnosti takéhoto nástroja
- ukážky opisov štruktúr protokolov TCP/IP, Netflow
- overenie funkčnosti nástroja
- zdokumentovanie analýzy a implementácie softvéru

---

## 2. Analýza modelovania a štandardizácie sieťových protokolov

Väčšina činností a komunikácie sa riadi podľa pravidiel, tieto pravidlá by sa dali zhrnúť pod pojmom protokol. Protokol je formálny opis množiny pravidiel a konvencií, ktoré riadia príslušný aspekt komunikácie zariadení na sieti. Protokoly stanovujú formát, časovanie, radenie a chybovú kontrolu v dátovej komunikácii. Bez protokolov by počítač nemohol zostaviť alebo obnoviť prúd prichádzajúcich bitov z iného počítača do pôvodného formátu. Sady protokolov umožňujú sieťovú komunikáciu z jedného hostiteľa na iného cez sieť [CCNA1]. Tieto pravidlá by mali spĺňať účel jednoznačného opisu vzájomnej komunikácie. Značne ale záleží na tom, ako majú tento protokol implementované komunikujúce strany. Je vhodné, aby nedochádzalo k nesprávnej interpretácii a snahy posledného obdobia vedú k tomu, aby takýto protokol bol interpretovateľný strojom, podľa pravidiel, ktoré sú rovnako platné a zrozumiteľné pre všetkých. Aký je stav?

### 2.1 Sieťové protokoly

Sieťový protokol je množina pravidiel nutných k tomu, aby mohli dva a viac výpočtových procesov vzájomne komunikovať. Tieto procesy môžu byť spúšťané na tom istom alebo na iných strojoch spojených rôznymi typmi sietí. Procesy môžu byť osobitné procesy operačného systému, bežiacie pod rôznymi programami alebo môžu byť virtuálnymi procesmi, modulárnymi časťami jedného programu alebo môžu byť súčasťami operačného systému. Kľúčové faktory sú, že musia existovať aspoň dva procesy a tie musia navzájom komunikovať. [McGuire04]

Výpočtové procesy vzájomne komunikujú výmenou dobre definovanými správami cez komunikačný kanál a povaha tejto výmeny definuje pravidlá, vytvárajúc sieťový protokol. Pozoruhodný aspekt týchto pravidiel je formát správ, ktoré si procesy vymieňajú. Závisí na ňom mnoho vecí (napr. zložitosť komunikácie) a ovplyvňuje priamo alebo nepriamo veľké množstvo faktorov a kvality sieťovej komunikácie (napr. náročnosť protokolu na šírku pásma – bandwidth, rýchlosť komunikácie, atď). Dôležité ale sú aj pravidlá výpočtu, ktorý musí každý z procesov vykonať, aby poslal korektnú správu obsahujúcu korektné hodnoty v korektný čas.

---

Pravidlá vytvárajúce formálnu špecifikáciu sieťového protokolu, formáty správy tak ako aj výpočty sú momentálne zakotvené v programoch. Programy teda implementujú špecifikácie. Návrh abstraktných pravidiel obsahujúcich tak štruktúru správ ako aj akcie je predmetom tejto práce. Prístup je teda iný a programy budú implementovať formalizmus, ktorý obsiahne sieťovú komunikáciu.

### **ISO OSI a TCP/IP Model**

OSI a TCP/IP modely majú vrstvy, ktoré vysvetľujú, akým spôsobom sa deje komunikácia z jedného počítača na druhý. Modely sa líšia v počte a funkciách jednotlivých vrstiev. Hociktorý z modelov nám však môže pomôcť pri opise a vie poskytnúť detaily o toku informácie zo zdroja do cieľa.

Paket – na informáciu, ktorá sa šíri sieťou sa všeobecne odkazujeme ako na dáta alebo balík (paket). Paket je logicky zoskupená jednotka informácie, ktorá sa pohybuje medzi počítačovými systémami. Ako dáta prechádzajú medzi rôznymi vrstvami, každá sieťová vrstva pridá dodatočné informácie, ktoré umožnia efektívnu komunikáciu s danou vrstvou na inom počítači.

Open System Interconnection (OSI) referenčný model bol uvoľnený v roku 1984 ako opisný sieťový model, vytvorený organizáciou ISO (International Standards Organization). Obsahuje 7 vrstiev – fyzickú, spojovaciu, sieťovú, transportnú, relačnú, prezentačnú a aplikačnú; každá z nich ilustruje príslušnú sieťovú funkciu. Takéto delenie má dosť výhod, okrem iného aj štandardizáciu sieťových komponentov za účelom vývoja a podpory takýchto zariadení viacerými spoločnosťami. Historickým a technickým štandardom Internetu je TCP/IP referenčný model, ktorý má oproti OSI iba 4 vrstvy – sieťovú, internetovú, transportnú a aplikačnú.

Na každej vrstve či OSI alebo TCP/IP modelu je protokol alebo súbor protokolov, ktoré určujú formát a pravidlá prenosu dát. Vrstva 4 na zdrojovom počítači komunikuje s vrstvou 4 na cieľovom počítači. Pravidlá a konvencie použité pre túto vrstvu sa nazývajú protokoly vrstvy 4.

### **Protokolový zásobník – ukážka vrstvenia protokolov**

Protokolový zásobník - pri sieťovej komunikácii sú protokoly vrstvené nad sebou, každá vrstva je zodpovedná za iný aspekt komunikácie. Inými slovami protokolový zásobník je vrstvená množina protokolov, ktoré spolupracujú za účelom poskytnúť množinu sieťových funkcií [RFC1983]. TCP/IP protokolový zásobník vyzerá

napríklad takto: HTTP Telnet POP3 SNMP bootp TCP UDP ICMP IP ARP PPP Ethernet. Použitie je takéto: Použije sa protokol HTTP na vyžiadanie web stránky. HTTP klient (prehliadač web stránok) kontaktuje HTTP server (miesto uloženia web stránky) pomocou protokolu TCP. Protokol TCP rozloží všetku svoju prácu do IP paketov (balíkov). Routre (smerovacie a transformovacie zariadenia) na internete vedia ako preposielať IP pakety. Použije sa napr. PPP alebo Ethernet aby sa poslali IP pakety k najbližšiemu routru alebo inému sieťovému prvku.

## **2.2 Súčasný stav opisu a štandardizácie sieťových protokolov.**

Internet Engineering Task Force (IETF) je štandardizačný orgán, ktorý nesie najväčšiu zodpovednosť za tvorbu Internetových štandardov. Pracovné skupiny IETF pod dohľadom Internet Engineering Steering Group (IESG) a Internet Architecture Board (IAB) nepretržite vyvíjajú nové protokoly a technológie a tieto vývojové trendy sú formalizované v RFC dokumentoch. [TCPIPG]

Proces vytvárania a publikovania Internetového štandardu je zaujímavý, ale zdĺhavý, a niektoré časti by sa dali zefektívniť. Každý IETF štandard je publikovaný ako RFC (Request For Comments) a každý RFC začína ako Internet Draft (Internetový koncept) (často nazývaný "I-D") [IETF]. Základné kroky ako publikovať niečo ako IETF štandard sú: publikovať dokument ako Internet Draft; zozbierať komentáre (pripomienky) k draftu; upraviť svoj draft na základe komentárov; zopakovať doterajšie kroky, kým nebude draft všeobecne uznaný; požiadať Area Director-a (AD) aby zbral draft do IESG (ak je to individuálny príspevok). Ak draft je oficiálny produkt skupiny, tak predstaviteľ pracovnej skupiny poprosí AD aby ho zbral do IESG; urobiť zmeny, ktoré uvážila IESG za vhodné (tieto môžu obsahovať aj to, že dokument nie je vhodný stať sa štandardom); počkať kým dokument bude publikovaný RFC editorom

K týmto krokom snáď len pár poznámok. Ak hodnotený I-D bol zvažovaný ako hodnotný, dobre zrozumiteľný a stabilný (čo znamená, že nie je rapídne aktualizovaný novými verziami) môže sa stať kandidátom na štandardizáciu. IESG môže umiestniť draft do dráhy pre Internetové štandardy zmenou jeho stavu na navrhovaný štandard - Proposed Standard. Akonáhle je špecifikácia dostatočne vypracovaná a široko akceptovaná, môže byť povzdvihnutá na konceptuálny štandard - Draft Standard.

---

Kľúčovou požiadavkou pre takýto postup je, že technológia musí byť demonštrovaná ako funkčná aspoň na dvoch navzájom nezávislých implementáciách schopných spolupracovať. To dokazuje fakt, že štandard je dostatočne jasný a kompletný, keďže ho aspoň dve rôzne skupiny dokázali kompatibilne implementovať. “Cieľová stanica” na dráhe Internetových štandardov je stav Internetového štandardu - Internet Standard. Toto označenie je aplikované iba na veľmi zrelé špecifikácie, ktoré sú populárne a všeobecne implementované. Treba pripomenúť, že publikované Internetové štandardy sú nemenné, aby nedošlo k chaosu spôsobeného viacerými verziami toho istého štandardu.

Viac komplexné vysvetlenie týchto krokov je obsiahnuté v BCP 9, “The Internet Standards Process” (<http://www.ietf.org/rfc/rfc2026.txt>). Je šesť typov RFC:

Navrhované štandardy - Proposed Standard

Konceptuálne štandardy - Draft Standard

Internetové štandardy (niekedy volané aj “plné štandardy”) - Internet Standard

Experimentálne protokoly - Experimental

Informačné dokumenty - Informational

Historické štandardy - Historic

Iba **prvé tri** sú štandardami v rámci IETF. Dobrý súhrn tohto môžete nájsť v RFC 1796, “Nie všetky RFC sú štandardy” (<http://www.ietf.org/rfc/rfc1796.txt>).

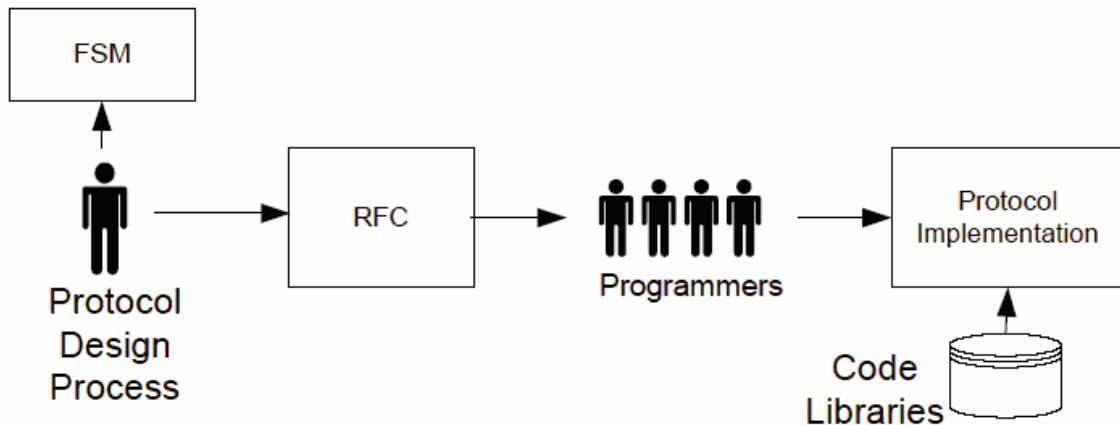
### 2.2.1 Čo sú RFC?

- požiadavka na komentáre (Request for comments) - oficiálne špecifikačné dokumenty zo sady Internetových Protokolov, ktoré sú definované IETF (Internet Engineering Task Force) a IESG (Internet Engineering Steering Group) sú zaznamenávané a publikované ako štandardné RFC. Ako výsledok publikačný proces RFC zohráva dôležitú úlohu v procese Internetových štandardov. RFC musia byť najprv publikované ako Internetové Drafty. [RFC]

### 2.2.2 Zhodnotenie analýzy

Manuálna produkcia protokolu vyzerá ako na obr. 2.1. Dizajnér protokolu na základe konečno stavového automatu navrhne protokol a ten, ak sa stane štandardom, je vydaný ako RFC. Medzičasom je implementovaný a je uvedený do praxe. Zvyčajne tento postup trvá roky a zvykne mať viacero iterácií v niektorých blokoch obrázku. Celý

postup by sa dal podstatne skrátiť a zjednodušiť práve použitím formalizmov a štandardného jazyka pre vývoj, ktorý sa dá zároveň použiť pri implementácii a tým pádom by sa mohla väčšia snaha venovať práve samotnému návrhu a dizajnu protokolov, zvyšok by sa dal viac, či menej, automatizovať.



**Obr. 2.1: Manuálna produkcia protokolu**

Vysvetlenie textov v obrázku:

- FSM = Finite State Machine – konečno stavový automat
- Protocol Design Process - proces návrhu protokolu
- Programmers - programátori
- Protocol Implementation – implementácia protokolu
- Code Libraries – knižnice kódu

Ako je to teda so štandardmi?

Nanešťastie, štandardné dokumenty sú často rozsiahle a nepresné. Často jediné formálne špecifikácie protokolu sú jeho implementácie, ktoré zväčša nie sú identifikované v štandarde, mnohokrát nie sú k dispozícii na nahliadnutie a sami o sebe majú ďaleko k stručnosti a zrozumiteľnosti. V prípade sieťových protokolov modularizácia skoro vždy zahŕňa vrstvenie.

### 2.3 Vízia návrhu opisu sieťových protokolov

Zrýchlenie a automatizácia vo finálnej implementačnej fáze tohto procesu je jedným z cieľov diplomovej práce.

### 2.3.1 Formálne modelovanie

Formálne modelovanie nás posunulo smerom k precíznejším a konkrétnejším špecifikáciám protokolov. Aj veľmi abstraktný formálny model vyžaduje vymenovanie – opis, čo **presne** sa myslí pod každou **štruktúrou a akciou protokolu**. Výsledkom je špecifikácia, ktorá je síce menej abstraktná, ale viac jasnejšia ako tradičné špecifikácie architektúr aj protokolov.

Formálne modelovanie sa tiež ukázalo ako skvelá pomoc dizajnérom zreteľne vyjadriť ich myšlienky. Niekedy protokol, ktorý je práve vyvíjaný, nie je tak komplikovaný ako koncový produkt, no môže byť zložité pre architekta zachytiť svoje myšlienky bez dvojzmyslov a stanoviť ich životaschopnosť, ak ich vyjadri iba v angličtine (ako štandard). Nejaký druh vysoko úrovňového modelovania je v tomto prípade potrebný na vyjadrenie „jednoznačnosti“, ktorá značne pomáha architektovi pochopiť povahu jeho problému.

