VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ
DEPARTMENT OF COMPUTER SYSTEMS

MONITOROVÁNÍ PROVOZNÍCH VLASTNOSTÍ IPFIX KOLEKTORU

MONITORING SERVICE PROPERTIES OF AN IPFIX COLLECTOR

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **JAN KALA**
AUTHOR

VEDOUCÍ PRÁCE **Ing. JAN WRONA,**
SUPERVISOR

BRNO 2019

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačových systémů (UPSY) Akademický rok 2018/2019

Zadání bakalářské práce

Student: **Kala Jan**
Program: Informační technologie
Název: **Monitorování provozních vlastností IPFIX kolektoru**
Monitoring Service Properties of an IPFIX Collector
Kategorie: Počítačové sítě

Zadání:

1. Nastudujte problematiku monitorování síťových toků a provozních parametrů měření.
2. Seznamte se s modulárním kolektorem IPFIXcol, který je vyvíjen organizací CESNET.
3. Navrhněte způsob měření a předávání provozních parametrů měřitelných na kolektoru.
4. Na základě zvolené metody implementujte do kolektoru zásuvný modul a demonstrujte jeho použití.
5. Diskutujte dosažené výsledky a možná rozšíření.

Literatura:

- RFC6615
- Dle pokynů vedoucího.

Pro udělení zápočtu za první semestr je požadováno:

- Splnění bodů 1 až 3 zadání.

Podrobné závazné pokyny pro vypracování práce viz http://www.fit.vutbr.cz/info/szz/

Vedoucí práce: **Wrona Jan, Ing.**
Vedoucí ústavu: Sekanina Lukáš, prof. Ing., Ph.D.
Datum zadání: 1. listopadu 2018
Datum odevzdání: 15. května 2019
Datum schválení: 26. října 2018

Abstrakt

Tato bakalářská práce se zabývá možnostmi sledování stavu IPFIX kolektoru, který je určen ke shromažďování metadat ohledně síťového provozu. Stručně seznamuje s problematikou monitorování a popisuje aktuální stav IPFIX kolektoru, který je vyvíjen organizací CESNET. Popisuje provozních vlastnosti, jež mohou být monitorovány v rámci sběru dat kolektorem při použití protokolu IPFIX. Navrhuje zásuvný modul kolektoru pro účel sběru a exportu provozních parametrů. Popisuje implementaci a obsahuje výsledky testování nového moodulu.

Abstract

This bachelor's thesis addresses possible ways of monitoring IPFIX collector, which is used for the collection of metadata about network traffic. The thesis briefly introduces the problematics of monitoring and describes the current state of IPFIX collector, which is being developed by an organization called CESNET. It also describes service properties, which can be monitored during the process of data collection using the IPFIX protocol. A new plugin is described, which is intended for the collection and the export of service properties. The thesis describes an implementation and contains results of testing of the new plugin.

Klíčová slova

Monitorování sítí, monitorování datových toků, IPFIX, SNMP, MIB, kolektor, sběr dat, správa sítě, provozní parametry

Keywords

Netwrok monitoring, data flow monitoring, IPFIX, SNMP, MIB, collector, data collection, network management, service properties

Citace

KALA, Jan. *Monitorování provozních vlastností IPFIX kolektoru.* Brno, 2019. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Jan Wrona,

Monitorování provozních vlastností IPFIX kolektoru

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Jana Wrony. Další informace mi poskytl pan Ing. Lukáš Huták z organizace CESNET. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Jan Kala
13. května 2019

Poděkování

Rád bych poděkoval panu Ing. Lukáši Hutákovi z organizace CESNET z.s.p.o., který mi byl po celou dobu vývoje ihned dostupný ke konzultacím. Dále panu Ing. Janu Wronovi za vstřícnost a motivaci při vedení této práce. A v neposlední řadě i své rodině za neustálou podporu během studia.

Obsah

Kapitola 1

Úvod

Sledování stavu sítě, neboli monitorování je nedílnou součástí správně fungující počítačové sítě. Pro obecný účel správy sítě se dnes široce využívá protokolu, který nese název SNMP. Díky němu může administrátor pozorovat a spravovat síť vzdáleně a mít tak přehled o prvcích sítě, o kterou se stará.