Napr. jedna z užitočných abstrakcií [Gouda1998] pre návrh protokolov sú vysoko úrovňové, komplexné akcie, ktoré sú atomické celým protokolom – t.j. akcia, ktorá môže byť spustená v jednom procese bez interakcie s inými akciami v tom istom alebo iných procesoch. Táto abstrakcia robí návrh jednoduchším tým, že sa vyhýba otázkam simultánnosti a synchronizácie medzi rôznymi akciami. Vysoko úrovňová atomistika akcií tiež znižuje počet stavov protokolu, ktoré treba preveriť na zaistenie správnosti protokolu. Nanešťastie táto abstrakcia je veľmi náročná na implementáciu, keďže si vyžaduje globálnu synchronizáciu a môže viesť k neefektívnym implementáciám. [McGuire04]

### 2.3.2 Formálne metódy v protokolovom inžinierstve

Aj napriek tomu, že môže zrýchliť proces vývoja a zvýšiť spoľahlivosť finálnej implementácie, formálny prístup ku komunikačným protokolom nie je veľmi rozšírený ani uznávaný. Jedna z hlavných námietok pre prijatie formálnych metód v priemyselnom svete je, že ich nesporné výhody sú na druhej strane vyrovnané vysokými nákladmi na čas a zdroje potrebné na oboznámenie používateľov s touto technikou. V skutočnosti učenie sa špecifikačného jazyka je také zložité ako naučenie sa programovacieho jazyka, pretože počet syntaktických operátorov je podobný avšak nie je potrebná zvláštna matematická príprava. [Babich]

### 2.3.3 XML

Technológie XML sú použité v rôznych sférach a pomaly sa začínajú využívať aj pri štandardizácii [XMLRFC99] alebo ako protokol [XMLP]. Používať XML ako modelovací jazyk nie je nová záležitosť, stačí spomenúť napr. VRML. Pri samotnom návrhu sieťových protokolov sa však ešte len začínajú objavovať prvé zmienky o ich využití, vid' [GXML03]. Za zmienku stojí aj to, že veľa dizajnérov Internetových protokolov zvažuje použitie XML práve na opis dát a boli zverejnené direktívy pre použitie XML v rámci sieťových protokolov ako [RFC3470].

#### **XML Schema, význam schém**

Primárny význam schém leží v ich použití pre formálnu definíciu značkovacích jazykov, resp. dátových formátov, založených na XML. Výhoda formalizovanej definície spočíva v tom, že je jednoznačná a znemožňuje rôzne interpretácie, na rozdiel od definície popísanej v prirodzenom jazyku. Všetci, ktorí chcú dokumenty XML zodpovedajúce danej schéme spracovávať, preto vedia, ako môžu dokumenty vyzeráť.

Vzhľadom k tomu, že XML schéma jednoznačne definuje, ako môže dokument XML vyzeráť, môžeme ju samozrejme použiť i pre validáciu. Viac menej sa jedná asi o najčastejšie použitie schém. Validácia je proces, pri ktorom sa overuje, či nejaký konkrétny dokument XML vyhovuje všetkým obmedzeniam definovaným v schéme. Môžeme si tak napríklad overiť, či dokument, ktorý nám niekto poslal, syntakticky vyhovuje formátu, na ktorom sme sa predtým dohodli. Ďalšou výhodou validácie je uľahčenie vývoja aplikácie. Tá nemusí pri čítaní dát z XML prevádzať kontrolu správneho vstupu, pretože väčšinu prípadných chýb odhalí validácia XML dokumentu.

Schéma dokumentu môže byť využitá ako zdroj pre vytváranie zodpovedajúceho objektového modelu vrátane kódu, ktorý ho implementuje. Princíp data-bindingu spočíva v tom, že s pomocou automaticky vygenerovaného kódu zo schémy môžeme potom v aplikácii jednoducho manipulovať s dokumentmi XML, ktoré budú reprezentované ako sada objektov v pamäti.

V neposlednej rade slúži schéma ako dokumentácia pre značkovací jazyk, ktorý definuje. Navyše pre mnoho jazykov pre opis schém existujú nástroje, ktoré z pôvodnej schémy vygenerujú prehľadnú dokumentáciu napríklad v podobe hypertextovo previazaných webových stránok. [KosekXML04]

## Použitie XML a XML schémy

XML schéma umožňuje definovať elementy a atribúty použiteľné v XML dokumente, prípustné možnosti kombinovania jednotlivých elementov, dátový typ pre obsah elementu/atribútu a ďalšie integritné obmedzenia. Prínosy použitia XML schémy sú hlavne v tom, že schéma je formálnou definíciou jazyka (výmenného formátu) založeného na XML. Ďalej vieme dokument XML kedykoľvek behom jeho životného cyklu validovať (overenie zhody dokumentu so schémou). Táto validácia výrazne zjednodušuje kontroly vstupu na úrovni aplikácie a na základe nej vieme komfortnejšie zadávať dáta do editoru XML. A nakoniec schéma uľahčuje programovú manipuláciu s dokumentmi XML (PSVI, data-binding).

XML schéma nie je jedinou možnosťou, existuje viacero jazykov pre popis schémy XML dokumentu. Stručný prehľad jazykov pre popis schémy XML dokumentu:

- DTD - najstarší, vychádza ešte z SGML, priamo súčasť špecifikácie XML; nepodporuje menné priestory a dátové typy
- W3C XML Schema - podpora menných priestorov, dátových typov; pomerne zložitá špecifikácia; široká podpora komerčných firiem: MS, IBM, Oracle, Sun, ...
- Relax NG - nový a elegantný jazyk pre popis schém; podpora zatiaľ iba vo svete OSS; štandardizované v rámci OASIS a ISO
- Schematron - sada XPath výrazov, ktoré musí dokument spĺňať

XML schéma bola použitá práve pre najväčšiu podporu zo strany rôznych firiem, pre jej schopnosť väčšej abstrakcie (i napriek zložitosti) oproti ostatným jazykom, je zapisovaná vo formáte XML a má mnoho ďalších výhod.

## XML schema a jej vlastnosti

- je samo dokumentujúci formát a priamo súčasťou schémy môže byť dokumentácia, pomocou XSLT vieme potom generovať prehľadnú dokumentáciu schémy napr. v HTML

```
<xs:element name="zamestnanec">
  <xs:annotation>
    <xs:documentation>Element sluzi pre uchovanie dôležitých údajov
      o zamestnancovi.</xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:sequence>
      <xs:element name="meno" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

```
<xs:element name="priezvisko" type="xs:string"/>
<xs:element name="plat" type="xs:decimal"/>
<xs:element name="plat" type="xs:date"/>
</xs:sequence>
</xs:complexType>
</xs:element>
```

- má podporu pre dátové typy – vieme ich použiť pre obsah atribútov i elementov; má komplexné aj jednoduché typy: komplexné – obsahujú ďalšie elementy a atribúty; jednoduché – obsahujú iba jednu hodnotu (reťazec, číslo apod.). Od existujúcich typov vieme odvodzovať vlastné typy. Jednoduché typy: textový reťazec, celé/desatinné čísla a ich varianty, binárne dáta, logická hodnota, dátum, čas, časový interval, typy prevzaté z DTD pre jednoduchšie konverzie; typy môžeme rozširovať/obmedzovať – obdoba integritných obmedzení; vieme dokonca definovať referenčnú integritu.

- vlastné typy vieme odvodiť z už definovaných typov pomocou reštrikcie, vytvorením zoznamu alebo zjednotením typov. U väčšiny typov vieme definovať rôzne integritné obmedzení: reťazce – length, minLength, maxLength, pattern, enumeration, whiteSpace; číselné typy – maxInclusive, maxExclusive, minInclusive, minExclusive, totalDigits, fractionDigits, pattern; binárne dáta – dĺžka (minimálna, maximálna), pattern, enumeration, whiteSpace

- komplexné typy – slúžia na modelovanie elementov, ktoré obsahujú ďalšie elementy alebo atribúty; vieme určiť poradie elementov, ich opakovanie, voliteľnosť atď.

- modelovanie obsahu:

- sekvencia za sebou idúcich elementov – <xs:sequence>

- výber jedného z elementov – <xs:choice>

- nezáleží na poradí elementov – <xs:all>

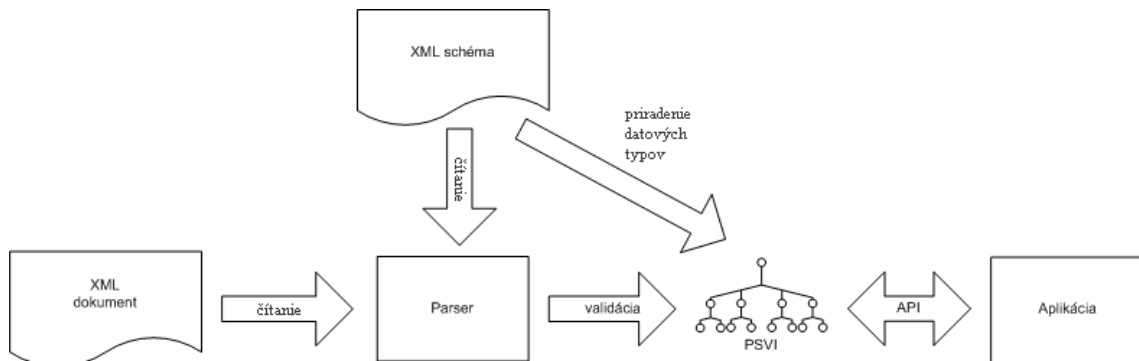
- zmiešaný obsah (medzi elementmi sa môže objaviť text) – <xs:complexType mixed="true">

- práca s PSVI - infoset = abstraktný dátový model dokumentu XML

PSVI = Post Schema Validation Infoset, t.j. otypovaný infoset dokumentu.

Vznikne priradením dátových typov na základe validácie oproti schéme. Pri čítaní cez API môžeme pracovať priamo s typovými hodnotami, vid' obr. , kdežto

bez PSVI nám API môže vracat' iba textové reťazce. Nad PSVI pracujú moderné dotazovacie jazyky ako XPath 2.0 alebo XQuery.



Obr. 2.2: Parser sprístupňujúci PSVI

### Ako validovať XML dokument pomocou XML schémy?

Každý dokument, ktorý používa gramatiku špecifikovanú XML schémou, musí špecifikovať umiestnenie gramatiky, ktorú používa, pomocou atribútu *xsi:schemaLocation*, ak sú použité menné priestory (namespace) a *xsi:noNamespaceSchemaLocation* v inom prípade. Tieto atribúty sú obyčajne zapísané v dokumente pri koreňovom elemente najvyššej úrovne, aj napriek tomu, že môžu byť umiestnené v akomkoľvek elemente; viac detailov v XML Schema Part 1 section 4.3.2 [XMLSSTR]. Ukážka, ktorá nepoužíva menné priestory:

```

<document
  xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
  xsi:noNamespaceSchemaLocation='document.xsd'>
  ...
</document>
  
```

Ukážka s cieľovým menným priestorom. Je chybou špecifikovať iný menný priestor (atribút *xmlns:xsi*) ako cieľový definovaný schémou.