Součástí monitorování počítačové sítě je monitorování datových toků. Díky němu získáváme přehled o provozu v určitých částech sítě a nasbíraná data můžeme použít k zvýšení bezpečnosti a bezporuchovosti sítě. Za tímto účelem je využit protokol s názvem IPFIX, určený ke sběru metadat o datových tocích v počítačové síti. Místo sběru zpráv je nazýváno kolektorem, jehož úloha a funkce nás bude v rámci této práce zajímat nejvíce – přesněji kolektor *IPFIXcol2*, který vyvíjí organizace CESNET z.s.p.o., u něhož si vysvětlíme jeho architekturu a význam jednotlivých modulů.

Položme si ale otázku, která vedla k tématu této práce, a to: Jak se monitoruje stav prvků systému, určeného pro monitorování počítačové sítě? Pokud použijeme systém pro monitorování síťových toků, nemusí nás ihned napadnout že i tento systém může být poruchový, takže i sledování jeho stavu je důležité. A právě napravení absence způsobu sledování stavu u kolektoru *IPFIXcol2* je cílem této práce.

V textu, strukturovaném do logických celků, si v 2. kapitole vysvětlíme obecné principy monitorování a popíšeme si používané technologie. V 3. kapitole podrobněji probereme část monitorování počítačové sítě, kterou je monitorování datových toků, včetně popisu protokolu IPFIX. Následně si ve 4 také uvedeme architekturu zmíněného kolektoru *IPFIXcol2*, včetně provozních parametrů, které můžeme monitorovat při používání tohoto kolektoru.

Následuje kapitola 5 s popisem zásuvného modulu kolektoru,včetně popisu principů a řešených problémů při návrhu architektury a implementaci. Zaměříme se na export provozních parametrů pomocí protokolu SNMP, načež si uvedeme příklady využití tohoto exportu. V závěrečné kapitole 6 si otestujeme vliv zásuvného modulu na výkon kolektoru, popíšeme si využití modulu v praxi a následně shrneme budoucí možná rozšíření tohoto nového zásuvného modulu.

Kapitola 2

Monitorování a správa sítě

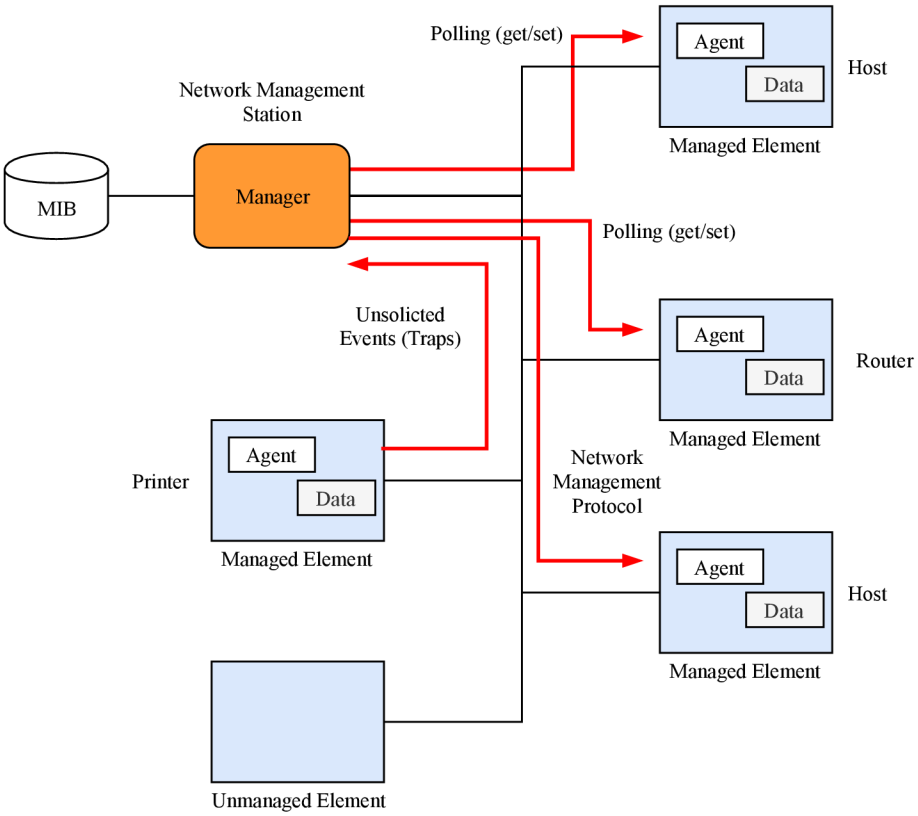
Počítačové sítě mohou v dnešní době nabývat různých rozměrů. Od malé domácí sítě, jejíž prvky je pouze několik síťových zařízení (počítač, tiskárna, směrovač) až po globální síť, která má tisíce zařízení. Jak ale probíhá správa těchto sítí, pokud se nám nabízí takto široká škála? U domácí sítě to většinou bývá takovým způsobem, že pokud nastane nějaký výpadek služby (např. síťová tiskárna netiskne dokumenty na ni zaslané), tak se až poté zjišťuje stav zařízení a problém je následně opraven. Tento způsob by u větších podnikových sítí mohl znamenat výpadky i v řádu hodin, podle toho, za jak dlouho by se správcům sítě podařilo problém odhalit. Zároveň by tyto výpadky mohly vést k dalším problémům. Představme si například situaci, kdy ve firmě přestane komunikovat server, na kterém běží webové aplikace, které zaměstnanci běžně používají, a tím pádem po dobu výpadku nemohou vykonávat svou práci. V případě, že by jako externí firma obstarávala služby firmě jiné (například účetnictví, správa informačního systému, ...), pak by tímto výpadkem ovlivnila i tuto další firmu.