```

<document
  xmlns='http://my.com'
  xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
  xsi:schemaLocation='http://my.com document.xsd'>
  ...
</document>
  
```

### Výhody XML

Je vhodné použiť XML schémy a XML na účel opisu sieťových protokolov. Je niekoľko dôvodov a aspoň 4 výhody vyplývajúce z tohto rozhodnutia:

1. špecifikácia je čitateľná strojmi, ktoré ju vedia implementovať
2. možné chyby, z hľadiska zlej interpretácie človekom, sú minimalizované
3. XML je neutrálny jazyk a nie je asociovaný so žiadnym špecifickým programovacím jazykom
4. špecifikácie protokolu v XML sa ľahko môžu prenášať sieťou, takže kód sa automaticky vygeneruje na vzdialenom sieťovom uzle. Tento prístup je užitočný pri distribúcii nových verzií protokolu. Každý uzol by si stiahol najnovšiu XML špecifikáciu a vygeneroval kód z nej.  
[Abdullah04]

## 3. Implementácia

Existuje viacero formálnych špecifikácií a nástrojov, napr. SDL (Specification and Description Language); Spin (Simple ProMeLa Interpreter); ESTELLE; LOTOS (Language Of Temporal Ordering Specifications); Petri Nets (PN); UPPAAL; MSC Message Sequence Chart; UML (Unified Modeling Language); ASN (Abstract Syntax Notation) a aj nejaké pokusy ohľadne XML [Abdullah04]. Viac o spomenutých špecifikáciách v literatúre [Abdullah04]. Z existujúcich nástrojov budú spomenuté iba relevantné, takže zďaleka nie všetky.

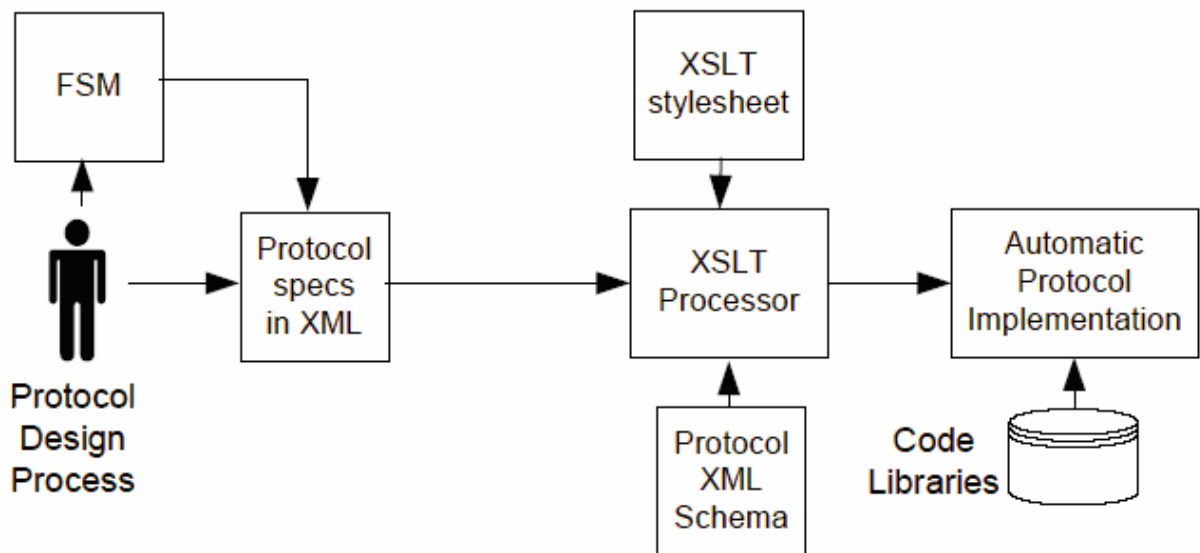
### 3.1 Existujúce nástroje a implementácie

#### 3.1.1 NetPDL

Je značkovací jazyk, ktorého cieľom je opísať protokoly od OSI vrstvy 2 do OSI vrstvy 7. NetPDL je akronym, ktorý znamená Network Protocols Description Language (opisný jazyk pre sieťové protokoly). Pod sieťovým protokolom sa myslí bunka, rámec, paket alebo PDU a opisný jazykom sa myslí to, že rôzne nástroje môžu použiť ten istý opis na splnenie rôznych úloh. [NetPDL03]

#### 3.1.2 FSM + XML špecifikácia + XSLT transformácie

Táto špecifikácia, a zároveň pokusná implementácia v Jave, je založená na modelovaní konečno stavového automatu pomocou XML subjazyka. Takýto model je potom pomocou XSLT transformácií a použitia CBSE (komponentovo založená technológia) priamo implementovaný vygenerovaním kódu v Jave [Abdullah04]. Výkonnosť takéhoto automaticky vygenerovaného kódu je samozrejme o čosi pomalšia ako manuálna implementácia, autor však udáva, že optimalizácia komponentov by to mohla zmierniť. Problém opisu štruktúr protokolu sa chystajú autori riešiť za pomoci integrácie ASN.1 do ich samotnej XML špecifikácie. Ukážka procesu takéhoto generovania je na obrázku 3.1.



**Obr. 3.1: Navrhovaný automatický proces produkcie protokolov**

XML bolo zvolené ako jazyk, ktorým sa vyjadruje špecifikácia z niekoľkých dôvodov: po prvé, je to otvorený štandard; po druhé, nástroje pre spracovanie a ovládanie XML dokumentov a súvisiacich technológií sa stávajú veľmi populárnymi a dostupnými; po tretie, pre poskytnutie schopnosti spolupráce medzi systémami sa autori rozhodli pre použitie neutrálneho jazyka, ktorý nie je asociovaný so žiadnym programovacím jazykom; po štvrté, dostupnosť XSLT technológií zjednodušuje transformáciu XML dokumentu do akéhokoľvek užívateľom uprednostňovaného programovacieho jazyka. [Abdullah04]

## 3.2 Použitie XML ako opisného jazyka

Nasleduje návrh použitia XML, jeho pravidlá a samotné príklady z praxe budú uvedené vo vlastnej kapitole, načrtnime jeho princípy.

Generujúci program na základe špecifikácie z XML schémy vytvorí plán dekompozície paketu, podľa ktorého potom vie označovať a extrahovať potrebné dáta. Nad nimi sa potom vykoná požadovaná operácia, opísaná v XML dokumente príslušnom modelovej XML schéme. Ťažisko tejto metódy je v dobre definovaných akciách, ktoré by mali spĺňať vyššie spomínané úvahy. Využíva sa silná schopnosť XML pre opis dát, ktoré môžu eventuálne prísť.

XML schéma slúži na všeobecný opis dát, listy stromu danej schémy sú polia, ktoré sa očakávajú. Je dobré plne využiť možnosti XML schémy pre modelovanie

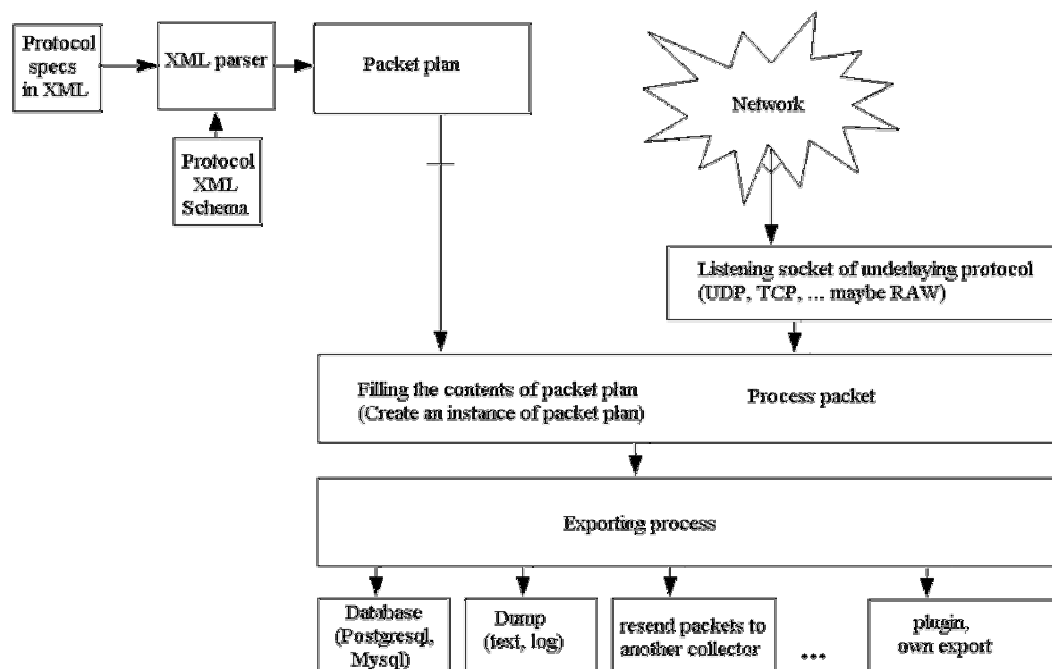
dátového modelu ako voľby tvorby vetvenia (choice), sekvencie (sequence) alebo všetkých (all) elementov. [XMLSPRI]

XML dokument, konformný danej schéme, opisuje jednotlivé akcie a ciele pre daný element. Dá sa povedať, že v XML dokumente priradíme dátam akcie, nazvime to „mappingom - mapovaním“ (v rámci tohto dokumentu). Dáta pre elementy (ich obsah) budú získané z prichádzajúcich paketov. Funkcia XML sa použitím akcií rozširuje z opisu štruktúry na opis .

XML súbor konformný schéme mapuje akcie priradené jednotlivým častiam paketu. Je tak dosiahnuté oddelenie vrstvy návrhu od vrstvy spracovania paketu. Funkcie, ktoré sa nad poliami paketu majú vykonať, sú teda budované už nad hotovou štruktúrou dát. XML povoľuje oveľa viac vecí ako prípadná implementácia softwaru na spracovanie a sparsovanie paketu, preto je nutné, aby opisy boli jednoznačné a akcie definované ako atomické a jednoduché. Je vhodné, ak by sa vytvorili štandardizované šablóny alebo rozhrania pre samotný parser a akcie, ktoré by viedli k zníženiu počtu zlých implementácií na minimum.

V zjednodušenej podobe sa dá povedať, že XML slúži ako akýsi konfiguračný / modelovací jazyk pre softvér vytvárajúci aplikácie pracujúce s daným protokolom.

Blokový diagram spracovania paketu aj s ďalším spracovaním, ktoré sa deje v nástroji pre zber flowov je načrtnutý na obrázku 3.2.



Obr. 3.2: Bloková schéma spracovania dát v nástroji JXColl

#### Bezstavové protokoly

Výhodu daného návrhu využijeme hlavne pri bezstavových protokoloch (napr. UDP), keďže po navrhnutí možných prípadov štruktúr dát, ktoré môžu nastať, a namapovaní akcií k nim, máme nosnú konštrukciu pripravenú na bezchybné spracovanie alebo vysielanie dát. Je nutné mapovanie akcií pre bod, ktorý vysiela a bod, ktorý prijíma dáta. V praxi teda pre jeden model štruktúry protokolu opísaný v XML schéme bude existovať špeciálny XML dokument mapujúci akcie pre vysielajúci aj prijímajúci bod.

#### Stavové protokoly

Spomenutý návrh pre stavové protokoly nepostačuje, keďže jeho primárne ťažisko leží v opise dát. Chýba možnosť špecifikovať stavy cez FSM. Podľa [Abdullah04] sa dá tiež využiť technológia XML pre opis týchto stavov.

Takýmto spojením získame opis FSM daného protokolu v XML, kde prenášané dáta sú tiež jednoznačne definované modelom v XML schéme. Spracovanie a akcie nad dátami sú mapované XML dokumente.

#### **Problémy a nejednoznačnosti**

Problém celého návrhu môže nastať v akciách. Okrem toho, že by mali spĺňať vyššie uvedené tézy, sú to práve ony, čo tvoria celú logiku a funkčnosť, preto je im venovaná osobitá pozornosť a špeciálna časť v ďalšej kapitole, rozpisujúca detaily tohto návrhu.

Poznámka: prípadová štúdia: pre Netflow (@ CISCO) zberač dát je dizajn vhodný i bez rozšírení pre FSM (čiastočne preto, že UDP bolo použité ako protokol prenosu dát).