Pro správnou funkčnost sítě je tedy důležité sledovat stav jejích prvků. Tím se dá předcházet kritickým stavům a zároveň tak získáváme obecný přehled o jejím aktuálním stavu. Na základě nasbíraných statistik poté můžeme síť rozšiřovat či zmenšovat a především opravovat provozní chyby, a to jak manuálně, tak automaticky. Dalším přínosem je i zvýšená bezpečnost sítě. Jak popisuje Richard Bejtlich ve svém článku *„The Self-Defeating networks“*[1] (v překladu „Sítě, které se zdolávají samy“), pokud administrátor nezná síť, o kterou se stará, nesleduje stav zařízení, která do ní patří, v síti neprobíhá kontrola datových toků, a tím se chování sítě stává neřízené, pak každá z těchto vlastností ještě více zvětšuje už tak existující bezpečnostní riziko. Pokud se však administrátor bude snažit tyto vlastnosti eliminovat, ztěžuje tak práci potencionálním útočníkům, a tím zvyšuje bezpečnost dané sítě.

V této kapitole si uvedeme způsoby a technologie, které se pro účel monitorování naskytují. Podrobněji si popíšeme protokol SNMP, který je v dnešní době nejpoužívanějším protokolem pro správu a sledování stavu sítě, vysvětlíme si jeho principy a způsoby nasazení do počítačové sítě.

2.1 Systém pro správu sítě

Obecně je systém definován jako „*množina elementů (prvků), které jsou mezi sebou vázány nějakým vztahem, respektive vazbou*“[19]. V případě systému pro správu sítě jsou hlavními prvky řídící jednotka (angl. *Managing Entity*) a spravovaná zařízení (angl. *Managed Entity*). Ty spolu komunikují pomocí protokolu pro správu sítě (angl. *Network Management Protocol*). Schéma tohoto systému je znázorněno na obrázku 2.1, včetně vazeb mezi těmito prvky. Informace v této sekci jsou čerpány primárně z knihy od Petra Matouška „*Síťové aplikace a jejich architektura*“[13], kapitola 10 – Správa sítě.