### **3.3 Praktické využitie pri meraní kvality služieb v sieťach v rámci IPFIX.**

Praktickou ukážkou využitia opisu štruktúr sieťových protokolov by mala byť implementácia takéhoto parser-a v rámci zberača nameraných údajov o tokoch (flowoch) v sieťach - JXColl. Skratka JXColl je odvodená od Java XML Collector a je to program starajúci sa o zber flow-ov (informácie o prúdoch údajov, Netflow 5/9) z jedného alebo viacerých bodov exportu. Jeho hlavná úloha je pripraviť dáta na ďalšie spracovanie a prezentáciu koncovou užívateľskou aplikáciou (analyzerom). Dokáže

spracovávať dáta z viacerých bodov exportu vysielajúcich UDP pakety s tokmi (UDP sa dá jednoducho zmeniť trebárs na SCTP pomocou úpravy jednej triedy). Vykoná zopár jednoduchých akcií s obsahom flowov a pripraví dáta na uloženie do archívov alebo ich pošle priamo analyzérovi pre okamžitý výstup užívateľovi. Zatiaľ bol testovaný s voľne prístupnými SRBD (systém riadenia bázy dát) ako napr. postgresql a mysql. Dizajn databázy a nízke rozdielne rozdistribúvanie dát však prispeli k slabej výkonnosti, z ktorej vyplynuli dlhé čakania pri požiadavkách na dáta. Z tohto dôvodu analyzér potreboval viac času na vytvorenie výstupov v prijateľnom čase. Takže sa vymyslelo a vytvorilo priame spojenie medzi analyzérmi a kolektorom. Optimalizuje dotazy a posiela iba dáta potrebné pre danú meraciu metódu. JXColl je vyvíjaný ako Java konzolová aplikácia; XML značí možnosť rozpoznávania a využívania obsahu prichádzajúcich paketov na základe ich opisu pomocou XML. Keďže nástroj je plne kompatibilný s protokolom Netflow 9, zväčša spĺňa aj požiadavky na protokol IPFIXu. Toto tvrdenie je podložené prácou L. Deriho [Deri04]. Neplatí to však celkovo, ale iba pre transport flowov pomocou UDP. S rozšírením o podporu SCTP alebo TCP sa počítalo a modul programu, ktorý má za úlohu prijímať dáta je ľahko rozšíriteľný o takúto podporu.

Čo teda IPFIX obnáša a kde je miesto tohto zberača sa dozvieme v nasledujúcich staniach.

### 3.3.1 IPFIX protokol

IPFIX zariadenie pozostáva zo skupiny navzájom kooperujúcich procesov, ktoré implementujú funkčné bloky - merací, exportný, zberací. Z iného pohľadu, IPFIX zariadenie predstavuje sieťový prvok, ktorý implementuje IPFIX protokol. Takéto zariadenie vykonáva nasledujúce funkcie:

- transformuje riadiace informácie do šablón,
- zahŕňa pakety zachytené pozorovacím bodom do záznamov tokov,
- konvertuje šablóny a záznamy tokov na správy protokolu IPFIX,
- posiela tieto správy kolektorovi.

Použitím mechanizmu šablón, nové atribúty opisujúce premávku môžu byť pridané do záznamov tokov bez nutnosti zmeny štruktúry exportovaných dát. Ďalšou výhodou je, že aj keď kolektor nepozná sémantiku niektorých atribútov v šablóne, aj napriek tomu dokáže záznam toku interpretovať. Aj keď bude tieto neznáme atribúty

ignorovať, stále môže dekodovať niektoré relevantné informácie o danom toku. V neposlednom rade je tu možnosť konfigurácie šablón, čo sa umožňuje exportovať len potrebné údaje, čím dochádza k šetreniu prenosových kapacít siete na jednej strane, ako aj hardvérových prostriedkov (napr. pamäť, výpočtová kapacita) exportného a zberacieho procesu na strane druhej. [IPFIX]

### **3.3.1.1 Zberací proces**

Získava údaje od jedného alebo viacerých exportných procesov. Na správne interpretovanie týchto údajov používa ID šablóny. Exportér by mal zaručiť doručenie opisu šablóny spolu s jej ID, a to periodickým zasielaním v určitom časovom intervale. Ďalším dôvodom na takéto periodické zasielanie je obmedzená časová platnosť šablóny. Ak kolektor nemá opis šablóny, mal by si záznamy tokov, ktoré nevie interpretovať uchovať a po prijatí príslušnej šablóny by ich mal následne spracovať. Takto sa zabráni strate informácií o tokoch. Na druhej strane, interval exportu opisu šablón musí byť zvolený optimálne, pretože pri príliš malej periodicite môže dôjsť k zahlteniu kolektora záznamami tokov, ktoré nie je schopný interpretovať.

### **3.3.1.2 Informačný model**

Zahŕňa zoznam atribútov, ktoré musí byť schopný exportovací proces exportovať v takom zmysle, že musí existovať možnosť konfigurácie zariadenia aby exportovalo ľubovoľný z týchto atribútov:

- verzia IP protokolu (len v prípade ak na danom pozorovacom bode je podporovaných viacero verzií)
- zdrojová IP adresa
- cieľová IP adresa
- typ protokolu (TCP, UDP, ICMP, ...)
- zdrojový a cieľový port (v prípade ak je použitý TCP alebo UDP protokol)
- počet prenesených paketov (ak je paket fragmentovaný, započítava sa každý fragment)
- počet prenesených bajtov (zahŕňa celý paket vrátane hlavičky)
- ToS (Type Of Service) – v prípade IPv4, alebo TC (Traffic Class) pri IPv6
- návestie toku (flow label) pri IPv6 protokole

- návěstie MPLS (ak je MPLS na pozorovacom bode podporované)
  - časová značka prvého paketu patriaceho do toku
  - časová značka posledného paketu patriaceho do toku
  - informácia o použítom vzorkovaní (ak je nejaké použité)
  - jedinečný identifikátor pozorovacieho bodu
  - jedinečný identifikátor exportného procesu
- Odporúčané atribúty, ktoré by exportný proces mal vedieť exportovať:
- typ a kód ICMP, ak sa jedná o ICMP správu
  - číslo (index) vstupného/výstupného rozhrania – neplatí v prípade, že pozorovacím bodom je sonda
  - replikačný faktor – počet výstupných paketov ktoré vzniknú zo vstupného paketu (jedná sa o dynamickú charakteristiku multicastových tokov, v prípade unicastového toku má hodnotu 1)
- Voliteľné atribúty, ktoré exportný proces môže byť schopný exportovať:
- TTL (Time To Live) v prípade IPv4, HL (Hop Limit) v prípade IPv6
  - príznaky (flags) hlavičky IP paketu
  - príznaky hlavičky TCP segmentu
  - počet zahodených paketov (v prípade fragmentácie sa počíta každý fragment)
  - počet fragmentovaných paketov (počet všetkých paketov, ktoré majú nastavený príznak fragmentácie)
  - IP adresa ďalšieho uzla resp. stanice (next hop)
  - číslo zdrojového autonómneho systému BGP
  - číslo cieľového autonómneho systému BGP
  - číslo ďalšieho nasledujúceho (next hop) autonómneho systému BGP

Zoznam voliteľných atribútov nemusí byť konečný, nakoniec, jedná sa o výhodu ak je merací proces schopný exportovať čo najviac údajov.

### **3.3.1.3 Dátový model**

Opisuje ako sú dáta reprezentované v záznamoch tokov. Tento formát musí byť rozšíriteľný aby sa v budúcnosti mohli pridávať nové atribúty a konfigurovateľný, aby sa exportovali len atribúty potrebné pre dané meranie, resp. sledovanie premávky. Musí existovať možnosť opísať poradie a typ atribútov, ktoré budú do exportu zahrnuté a

taktiež musí byť podporovaný protokol, ktorý dokáže indikovať zahltenie alebo preťaženie siete, aby nedochádzalo k strate exportovaných údajov bez upozornenia.

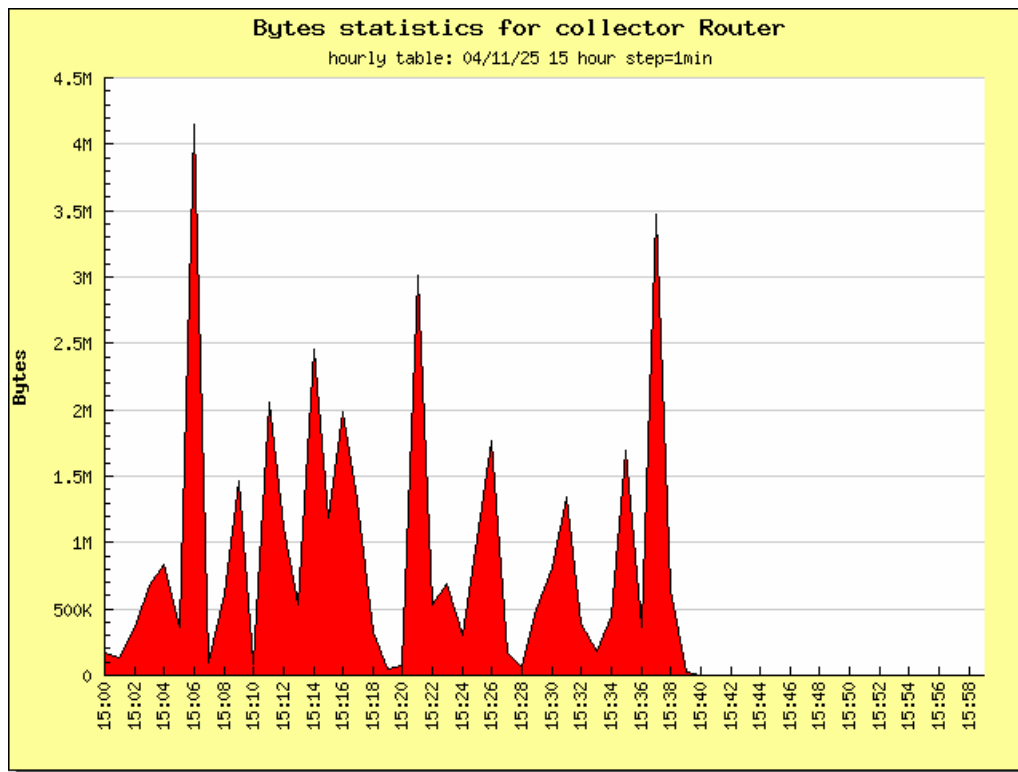
### **3.3.2 Existujúce nástroje a implementácie podobného účelu**

Existuje pár desiatok nástrojov určených na spracovávanie tokov, t.j. fungujúcich ako zberač. Iba zopár z nich však podporuje Netflow 9, teda de facto IPFIX protokol. Spomeniem iba tieto, keďže ako jediné riešia podobný problém, len s rozdielom, že sú napísané manuálne, teda „natvrdo“.

#### **NetFlow 2.1**

[http://netflow.cesnet.cz/n\\_netflow.php](http://netflow.cesnet.cz/n_netflow.php)

Tento projekt sa vyvinul do komerčného riešenia Flow Inspector (viac na [http://www.caligare.com/s\\_product.php](http://www.caligare.com/s_product.php)). Jeho súčasťou je práve spomínaný zberač, ktorý distribuuje dáta medzi viac databáz, čiže prevádza zároveň akúsi minimálnu analýzu. Nad týmto je ako vyhodnocovacia aplikácia webové rozhranie. Medzi jeho vlastnosti sa zaraďuje podpora pre Netflow 9, plne distribuovaná architektúra, podpora swapového (dočasného) lokálneho súboru v prípade dočasnej nedostupnosti databázy, možnosť špecifikácie ktoré položky nás zaujímajú, v rámci analýzy heuristiku detekcie aplikácií, ktoré vygenerovali daný tok a odstraňovanie duplicitných tokov (v prípade, že viac zdrojov sa zberá do jedného kolektora). Ukážka na obr. 3.3 ukazuje meranie šírky pásma – bandwidth-u.



**Obr. 3.3: Meranie bandwidth-u pomocou Flow Inspector**

### **FLOWD**

<http://www.mindrot.org/flowd.html>

Podľa autorov je to malý, rýchly a bezpečný kolektor pre Netflow. Bezpečný preto, lebo dokáže bežať ako proces s oddelenými privilégiami. Okrem tohto vie filtrovať a značkovať flowy na základe opisu filtra syntaxou podobného paketovým filtrom (viac tcpdump, libpcap). Flowy ukladá do kompaktného binárneho formátu, vieme určiť, ktoré políčka sa majú ukladať. Zaujímavosťou je, že podporuje prijímanie flowov aj z multicastových skupín, t.j. je jednoduché ho začleniť do skupiny redundantných kolektorov.

### **ntop**

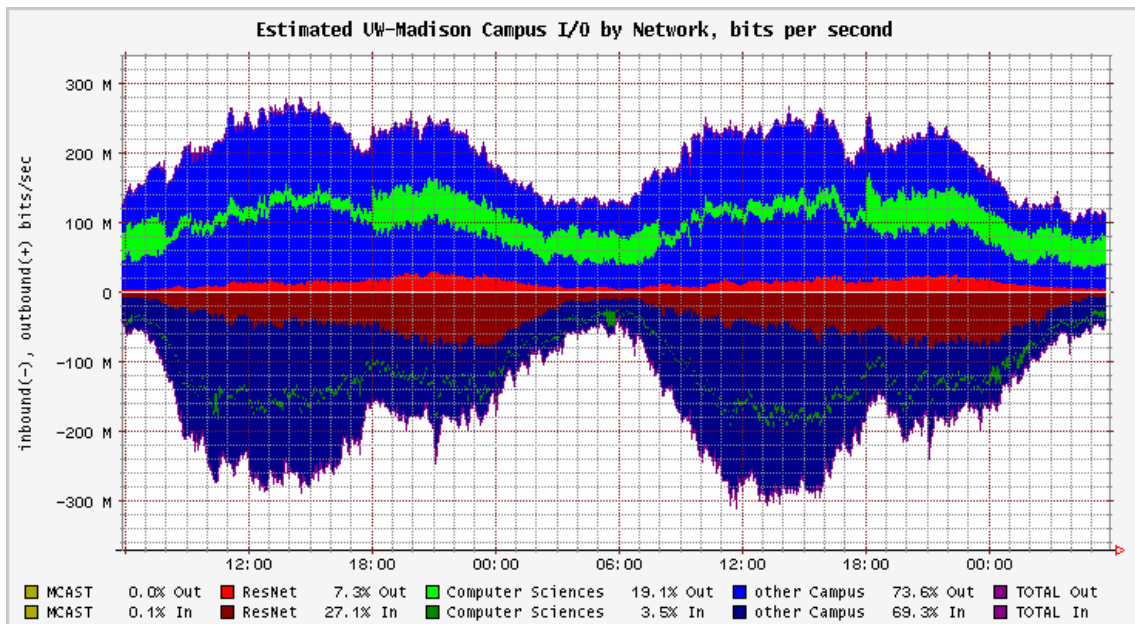
<http://www.ntop.org/>

Je komplexné riešenie pre spracovávanie nielen Netflow-u ale aj sFlow. Dokonca má aj schopnosť vlastného monitoringu sieťových rozhraní a autor vytvoril tiež prácu schopnú vyhodnocovať toky a exportovať ich vo forme Netflow. Štatistiky tokov sú zberané do RRD databázy. Program vie vyhodnotiť množstvo štatistík a zobrazí rôzne grafy. Beží ako démon, ktorý je ovládaný z užívateľského webového rozhrania.

### flow-tools

<http://www.splintered.net/sw/flow-tools/>

Je knižnica a zbierka programov určených na zber, posielanie, spracovanie a generovanie správ z Netflow dát. Dáta sú podobne ako u ntop-u ukladané do RRD súborov. Tieto programy síce zatiaľ nepodporujú formát Netflow 9, uvádzam ich ale pre ich značné rozšírenie a veľkú obľubu. Jeden z autorov dokonca predsedá skupine, ktorá vyvíja IPFIX, predpokladám, že podpora pre IPFIX sa dostaví v dohľadnej dobe. Ukážka na obr. 3.4 ukazuje meranie šírky pásma - bandwidthu pomocou flow-tools a flowscanu.



Obr. 3.4: Meranie bandwidth-u pomocou flow-tools

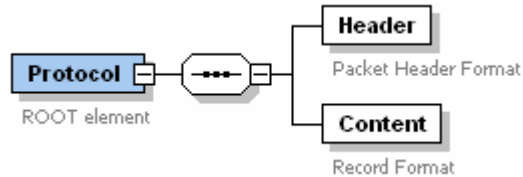
## 3.4 Návrh, koncepcia a architektúra JXColl ako aplikácie využívajúcej opis pre strojové spracovanie protokolu.

Typografické konvencie: úryvky zdrojových kódov alebo častí XML budú vyznačené farebne, prípadne v *kurzívě*.

### 3.4.1 Lexikálny analyzátor (parser) XML opisu

#### Model štruktúry dát:

- je určený XML schémou, zodpovedá teda dobre formovanému popisu, podľa pravidiel zápisu XML schémy, predpísaná je základná kostra, vid' obr. 3.5



Obr. 3.5: Predpísaná kostra pre model štruktúry

Zdrojový kód pre danú kostru:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="unqualified" attributeFormDefault="unqualified"
version="1.0">
  <xs:annotation>
    <xs:documentation>Protocol packet format</xs:documentation>
  </xs:annotation>
  <xs:element name="Protocol">
    <xs:annotation>
      <xs:documentation>ROOT element</xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:sequence>
        <xs:element name="Header">
          <xs:annotation>
            <xs:documentation>Packet Header Format</xs:documentation>
          </xs:annotation>
          <xs:complexType>
            <xs:annotation>
              <xs:documentation>Header Format</xs:documentation>
            </xs:annotation>
          </xs:complexType>
        </xs:element>
        <xs:element name="Content">
          <xs:annotation>
            <xs:documentation>Record Format</xs:documentation>
          </xs:annotation>
          <xs:complexType>
            <xs:annotation>
              <xs:documentation>Contents of Body</xs:documentation>
            </xs:annotation>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Lexikálny analyzátor na základe tejto kostry vytvorí v pamäti plán dekompozície vstupných dát. Pod plánom dekompozície rozumieme spojený zoznam štruktúr - uzlov, z ktorých každý popisuje časť vstupných dát a ktorého prechodom vieme pre každý takýto uzol načítať príslušné dáta. Pod elementmi Header a Content je možné vytvoriť akúkoľvek štruktúru, ktorá je dobre formovaná a je možné oproti nej validovať XML dokument. Táto kostra je fixná a parser ignoruje atribúty navyše. Výsledná štruktúra dát

je určená listami stromu XML schémy. Jednoznačný identifikátor je meno elementu, pre ďalšie použitie v samotnom programe je kľúčový.

Pre ďalšie elementy, ktoré sú deťmi Header alebo Content parser rozpoznáva nasledujúce elementy XML schémy a ich atribúty ():

Klasické elementy: *xs:element*, *xs:complexType*

Elementy dokumentácie: *xs:annotation* a *xs:documentation*

Všeobecné atribúty pre modely:

*name* – meno elementu

*minOccurs*, *maxOccurs* – určujú početnosť daného elementu, ak je udaná horná hranica „*unbounded*“, sú dve možnosti: udanie početnosti akciou určujúcou maximálnu hranicu z predošlého elementu a (alebo) neudanie explicitnej hranice, tým pádom sa berie hranica veľkosť dát, ktoré prišli. Ak je udaná spodná hranica 0, t.j. ak daný element môže a nemusí byť, je nutné špecifikovať akciu, ktorá determinuje presné prípady, kedy má byť element prítomný.

*content* – obsah elementu

Modely obsahu:

ak je to nutné, modely obsahu môžu mať tiež svoju identifikáciu, ktorú špecifikujeme v atribúte *id*

- *sequence* – sekvencia za sebou idúcich elementov
- *all* – nezáleží na poradí elementov
- *choice* – voľba; rozhodovanie je založené na mene prvého elementu pre každú skupinu elementov v danej voľbe, tento musí obsahovať podmienku vo formáte: „*MenoElementu\_is\_podmienka*“, kde podmienka môže byť „*equalInteger*“, „*greaterInteger*“, kde *Integer* je prirodzené číslo v rozpätí daného typu (atribút *base*), je nutné písať podmienky a názvy tak, aby bol model deterministický, kontrola či je model deterministický sa neprevádza

Listy tvoria samotné dáta, preto parser rozpoznáva pre ne navyše tieto atribúty:

*base* – určuje typ listu, teda jeho veľkosť (počet bytov, príp. bitov), v implementácii sú podporované tieto typy: *xs:unsignedByte*, *xs:unsignedShort*,

*xs:unsignedInt*, planovaná je podpora pre vlastný typ bitového poľa (viď príklad TCP/IP, element *xs:simpleType*).

- každý list by mal mať predpísanú akciu, preto sa rozoznávajú navyše elementy určujúce atribúty: `<xs:attribute name="action" type="xs:string"/>` a `<xs:attribute name="target" type="xs:string"/>`

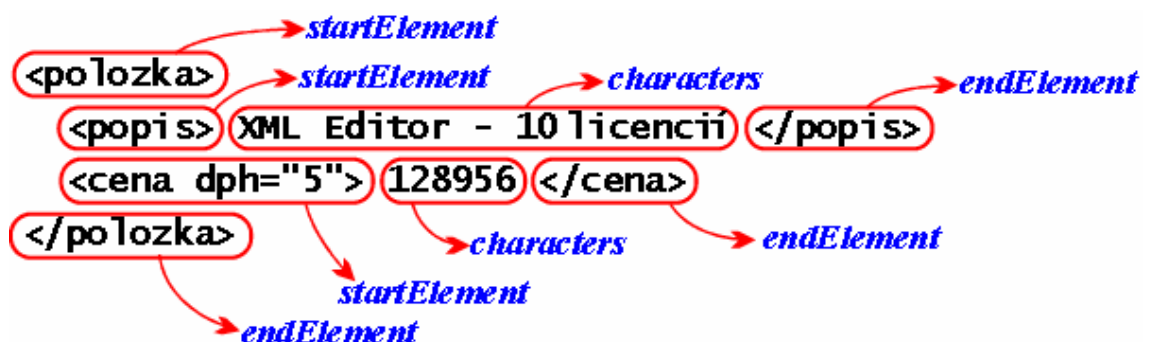
XML schéma je sama o sebe veľmi obširna, parser iné typy elementov alebo atribútov ignoruje, resp. prehlási, že schéma nie je uskutočniteľná ako model. Samozrejme, je možné rozšíriť jeho schopnosti, ako pri každom jazyku je stále treba zvážiť aj možné problémy alebo nejednoznačnosti, ktoré môžu nastať. Detaily a ukážky v kapitole o príkladoch použitia.