Obrázek 2.1: Schéma architektury systému pro správu sítě [13]

- **Řídící jednotka (Managing Entity)** Centrální bod systému pro správu sítě, na které agenti zasílají své stavové zprávy a odkud probíhá řízení sítě. Stavové zprávy jsou zde shromažďovány a využity pro analýzu, grafické znázornění naměřených hodnot a prezentaci administrátorovi. Pro jednodušší práci s daty bývá řídící jednotka opatřena grafickým uživatelským rozhraním, které administrátorovi umožňuje jednoduché vyčítání hodnot a stavu jak celé sítě, tak i jejích jednotlivých prvků.
- **Spravovaná zařízení (Managed Entity)** Konkrétní zařízení v síti, která chceme vzdáleně spravovat a sledovat jejich stav. Každé z těchto zařízení se skládá ze spravovaných objektů (angl. *Managed Objects*), o kterých řídící jednotce odesílá informace. Informace si lokálně uchovává v databázi MIB (angl. *Management Information Base*). Další podstatnou součástí spravovaného zařízení je agent (angl. *Network Management Agent*), pomocí kterého komunikuje s řídící jednotkou.

- **Komunikační protokol (Network Management Protocol)** je standard, na základě kterého probíhá komunikace mezi řídící jednotkou a spravovaným zařízením. Jedním z jeho hlavních cílů je zefektivnit přenos dat z databáze informací na spravovaném zařízení na řídící jednotku. Proto definuje přesný formát zpráv a také příkazy, které jsou v této komunikaci použity. Samotná komunikace pak může probíhat buď v režimu vyzývání k extrakci dat z databáze (angl. *polling*), při kterém komunikaci započíná řídící jednotka dotazováním se na informace z databáze informací konkrétního agenta, který mu odpovídá. Druhým režimem je ohlašování událostí (angl. *event reporting*), při kterém naopak komunikaci zahajuje agent tím, že na řídící jednotku vyšle zprávu indikující určitou událost (například výpadek služby).

Systém pro monitorování a správu sítí je systémem distribuovaným, tzn. jeho prvky jsou od sebe vzdálené a propojeny sítí. V obecné teorii systémů by se tento systém dal nazvat systémem centralizovaným. Primárním účelem je sledování stavu spravovaných zařízení a jejich vzdálená správa. Další informace, které systém poskytuje, se dají využít k plánování zdrojů využitých v síti, detekci útoků nebo analýze chování jejích uživatelů. K tomu je využita sada nástrojů a programů, která tyto činnosti ulehčuje.

2.2 Monitorování síťových zařízení

Monitorování neboli sledování stavu síťových zařízení se především rozděluje do dvou kategorií, a to monitorování **pasivní** a **aktivní**[14].

Pasivní monitorování probíhá tak, že v určitém bodě sítě sledujeme datové toky, které tímto bodem prochází. Následnou analýzou nasbíraných dat poté zjišťujeme stav a charakteristiky pro daný segment sítě. Jimi mohou být aktuální vytíženost zařízení, používané komunikační protokoly nebo například neoprávněné přístupy či snaha o přetížení služby[1]. Při použití této metody naměřená data nejsou nijak výrazně zkreslena přítomností monitorovacího systému (v tom smyslu, že při použití monitorování by naměřené statistiky na zařízení měly s malými odchylkami odpovídat statistikám naměřeným bez použití monitorování).

Oproti tomu při **aktivním monitorování** se do sítě vysílá monitorovací provoz, kterým zjišťujeme dostupnost a stav zařízení, a tímto naměřená data ovlivňujeme. Tento provoz by se také dal nazvat *dotazováním*, které započíná administrátor, a to buď periodicky, či neplánovaně. Při aktivním monitorování je však vhodné dbát na to, aby nedošlo k přetížení zařízení právě monitorovacím provozem. Zde se proto volí kompromis mezi co nejaktuálnějšími daty a co nejmenší zátěží sítě.

Pro porozumění chování a stavu sítě bývá zapotřebí použít kombinaci obou výše zmíněných technik. Sledováním datových toků odvozovat aktuální stav a v případě pochybností či výskytu události aktivním dotazováním zjistit dostupnost poskytovaných služeb. Touto událostí nemusí být pouze výpadek, ale i například rozšíření sítě o nová zařízení nebo přenastavení směrovacího protokolu. Podrobnější popis monitorování datových toků si uvedeme v kapitole 3.