#### Akcie priradené modelu:

- sú opísané v XML súbore konformnom modelovej XML schéme, súbor by mal obsahovať opisy akcií aj pre všetky vetvy modelu *choice* t.j. pre všetky elementy, ktoré sú listami. Formáty atribútov *action* a *target*: *action*="{akcia;}"; *target*="{cieľ;}". Kde akcia je volanie akcie s parametrami a cieľ je jeden z cieľov podporovaných aplikáciou. Viac detailov a ukážok v kapitole o príkladoch použitia.

#### Technológia parsera:

- spomedzi rôznych technológií a všetkých možných parserov je najvhodnejšia implementácia SAX (Simple API for XML) – analyzátor založený na udalostiach, pre jeho rýchlosť a jednoduchosť implementácie. Je určený špecifikáciou na <http://www.saxproject.org> a je použitá jeho implementácia v XML parseri Xerces2 Java Parser [XERCES2]. Keďže je riadený udalosťami, vieme presne určiť následnosť elementov, príp. kontrolovať, či sú podporované a či obsahujú príslušné atribúty. Ukážka princípu volania udalostí pri takejto analýze je na obr. 3.6.



Obr. 3.6: Udalosti pri analýze pomocou SAXu

Pri vyvolaní udalosti `startElement` sa vytvorí uzol v pláne dekompozície dát, ktorý po vyvolaní príslušnej udalosti `endElement` už obsahuje dostatok informácií ako načítať jemu prislúchajúce dáta zo vstupu. Plán dekompozície je implementovaný pomocou kolekcii v Jave ako dynamický `LinkedList`.

Alternatívny analyzátor je založený na DOM (Document Object Model), ktorý vytvorí zo všetkých elementov v XML súbore v pamäti strom. Následným prechádzaním stromu získavame potrebné informácie. Nevýhody tohto prístupu sú hlavne väčšia náročnosť na čas CPU a zbytočné mrhanie miestom v pamäti. Nás zaujímajú po prejení stromu dokumentu iba jeho listy, zvyšok stromu je návod ako dostať dáta do príslušného listu, nie je teda potreba mať ho k dispozícii po celý čas.

### **AKCIE**

Akcie určujú sémantiku a predspracovávajú dáta, prípadne pri dobrom návrhu nimi vieme nadefinovať aj samotné celkové spracovanie dát. Dobre vytvorená knižnica všeobecných akcií, ktoré vieme odvodiť z doterajších skúseností s vytváraním sieťových protokolov, dokáže byť základom pre návrh protokolu podľa vlastného gusta, bez nutnosti dodefinovania vlastných akcií. Značnú časť implementácie vieme urýchliť a niekedy aj celkovo odbúrať, ak máme k dispozícii knižnicu dobrých akcií. Dobre navrhnutá akcia by podľa analýzy mala byť atomická, jednoduchá, mala by byť základným kameňom. Spájanie takýchto akcií nad jedným poľom potom vedie k zložitejším akciám, ktoré v konečnom dôsledku nemusíme explicitne definovať. Dá sa povedať, že takto vieme zadeliť akcie do typov. Jednoduché akcie, zložené, vlastne definované tvoria knižnicu akcií použiteľných v XML dokumente. Vieme vyčleniť akcie, ktoré rozširujú model (t.j. akcie, ktoré chceme použiť v spolupráci s modelom obsahu). Ukážka niektorých akcií, ktoré boli navrhnuté pre potreby pri opise protokolu Netflow je v tabuľke 3.1 (pozn. ID určuje, že parametrom by mala byť identifikácia už načítaného poľa). Treba podotknúť, že iba niektoré z nich by boli vhodné ako všeobecné, väčšina je šitá priamo pre potreby danej aplikácie.

Meno akcie	Parametre	Popis
checkver	x	slúži na kontrolu, či dané políčko je rovné x. V prípade, že nie je, dáta sa považujú za chybné.
checksequence	ID	kontroluje relatívnu číselnú sekvenciu, správny prírastok je definovaný obsahom poľa ID (pozn. môže byť aj konštanta).
uptimetounixsec	ID_suptime, ID_unixsec	konvertuje dáta z uptime-u na unix seconds, podľa vzorca $ID\_unixsec + (dáta - ID\_suptime)$
exporttodb	typ, tabulka, pole	exportuje dáta z daného poľa ako pole v tabuľke tabulka do databázy typu typ
store_to_template	ID_template	dané dáta elementu sa uložia ako súčasť šablóny
decode_from_template	ID_template	pole bude dekodované na základe šablóny
...	...	...

**Tab. 3.1 Akcie nad dátami**

### Problémy s akciami

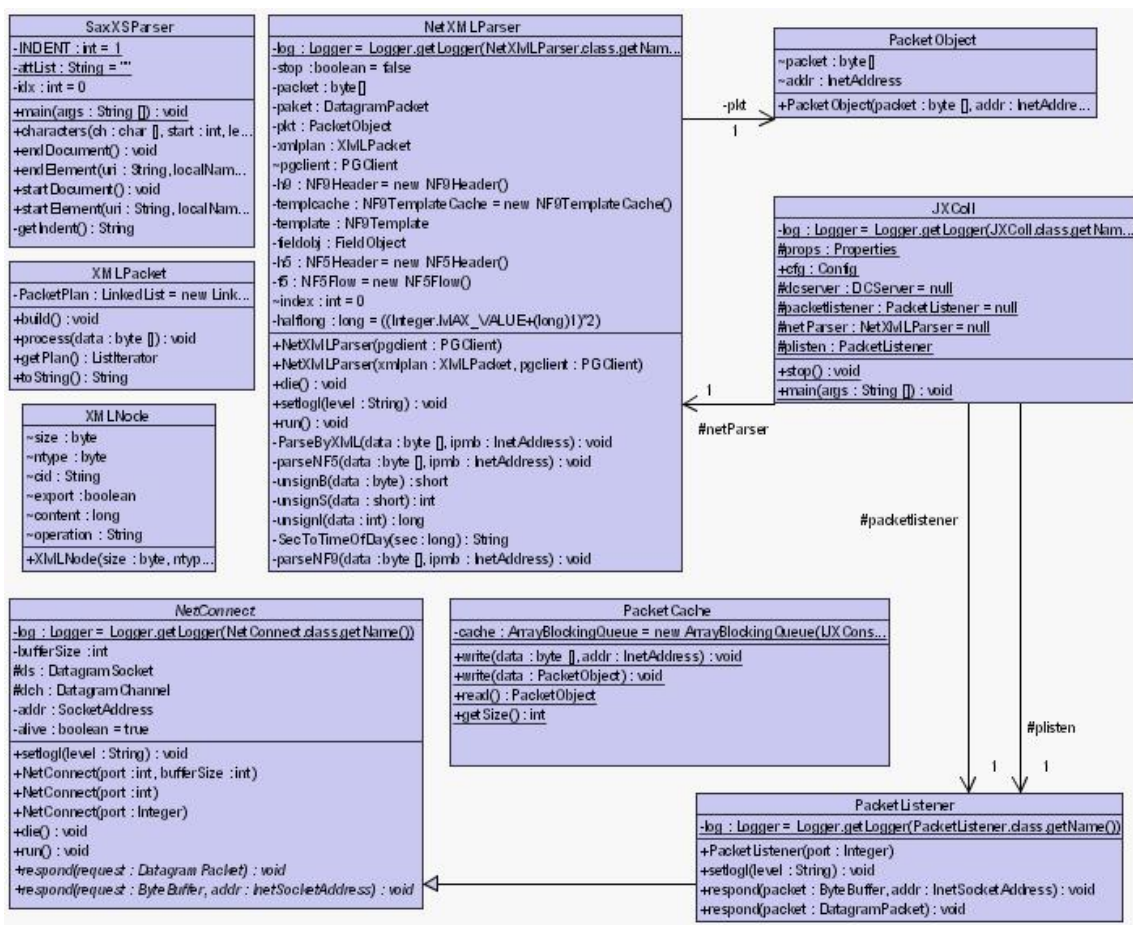
Pri návrhu akcií je treba rátať aj s problémami, ktoré môžu vzniknúť. Stačí spomenúť nejednoznačnosť, nízky stupeň všeobecnosti alebo zložitosť akcie, ktorá môže viesť k nedeterminovanému správaniu, príp. k zablokovaniu aplikácie.

Problémy s akciami sa môžu týkať aj požiadavky na predčítavanie (prefetch) dát. Akcia k svojej funkcii potrebuje vedieť dáta, ktoré sú načítané neskôr, a ak táto akcia je kľúčová môže aj toto viesť k nejednoznačnému chovaniu programu. Ako príklad možno spomenúť zle navrhnutý model a akciu, ktorá by mala riadiť počet iterácií, no ich počet nie je dopredu známy a má sa načítať v budúcnosti v závislosti práve na danej akcii. Je samozrejmé, že táto akcia je zle navrhnutá a celkom isto povedie k nepredvídateľnému chovaniu aplikácie.

### 3.4.2 Model implementácie parsera v JXColl

Obr. 3.7 zachytáva štruktúru a objektový model časti programu JXColl (JXColl je zároveň hlavná trieda). Je znázornená časť, ktorá tvorí rozhranie pre prístup

k sieťovým soketom. Sieťový soket (zásuvka) je jeden koniec dvojcestného komunikačného spojenia medzi dvomi programami odohrávajúcего sa na sieti. Soket je viazaný na číslo portu (brána), takže TCP vrstva vie identifikovať aplikáciu, ku ktorej sú dáta smerované a predať jej ich. Vrstva rozhrania je implementovaná pomocou abstraktnej triedy NetConnect, aby sa dalo jednoducho prepnúť na iný protokol nižšej vrstvy. Momentálna podpora zahŕňa TCP a UDP, keďže tieto sú podporované aj v Java API. Výstup z tohto rozhrania, ktoré prijíma pakety a konvertuje ich na bytové buffre (zásobníky), sa ukladá do vyrovnávacej pamäte – PacketCache, odkiaľ sa berie do ďalšej časti, vlákna, ktoré sa stará o samostatnú analýzu paketu a jeho spracovanie, príp. export.



**Obr. 3.7: Objektový model zachytávania a analýzy paketov v JXColl**

Analýzu obsahu paketu vytvára trieda NetXMLParser, ktorá má implementované metódy analyzujúce na základe plánu dekompozície z tried XMLPacket. Trieda XMLPacket je vytvorená z uzlov XMLNode a samotný parser,

ktorý ju vytvorí je v triede SaxXSParser. Trieda NetXMLParser obsahuje aj klasickú implementáciu protokolu Netflow vo verziách 5 a 9. Za pomoci týchto tried a testovacích meraní bolo možné otestovať aké oneskorenie spôsobuje všeobecnejší prístup k analýze paketu. V neskoršej kapitole je popísané koľko percent času strávil program v ktorej triede a aké z toho vyplynuli závery.

### 3.4.3 Zdieľané opisy dát - šablóny

IPFIX v svojom protokole (principiálne upravený Netflow 9) využíva dynamické rozpoznávanie dát na základe vopred poslaných šablón. Telo paketu tak môže obsahovať šablónu alebo dáta, ktoré treba na základe šablóny dekodovať. Šablóna (template) je sekvencia dvoch polí, z ktorých prvé určuje typ a druhé jeho dĺžku. Na základe takejto sekvencie sa dá obsah dát dekodovať. Exportovací bod takto posielal iba údaje pre dané meranie. Táto šablóna sa však prenáša iba medzi exportovacím a zberacím bodom. Zberací bod, v tomto prípade JXColl ju ďalej nemusí propagovať. Je však potrebné, aby šablóny boli k dispozícii pre všetky časti meracieho systému podľa IPFIXu. Urýchli to komunikáciu a odpadnú problémy s prenosom šablón. V implementácii JXColl sú šablóny v zdieľanej triede Templates, ktorá sa zdieľa s analyzujúcou aplikáciou.

Je zaujímavé mať šablóny v jednom centrálnom úložisku opísané napr. pomocou XML. Každá časť si vie stiahnuť príslušnú šablónu, šetrí sa miestom, zjednodušuje sa správa šablón a riadenia celej architektúry. Viac o šablónach v príslušnom štandarde [Claise04].

Tabuľka 3.2 ukazuje niektoré ID polí z Netflow 9. Podľa poslednej verzie štandardu je ich skoro 90 [Claise04].

Typ poľa	Hodnota ID	Dĺžka (byte)	Opis
IN_BYTES	1	N (štandardne 4)	Delta počítadlo (relatívne k predošlej hodnote) prichádzajúcich bajtov asociovaných s IP Flowom
IN_PKTS	2	N (štandardne 4)	dtto čo IN_BYTES, počíta ale pakety
FLAWS	3	N (štandardne 4)	počet flowov, ktoré boli agregované
PROTOCOL	4	1	číslo IP protokolu
SRC_TOS	5	1	typ služby nastavený pri príchode na vstupné rozhranie

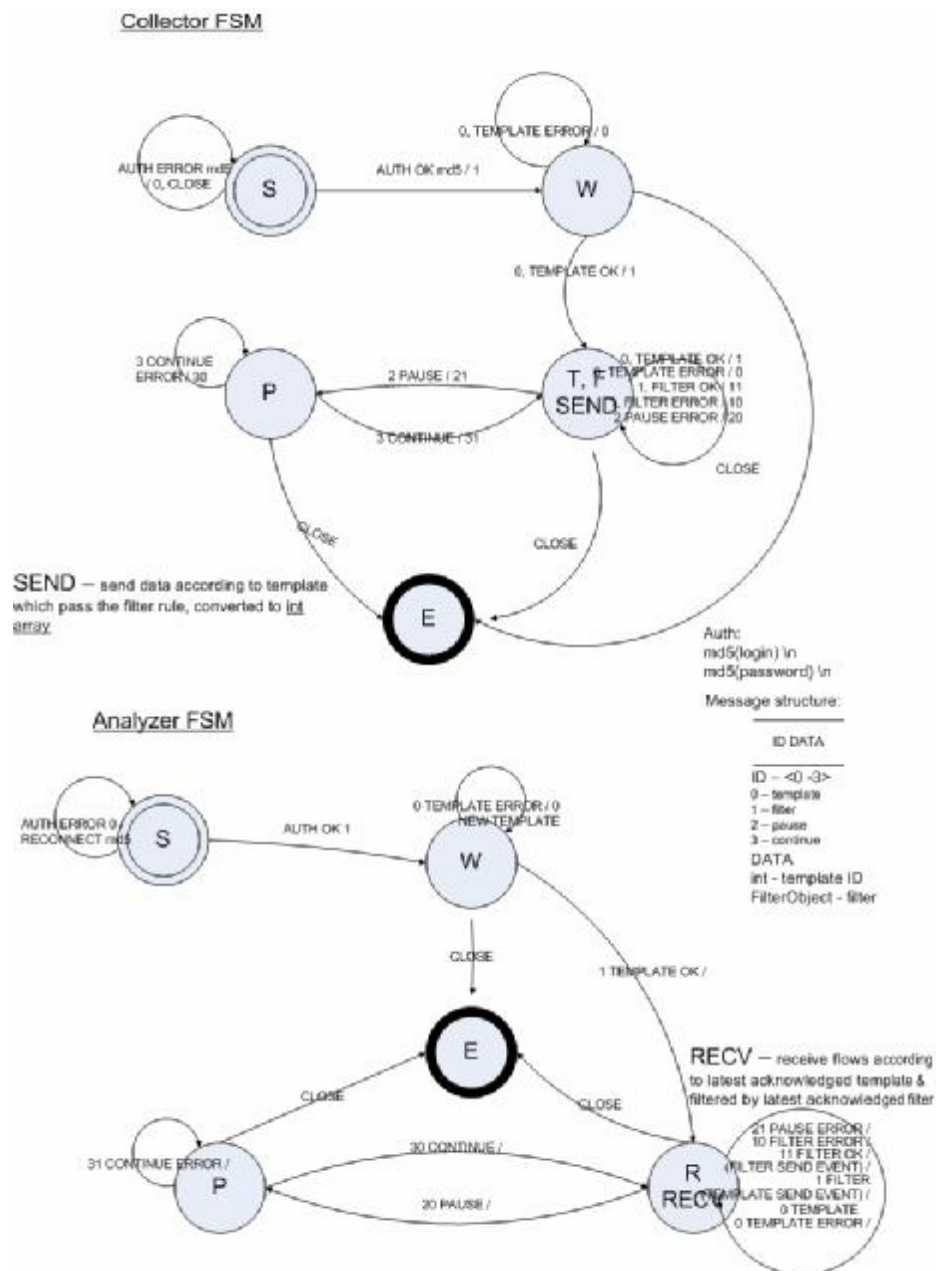
TCP_FLAGS	6	1	kumulatívny byte všetkých TCP indikátorov pre tento flow
L4_SRC_PORT	7	2	TCP/UDP zdrojové číslo portu napr. FTP, Telnet, a pod.
IPV4_SRC_ADDR	8	4	IPv4 zdrojová adresa
SRC_MASK	9	1	zdrojová sieťová submaska v lomítkovej notácii, t.j. počet súvisiacich bitov v maske podsiete
INPUT_SNMP	10	N (štandardne 2)	index vstupného rozhrania
L4_DST_PORT	11	2	TCP/UDP cieľové číslo portu napr. FTP, Telnet, a pod.
IPV4_DST_ADDR	12	4	IPv4 cieľová adresa
DST_MASK	13	1	cieľová sieťová submaska v lomítkovej notácii, t.j. počet súvisiacich bitov v maske podsiete
OUTPUT_SNMP	14	N(štandardne 2)	index výstupného rozhrania
IPV4_NEXT_HOP	15	4	IPv4 adresa next-hop router-a (smerovača pre ďalší bod cesty paketu)
SRC_AS	16	N (štandardne 2 alebo 4)	zdrojové číslo BGP autonómneho systému
DST_AS	17	N (štandardne 2 alebo 4)	cieľové číslo BGP autonómneho systému
BGP_IPV4_NEXT_HOP	18	4	IPv4 adresa next-hop router-a v BGP doméne
MUL_DST_PKTS	19	N (štandardne 4)	počítadlo IP paketov odchádzajúcich cez multicast asociovaných s IP flowom
MUL_DST_BYTES	20	N (štandardne 4)	počítadlo IP bajtov odchádzajúcich cez multicast asociovaných s IP flowom
LAST_SWITCHED	21	4	systemový uptime (doba prevádzky v ms), keď bol presmerovaný (komutovaný) posledný paket flowu
FIRST_SWITCHED	22	4	systemový uptime (doba prevádzky v ms), keď bol presmerovaný (komutovaný) prvý paket flowu
...	... 90	...	...

**Tab. 3.2 Ukážka niektorých identifikátorov a kódov polí Netflow 9**

Šablóny sú zaujímavou črtou Netflow-u, z pohľadu parsera XML tvoria zaujímavú výzvu hlavne ohľadne návrhu akcií. Väčšinu záťaže procesora pri parsingu tvorili práve ony.

### 3.4.4 Protokol pre priame pripojenie na analyzátor

DC (direct connect, priame pripojenie) protokol bol vymyslený z dôvodu zrýchlenia spracovávania dát medzi kolektorom a vyhodnocovacou aplikáciou. Umožňuje priame spojenie medzi JXColl a Analyzerom, kde sa namerané dáta spracovávajú skoro v real - time režime a premieta sa chovanie siete s minimálnym oneskorením. Obrázok 3.8 ukazuje stavy a formát dát, ktoré sa prenášajú. Vzhľadom na to, že protokol obsahuje viac ako dve stavy bolo jednoduchšie implementovať ho klasicky.

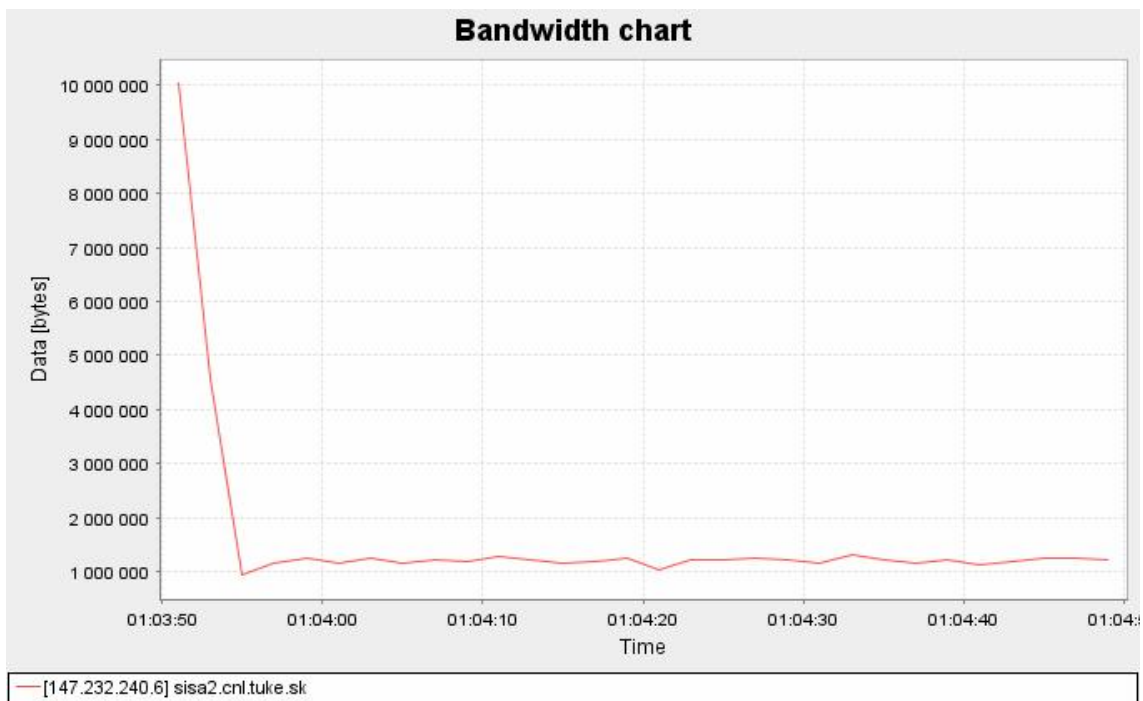


Obr. 3.8: FSM DC protokolu

### 3.4.5 Ukážkové merania, vyhodnotenie rýchlosti nástroja

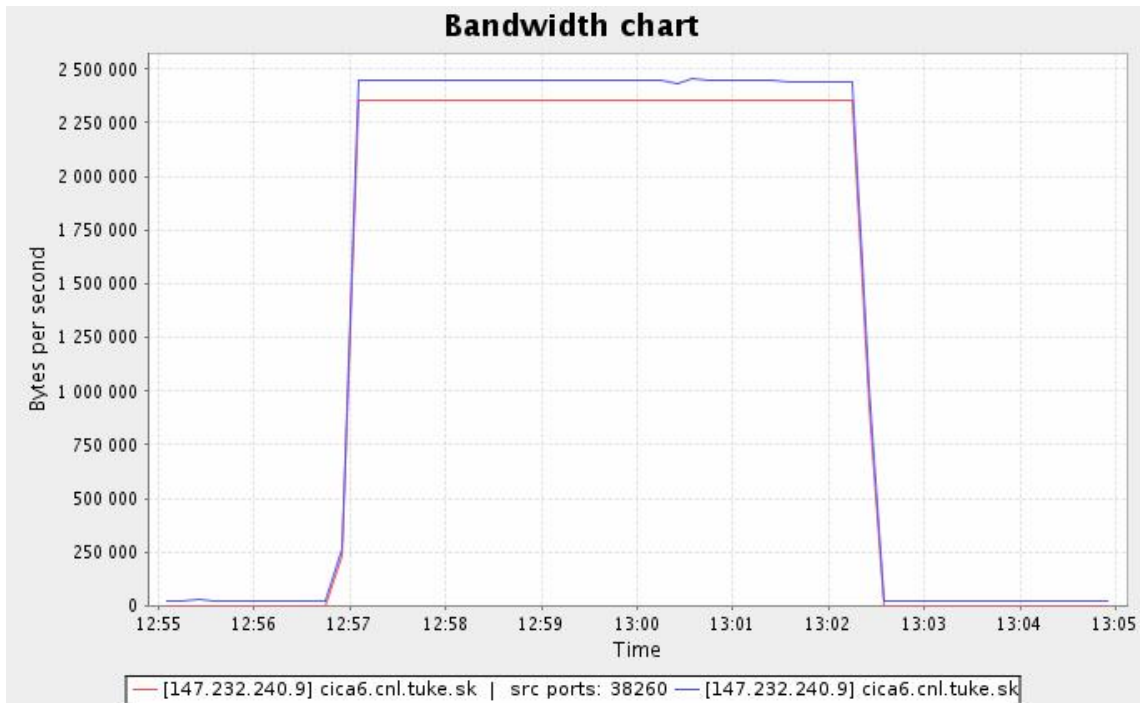
Testy nástroja prebehli za rôznych podmienok. Analyzujúca aplikácia zobrazovala iba merania šírky pásma (bandwidth) a počtu paketov (packet rate). JXColl exportoval údaje buď priamo alebo sa archivovali do databázy. Merania prebehli za použitia rôznych exportovacích bodov – Cisco router s podporou Netflow 9, linuxová nProbe L. Deriho a vlastná implementácia exportného a meracieho bodu basicmeter [Andre04]. Topológia siete pre linuxový merač bola založená na funkcii port mirroring-u (kopírovania celkového toku dát cez router na pasívny port), kde sa odzrkadľoval tok dát zo segmentu laboratória počítačových sietí. Linuxový merač fungoval samozrejme ako pasívny poslucháč a spracovával tento tok pomocou nprobe alebo basicmeter-a. Cisco router s podporou Netflow 9 bol v testovacom segmente pre IPV6 a sledoval vysielania IPV6 audio alebo video streamov. Prehľad meraní je v nasledujúcej ukážke obrázkov.

Na obr. 3.9 je meranie šírky pásma, kde je sledovaný prenos súboru pomocou FTP na 100Mbit Ethernet linke, tok dát bol spočiatku neobmedzený, potom sa v aplikácii na prenos FTP súborov limitoval na 1MB/s. Oneskorenie meracieho systému za použitia DC spojenia bolo približne 1.2s, bralo do úvahy oneskorenie linky a exportovacieho bodu.



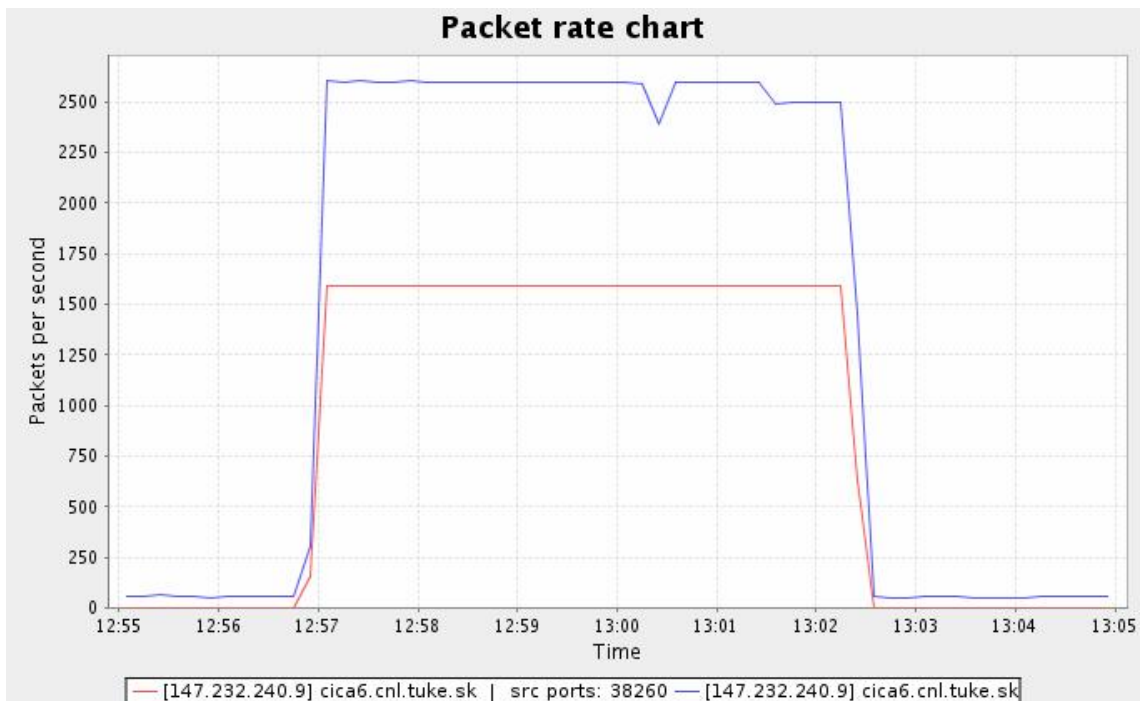
Obr. 3.9: Linuxový merací bod, basicmeter, skoro real –time meranie

Obrázok 3.10 ukazuje meranie šírky pásma pre audio a video streaming z IPV6 segmentu. Exportovací bod je Cisco router schopný exportu Netflow 9. Čiara, ktorá je nižšie je video zložka, vyššie je celkový tok dát.



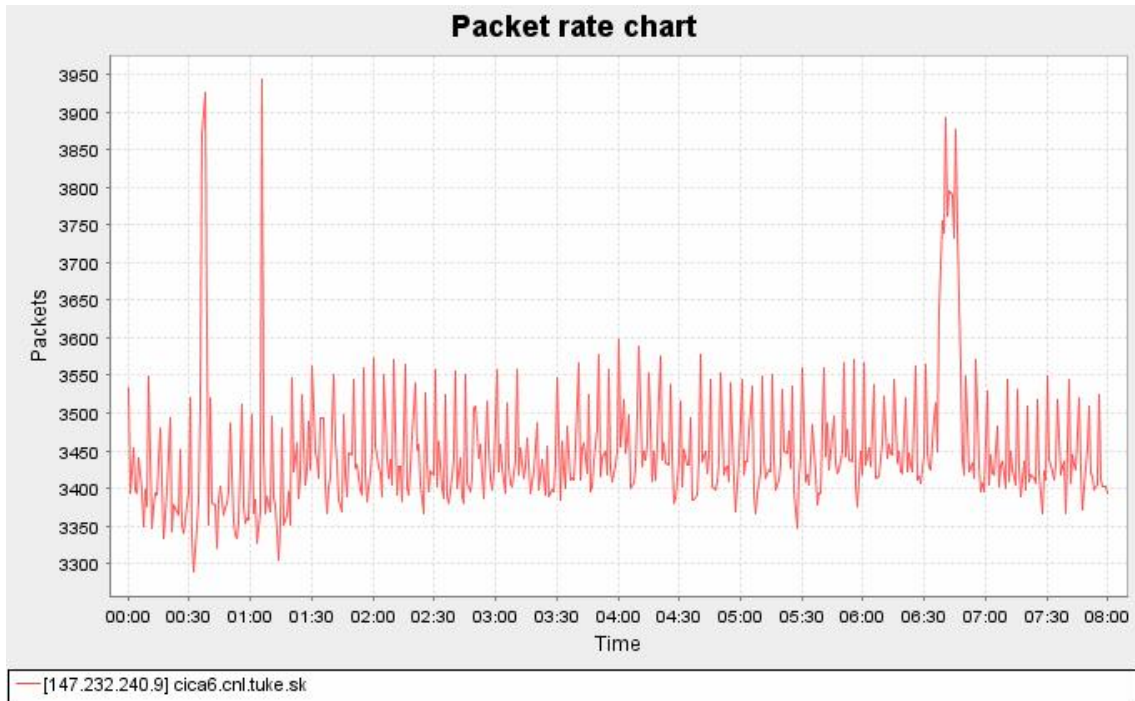
**Obr. 3.10: Bandwidth pre audio a video streaming, IPV6**

Na obr. 3.11 je podobná situácia, ale meral sa packet rate.



**Obr. 3.11: Packet rate pre audio a video streaming, IPV6**

A nakoniec obrázok 3.12 z archívnych meraní z databázy. Bola sledovaná šírka pásma v IPV6kovom segmente laboratória.



**Obr. 3.12 Archívne merania**

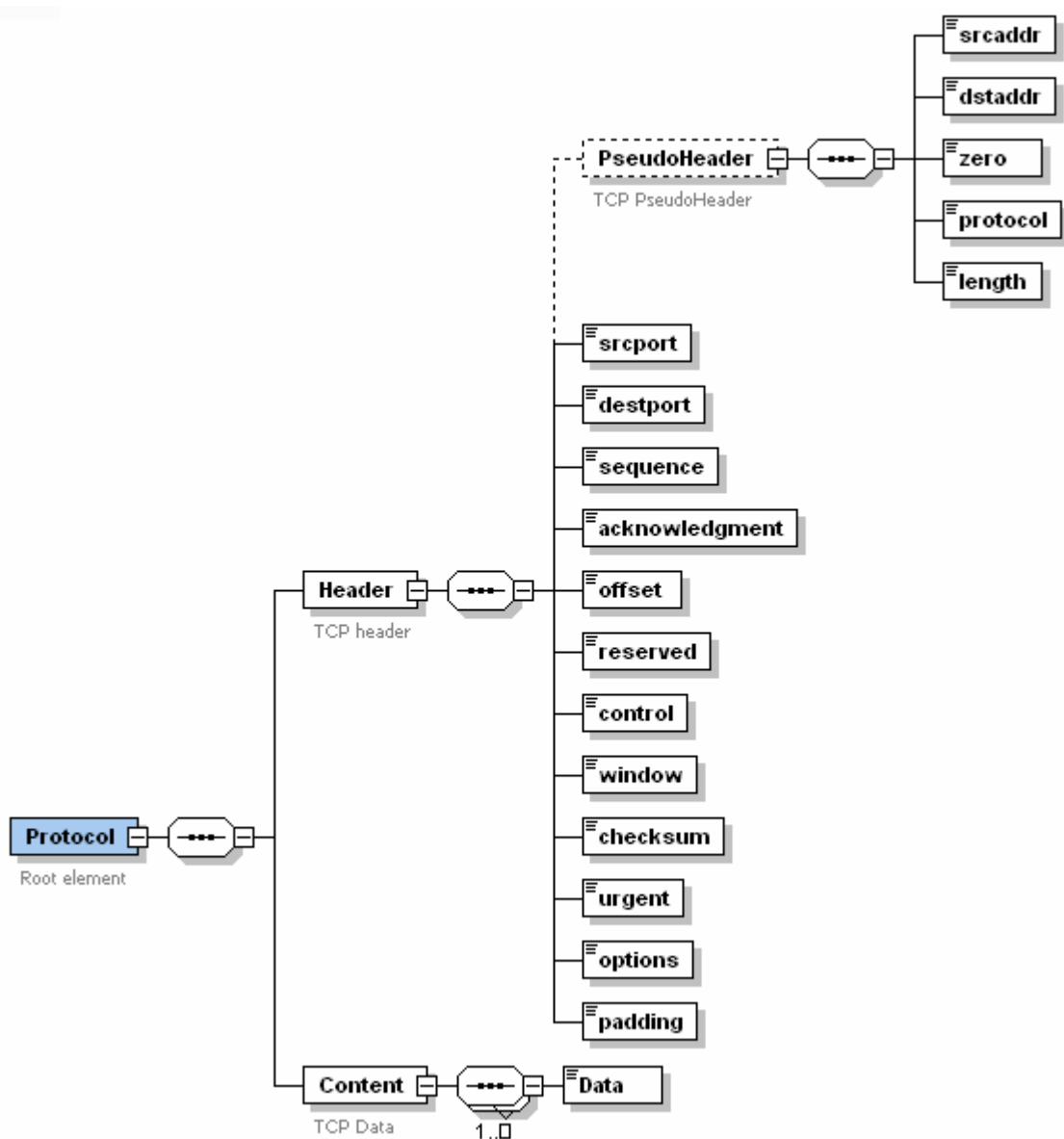
Popri meraniach sa sledoval aj výkon a zaťaženie procesora kolektorom. Pomocou optimalizačných nástrojov pri rovnakom zaťažení bola sledovaná vyťaženosť procesora tými – ktorými metódami objektov kolektora JXColl. Metódy klasického spracovávanía paketu bez použitia XML popisu zaberali skoro 29% z času prideleného aplikácií. Pri použití XML parsera tento čas vzrástol o 11%, teda potvrdilo sa očakávanie, že manuálna implementácia protokolu je rýchlejšia ako automatická. Toto spomalenie sa dá znížiť optimalizáciou parsera a akcií.

### **3.5 Príklad použitia opisu v XML pre NF5, NF9 a TCP/IP**

Pre dve strany, bez nejakých ďalších stavov potrebujeme schému určujúcu štruktúru, opis akcií pre vysielajúcu a prijímajúcu stranu. V prípade viacerých stavov bude nutné určiť takúto trojicu každému z možných prechodov medzi stavmi.

Výhoda XML opisu je, že môže obsahovať aj dokumentáciu, takže prípadné vygenerovanie opisu štruktúr je jednoduché. Nie je nutné distribuovať opisný súbor a dokumentačný súbor, rozsiahlejšia dokumentácia je priamo k dispozícii.

Na obrázku 3.13 je príklad pre model štruktúry TCP/IP podľa [RFC793]. Neobsahuje však opis akcií (ani v modeli), tie je nutné definovať do osobitného XML dokumentu pre každý stav.



**Obr. 3.13: Príklad modelu štruktúry TCP/IP**

Zdrojový kód pre takéto opis vyzerá zložitejšie. Je však jednoznačné, že sa bude ručne upravovať iba v najvyššej núde. V dnešnej dobe existuje mnoho editorov XML, ktoré uľahčujú prácu a vlastne aj dizajn alebo návrh opisu štruktúry. Za povšimnutie stojí definícia vlastného typu reprezentujúca binárne pole.

Zdrojový kód pre opis TCP/IP:

```

<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified" attributeFormDefault="unqualified">
<xs:annotation>

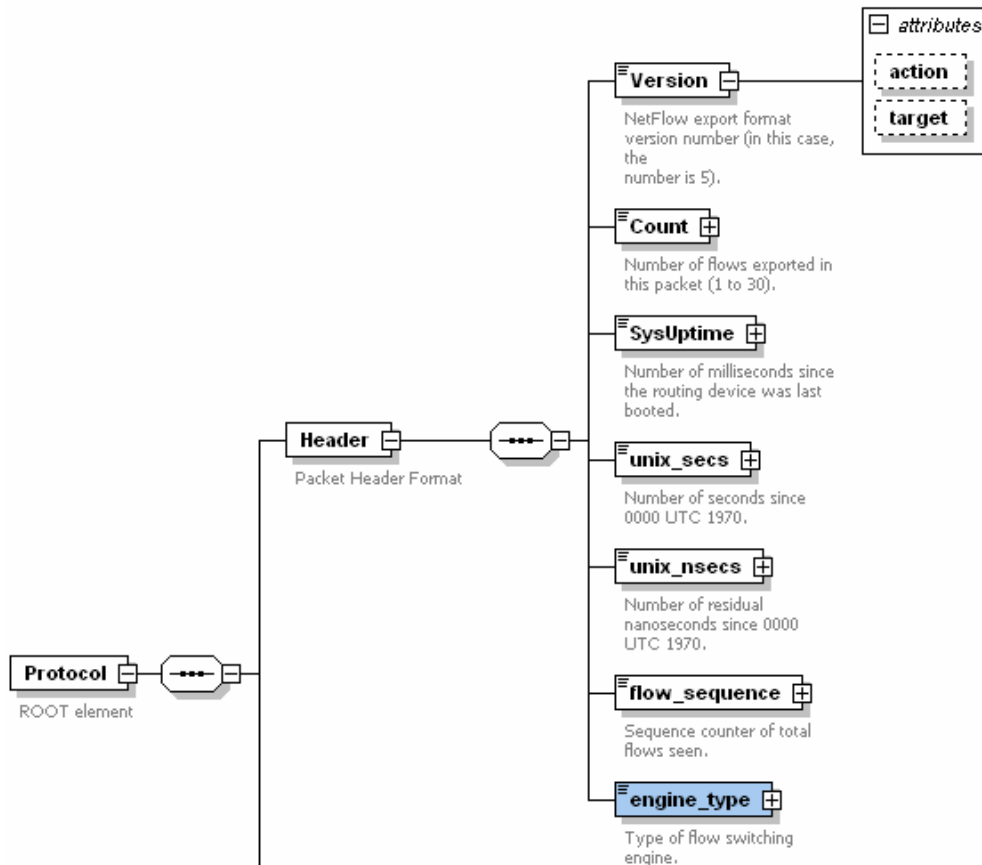
```

```
<xs:documentation>TCP/IP structure scheme</xs:documentation>
</xs:annotation>
<xs:element name="Protocol">
  <xs:annotation>
    <xs:documentation>Root element</xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:sequence>
      <xs:element name="Header">
        <xs:annotation>
          <xs:documentation>TCP header</xs:documentation>
        </xs:annotation>
        <xs:complexType>
          <xs:sequence>
            <xs:element name="PseudoHeader" minOccurs="0">
              <xs:annotation>
                <xs:documentation>TCP PseudoHeader</xs:documentation>
              </xs:annotation>
              <xs:complexType>
                <xs:sequence>
                  <xs:element name="srcaddr">
                    <xs:complexType>
                      <xs:simpleContent>
                        <xs:extension base="xs:unsignedInt"/>
                      </xs:simpleContent>
                    </xs:complexType>
                  </xs:element>
                  <xs:element name="dstaddr">
                    <xs:complexType>
                      <xs:simpleContent>
                        <xs:extension base="xs:unsignedInt"/>
                      </xs:simpleContent>
                    </xs:complexType>
                  </xs:element>
                  <xs:element name="zero" type="xs:unsignedByte"/>
                  <xs:element name="protocol" type="xs:unsignedByte"/>
                  <xs:element name="length" type="xs:unsignedShort"/>
                </xs:sequence>
              </xs:complexType>
            </xs:element>
            <xs:element name="srcport" type="xs:unsignedShort"/>
            <xs:element name="destport" type="xs:unsignedShort"/>
            <xs:element name="sequence" type="xs:unsignedInt"/>
            <xs:element name="acknowledgment" type="xs:unsignedInt"/>
            <xs:element name="offset">
              <xs:simpleType>
                <xs:restriction base="xs:decimal">
                  <xs:totalDigits value="4"/>
                  <xs:minInclusive value="0"/>
                  <xs:maxExclusive value="16"/>
                  <xs:fractionDigits value="0"/>
                </xs:restriction>
              </xs:simpleType>
            </xs:element>
            <xs:element name="reserved">
              <xs:simpleType>
                <xs:restriction base="xs:decimal">
                  <xs:minInclusive value="0"/>
                  <xs:maxExclusive value="64"/>
                  <xs:totalDigits value="6"/>
                  <xs:fractionDigits value="0"/>
                </xs:restriction>
              </xs:simpleType>
            </xs:element>
            <xs:element name="control">
              <xs:simpleType>
                <xs:restriction base="xs:decimal">
                  <xs:minInclusive value="0"/>
                  <xs:maxExclusive value="64"/>
                  <xs:totalDigits value="6"/>
                  <xs:fractionDigits value="0"/>
                </xs:restriction>
              </xs:simpleType>
            </xs:element>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

```
</xs:element>
<xs:element name="window" type="xs:unsignedShort"/>
<xs:element name="checksum" type="xs:unsignedShort"/>
<xs:element name="urgent" type="xs:unsignedShort"/>
<xs:element name="options">
  <xs:simpleType>
    <xs:restriction base="xs:decimal">
      <xs:minInclusive value="0"/>
      <xs:maxExclusive value="16777216"/>
      <xs:totalDigits value="24"/>
      <xs:fractionDigits value="0"/>
      <xs:pattern value="[0-1]"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
<xs:element name="padding">
  <xs:simpleType>
    <xs:restriction base="xs:decimal">
      <xs:totalDigits value="8"/>
      <xs:minInclusive value="0"/>
      <xs:maxExclusive value="256"/>
      <xs:fractionDigits value="0"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="Content">
  <xs:annotation>
    <xs:documentation>TCP Data</xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:sequence maxOccurs="unbounded">
      <xs:element name="Data" type="xs:unsignedInt"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:schema>
```

Ukážka XML modelu a súboru kde sú popísané akcie nad protokolom Netflow

5. Model je na obr. 3.14, príslušná ukážka XML súboru s akciami nasleduje za ním.



Obr. 3.14 XML Model štruktúry protokolu Netflow 5

Úryvok zo zdrojového kódu schémy:

```
<xs:element name="Version">
  <xs:annotation>
    <xs:documentation>NetFlow export format version number (in this case, theis 5).</xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="xs:unsignedShort">
        <xs:attribute name="action" type="xs:string"/>
        <xs:attribute name="target" type="xs:string"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>
```

Príslušný kód v XML dokumente:

```
<?xml version="1.0" encoding="UTF-8"?>
<Protocol xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="Netflow5.xsd">
  <Header>
    <Version action="checkver(5)" target="">0</Version>
  </Header>
</Protocol>
```

## 4. Zhodnotenie riešenia

Opis sieťových protokolov pomocou XML sa ukazuje ako vhodný formálny prístup. XML ako štandard podporuje veľké množstvo spoločností a existuje nespočetné množstvo nástrojov pre prácu s XML. S jeho pomocou sa dá zrýchliť vývoj a návrh sieťových protokolov. Zvýši sa ich spoľahlivosť a jednoznačnosť implementácie. Toto je však vykúpené nižšou výkonnosťou a vyšším nárokom aplikácie, ktorá automaticky implementuje takto opísaný protokol, na procesorový čas. V práci je ukázaný a implementovaný iba návrh opisu štruktúry sieťového protokolu. Nie je v ňom zahrnutý opis stavov protokolu, preto sa nemôže považovať za plnohodnotný opis sieťových protokolov.

Aplikácia vyvinutá v rámci diplomovej práce slúži ako zberač nameraných údajov o stave tokov v sieti. Dokáže ich priamo preposielať vyhodnocovacej aplikácii, archivovať do databázy pre neskoršie vyhodnotenie alebo zapisovať v textovom formáte do súboru alebo na obrazovku. Experimentálne v nej bol otestovaný XML parser na opisy štruktúry protokolu. Aplikácia aj naďalej slúži ako jeden z funkčných blokov systému pre merania tokov v sieťach podľa IPFIX.

V budúcnosti by bolo dobré plne vyšpecifikovať opis sieťových protokolov, t.j. zahrnúť doňho aj stavy protokolu, čo podľa [Abdullah04] je tiež možné modelovaním v XML. Optimalizácia a urýchlenie XML parsra pre opis je ďalšou z úloh, ktoré vylepšia tento prístup k návrhu sieťových protokolov. Aplikácia JXColl je implementovaná s ohľadom na IPFIX, ktorý sa stále vyvíja a preto aj v tomto smere treba doimplementovávať stále nové funkcie a požiadavky.

## 5. Zoznam použitej literatúry

- [Abdullah04] I. S. Abdullah, D. A. Menascé, „Protocol specification and automatic implementation using XML and CBSE“, Department of Computer Science, MS 4A5 George Mason University, <http://mason.gmu.edu/~iabdulla/>, Marec 2005
- [Andre04] Marian André (Ing. František Jakab) - Meranie a vyhodnocovanie prevádzkových parametrov v počítačových sieťach, DP, FEI TU KE 2004
- [Babich] F. Babich and L. Deotto, „Formal methods for specification and analysis of communication protocols“, IEEE Communications Surveys, <http://www.comsoc.org/pubs/surveys>
- [CCNA1] Cisco Networking Academy Program - CNAP, Cisco Certified Networking Associate - CCNA 1: Networking Basics v3.0, kapitola 2.1.5 Network protocols, e-learning-ový materiál, marec 2005
- [Claise04] B. Claise, „Cisco Systems NetFlow Services Export Version 9“, IETF RFC Informational, RFC3954, <http://www.rfc-editor.org/rfc/rfc3954.txt>, Október 2004
- [CNLTUKE] Laboratórium počítačových sietí, <http://www.cnl.tuke.sk>
- [Deri04] L. Deri, “IPFIX Implementation Notes”, <http://luca.ntop.org/draft-deri-ipfix-impl-00.txt>, Júl 2004
- [Gouda98] M.G. Gouda, „Elements of Network Protocol Design“, John Wiley & Sons, 1998.
- [IETF] Internet Engineering Task Force, <http://www.ietf.org/tao.html>

- [IPFIX] J. Quittek, T. Zseby, B. Claise, S. Zander, G. Carle and K.C. Norseth,  
“Requirements for IP Flow Information Export”, RFC3917, <http://www.rfc-editor.org/rfc/rfc3917.txt>, Október 2004.
- [KosekXML04] J. Kosek, “XML schema tutorial”,  
<http://www.kosek.cz/xml/schema/uvod.html>, Marec 2005
- [McGuire04] T. M. McGuire, “Correct Implementation of Network Protocols“,  
Department of Computer Sciences, The University of Texas at Austin,  
<http://www.cs.utexas.edu/users/mcguire/research/>, Apríl 2004
- [Netflow] Netflow technologies, <http://www.cisco.com/go/netflow>, Marec 2005
- [NetflowWP] Netflow 9 flow record format,  
[http://www.cisco.com/en/US/tech/tk648/tk362/technologies\\_white\\_paper09186a00800a3db9.shtml](http://www.cisco.com/en/US/tech/tk648/tk362/technologies_white_paper09186a00800a3db9.shtml), Marec 2005
- [NetPDL03] NetPDL Language Specification,  
<http://analyzer.polito.it/30alpha/docs/dissectors/NetPDL.htm>, Január 2005
- [RFC] Request For Comments, <http://www.rfc-editor.org/>
- [RFC793] Information Sciences Institute, “Transmission Control Protocol”, RFC793,  
<http://www.rfc-editor.org/rfc/rfc793.txt>, September 1981
- [RFC1983] “Internet Users' Glossary”, RFC3917, <http://www.rfc-editor.org/rfc/rfc1983.txt>
- [RFC3470] S. Hollenbeck, M. Rose, L. Masinter, „Guidelines for the Use of Extensible Markup Language (XML) within IETF Protocols“, RFC3470, <http://www.rfc-editor.org/rfc/rfc3470.txt>, Január 2003

[TCPIPG] Ch. M. Kozierok , “The TCP/IP Guide”,

<http://www.tcpipguide.com/index.htm>, Január 2005

[XERCES2] The Apache XML Project, Xerces2 Java Parser,

<http://xml.apache.org/xerces2-j/>, Február 2005

[XMLP] W3C XML Protocol Working Group, <http://www.w3.org/2000/xmlp/Group/>,

Január 2005

[XMLSPRI] W3C XML Schema Part 0: Primer Second Edition,

<http://www.w3.org/TR/xmlschema-0/>, December 2004

[XMLSSTR] W3C XML Schema Part 1: Structures Second Edition,

<http://www.w3.org/TR/xmlschema-1/>, December 2004

## 6. Zoznam príloh

1. CD médium – diplomová práca v elektronickej podobe, prílohy v elektronickej podobe.
2. Používateľská príručka
3. Systémová príručka

## 7. Zoznam obrázkov a tabuliek

Táto časť obsahuje zoznam všetkých tabuliek a obrázkov v diplomovej práci aj s uvedením čísla strany.

### Zoznam obrázkov

Obr. 2.1: Manuálna produkcia protokolu .....	7
Obr. 2.2: Parser sprístupňujúci PSVI .....	12
Obr. 3.1: Navrhovaný automatický proces produkcie protokolov.....	15
Obr. 3.2: Blokovaná schéma spracovania dát v nástroji JXColl .....	16
Obr. 3.3: Meranie bandwidth-u pomocou Flow Inspectoru .....	22
Obr. 3.4: Meranie bandwidth-u pomocou flow-tools.....	23
Obr. 3.5: Predpísaná kostra pre model štruktúry .....	24
Obr. 3.6: Udalosti pri analýze pomocou SAXu .....	26
Obr. 3.7: Objektový model zachytávania a analýzy paketov v JXColl .....	29
Obr. 3.8: FSM DC protokolu.....	32
Obr. 3.9: Linuxový merací bod, basicmeter, skoro real –time meranie .....	33
Obr. 3.10: Bandwidth pre audio a video streaming, IPV6 .....	34
Obr. 3.11: Packet rate pre audio a video streaming, IPV6 .....	34
Obr. 3.12 Archívne merania .....	35
Obr. 3.13: Príklad modelu štruktúry TCP/IP .....	36
Obr. 3.14 XML Model štruktúry protokolu Netflow 5 .....	39

### Zoznam tabuliek

Tab. 3.1 Akcie nad dátami.....	28
Tab. 3.2 Ukážka niektorých identifikátorov a kódov polí Netflow 9 .....	31