[1] Známé jako Denial of Service (DoS)

Sběr stavových informací

K informacím o aktuálním stavu zařízení se můžeme dostat několika způsoby. Ať už je to výpis pomocí příkazové řádky, shromažďování logů[2] ze zařízení či použití monitorovacího protokolu, jako je například protokol SNMP (viz sekce 2.3). Který z těchto způsobů použijeme, záleží především na podpoře od výrobce zařízení. Z tohoto hlediska je tedy důležité zvážit nabízené technologie již při výběru zařízení. Poté záleží na monitorovací aplikaci, jakým způsobem nashromážděná data prezentuje. Pokud administrátorovi stačí pouhý výpis aktivit, může například použít protokol *syslog*, díky kterému síťová zařízení shromažďují své výpisy v centrálním úložišti. Pokud chce provádět porovnání dat, jejich grafické znázornění a další operace nad nimi, je vhodné použít protokol SNMP. [13]

2.3 Protokol SNMP

Historie tohoto protokolu sahá do roku 1988, kdy byla definována jeho první verze, která se stala v roce 1990 internetovým standardem. Následovala verze druhá, definovaná v roce 1996, která opravovala nedostatky první verze a navíc v ní byly definované nové příkazy. Poslední a také nejaktuálnější třetí verze byla vydána v roce 1999 a standardizována v roce 2004. Oproti dvěma předchozím verzím nabízí především zvýšenou bezpečnost. Základní rozdíly si podrobněji popíšeme dále v této sekci.

Jak již slovo *simple* v názvu protokolu „*Simple Network Management Protocol*“ napovídá, od počátku byla cílem jeho jednoduchost. V RFC 2571 [17] je dokonce přímo jedním z hlavních cílů „*Udržet protokol SNMP co nejjednodušší, jak to jen jde*“. Důvodem je lehké nasazení tohoto protokolu bez větších nákladů, aby bylo použití tohoto protokolu výhodné i v menších sítích. Tento přístup se opravdu osvědčil a první dvě verze se po svém vydání staly nejpoužívanějším protokolem pro správu sítí [16].

2.3.1 Architektura SNMP

SNMP protokol splňuje základní požadavky pro obecný systém správy sítě, popsané v sekci 2.1. Tedy skládá se z **manažerů**, **agentů** a **komunikačního protokolu**. Ovšem jediným rozdílem, který je v tomto případě výhodou, je snaha o decentralizovaný model. Tím je myšleno, že v tomto systému může existovat více než jeden manažer. Výhoda je především v možnosti rozdělit velké sítě na menší monitorované celky, přičemž každý bude mít svého síťového manažera, a tím se rozloží vytížení řídící jednotky na více zařízení. Další výhodou je instalace záložních manažerů, kteří mohou v síti koexistovat spolu s hlavními manažery bez toho, aniž by narušovali jejich funkčnost.

V referenčním modelu TCP/IP se tento protokol pohybuje na úrovni aplikační vrstvy. Při komunikaci využívá k přenosu zpráv nejčastěji transportní protokol UDP, který i v této práci je považován za hlavní transportní protokol pro SNMP, ale i použití TCP je možné. Transportní protokoly využívají porty 161 pro UDP a 162 pro TCP, které jsou zaregistrované organizací IANA[3].

Příkazy, které protokol SNMP používá v komunikaci jsou následující:

- **GetRequest** – Požadavek o zaslání dat. Tento požadavek zasílá manažer agentovi a specifikuje v něm přesné identifikátory dat, která mu mají být zaslána jako odpověď.

[2] Log = Stavový výpis zařízení

[3] Seznam registrovaných portů je na adrese `https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml`

Může se jednat o jednu hodnotu, ale také i o celý seznam hodnot, který je přenášen v rámci jedné transakce. V případě, že pro daný identifikátor hodnota není dostupná, dostává se manažerovi odpovědi s chybovým kódem.

- **GetNextRequest** – Tento příkaz je svou povahou podobný příkazu GetRequest. Stejně jako on slouží k dotazu na hodnotu proměnné, ale liší se v tom, že se dotazuje na další dostupnou hodnotu. Tento mechanizmus je využitelný v případě, že manažer nemá přehled o dostupných hodnotách na agentovi a chce zjistit, jaké hodnoty monitorovaných objektů jsou k dispozici. Například pro průchod všemi dostupnými hodnotami všech dostupných sloupců v konkrétní tabulce manažer zasílá žádost s identifikátorem tabulky. Agent poté musí najít první hodnotu prvního dostupného sloupce a data mu zasílá v odpovědi společně s identifikátorem zaslané hodnoty. Manažer poté vykonává další GetNextRequest příkaz, ve kterém již specifikuje identifikátor předchozí zaslané hodnoty a tím opakuje celý proces. Takto může postupně projít celou tabulku. Samotný příkaz však nespecifikuje automatický průchod, pouze zaslání další dostupné hodnoty.

- **SetRequest** – Nastavení hodnoty. Příkaz, používaný pro vyzvání agenta k přenastavení jedné nebo více hodnot monitorovaných objektů na hodnoty nové, specifikované manažerem. K úspěšnému vykonání tohoto příkazu na agentovi je třeba, aby požadovaná hodnota byla přístupná k přenastavení (nesmí být tedy pouze pro čtení). V případě pokusu o přenastavení hodnoty pouze pro čtení je vykonání příkazu neúspěšné, o čemž agent manažera informuje v odpovědi.

- **GetResponse** nebo také pouze **Response** – Odpověď agenta manažerovi. Obsahuje buď data, která si manažer vyžádal jedním ze dvou předchozích Get příkazů, nebo chybový kód, kterým informuje například o absenci dotazovaných dat. Dalším účelem je i odpověď na příkaz Set, kdy je tato zpráva buďto potvrzením úspěšného přenastavení hodnoty, nebo opět obsahuje chybový kód pro identifikaci problému.

- **Trap** – Příkaz určený k ohlášení události. Oproti předchozím příkladům se používá v komunikaci, kterou započíná agent. Ten musí mít pro zaslání zprávy specifikovaný seznam IP adres manažerů, kterým tuto zprávu zašle. O jaké události bude agent manažera informovat, se specifikuje na agentovi pomocí nastavení pravidel. Pravidla se mohou týkat jak hodnot monitorovaných objektů, tak i dalších stavů monitorovaného zařízení

Všechny výše zmíněné příkazy byly definovány již v první verzi SNMP. Poté, co se protokol začal široce využívat se však přišlo na to, že trpí několika nedostatky. Opomenemeli slabé zabezpečení komunikace, nedostatky nastaly právě v dotazování se na velký počet hodnot monitorovaných objektů a v komunikaci manažerů mezi sebou.

Dotazování na velký počet hodnot nastalo v případě průchodu tabulkami a shromažďování všech hodnot z jednoho sloupce najednou. Pro každou hodnotu musel manažer vytvořit GetRequest nebo GetNextRequest příkaz, a tím vznikala nadbytečná komunikace ze strany manažera. Pokud například agent obsahoval tabulku s 5 sloupci a třemi záznamy, musel manažer vytvořit 15 požadavků na tyto hodnoty a ty agentovi zaslat. Předpokladem pro další verzi se tedy stal i fakt, že manažer bude vyžadovat všechna data bez nutnosti automatizace vytváření dotazů na každou hodnotu zvlášť. V druhé verzi je tento nedostatek napraven příkazem **GetBulkRequest**.

V případě SNMPv1, jak je označena první verze, bylo sice možné rozdělit síť na menší monitorované celky s větším počtem manažerů. Tito však museli mezi sebou komunikovat pomocí nestandardizovaných zpráv a protokolů a tak si vzájemně ohlašovat situace a události v jiných monitorovaných oblastech. Tento problém byl vyřešen ve verzi 2 doplněním příkazu **InformRequest**. [10]

- **GetBulkRequest** Příkaz dotazující se na všechny dostupné hodnoty monitorovaných objektů na agentovi. Jeho využitelnost je právě ve zmíněných průchodech. Manažer zasílá jedinou zprávu s tímto příkazem a přesným identifikátorem začátku průchodu. Agent poté shromažďuje data jak pro daný identifikátor, tak i pro všechny identifikátory následující v lexikografickém uspořádání[4] po tomto identifikátoru. Nashromážděná data poté odesílá seskupená v odpovědích zpět manažerovi. Tímto se šetří počet přenášených zpráv a vytížení linky, po které spolu manažer a agent komunikují.

- **InformRequest** Jediný z příkazů, jehož původním účelem bylo využití v komunikaci manažer–manažer. Informuje ostatní manažery o události, která nastala na jiném manažerském zařízení. Touto událostí je zde myšleno překročení předem nastavených limitů pro určitý monitorovaný objekt. Není tedy třeba specifikovat seznam informovaných manažerů na agentovi, na kterém se monitorovaný objekt nachází, ale stačí tuto entitu monitorovat jediným manažerem, který poté informuje manažery další. Toto je výhodné v situaci, kdy manažery uspořádáme do hierarchie, která nám poskytuje různé stupně abstrakce stavu sítě. Čím níže se budeme v hierarchii pohybovat, tím podrobnější informace o stavu sítě budeme dostávat. Později se tento typ příkazu začal používat i jako potvrzování v komunikaci manažer–agent například pro příkaz SetRequest.

2.3.2 Zabezpečení

Zpočátku bylo zabezpečení SNMP velmi opomíjené. Prvotním předpokladem bylo, že tento monitorovací systém bude převážně nasazován v privátních sítích bez toho, aby jeho prvky spolu komunikovaly přes Internet. Proto se v SNMPv1 definovalo pouze zabezpečení na základě tzv. *community string*, což je pouhý textový řetězec, který se musel specifikovat u agentů a manažerů a ti tento řetězec obsáhli do odesílaných zpráv. Prvek, s nímž probíhala komunikace, pak po přijetí zprávy provedl porovnání přijatého a interního řetězce, a pokud byly stejné, provedl požadovaný příkaz. Problém nastal v tom, že textový řetězec se po síti přenášel v otevřeném formátu a každý, kdo by na síti naslouchal, by k němu měl snadný přístup. Nejenže by pak mohl řetězec použít k zasílání příkazů dotazující se na data na agentech, ale především mohl zasílat SetRequest příkazy, kterými by mohl přenastavit hodnoty monitorovaných objektů, a tím zkreslit nasbírané statistiky na manažerech či dokonce přímo ovlivnit chod celé sítě. Jedinou ochranou proti tomuto útoku tedy bylo zakázání SetRequest příkazů na agentovi, čímž ovšem manažer přišel o možnost vzdálené správy monitorovaných objektů pomocí SNMP.

Řešení této situace se objevilo s novou verzí SNMPv2, kde byl nově zaveden model autentizace a šifrování dat[9]. Při autentizaci se využilo *Digest Authentication* algoritmu MD5, při kterém si komunikující entity zasílají časové známky (angl. *timestamps*) a otisk. A k šifrování dat bylo využito symetrického algoritmu DES, kde komunikující entity šifrovaly a dešifrovaly data na základě stejného klíče. Tento model se však příliš neujal kvůli své komplexnosti, a proto vyšla aktualizace protokolu s názvem SNMPv2c. Tento model opět

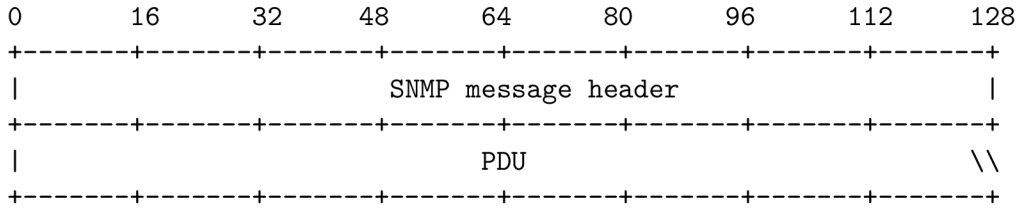
[4] Slovníkové řazení, neboli „podle abecedy“[18]

používal textových řetězců, jak tomu bylo u verze první. Verze SNMPv2c se také následně stala standardem. Kompromisem mezi těmito dvěma přístupy v zabezpečení se stala verze SNMPv2u, která nabízela *user-based* model pro ověření komunikace mezi dvěma entitami. Tento model se začal až dále rozšiřovat ve verzi 3.

Konečné řešení zabezpečení přinesla až verze SNMPv3. Ta nabízela jak uživatelský model pro ověřování přístupu, tak autentizaci pomocí MD5 a SHA algoritmů a šifrování dat algoritmy AES a DES. Aby si správce sítě mohl sám vybrat úroveň zabezpečení SNMPv3 zpráv, je možné využít varianty bez autentizace a šifrování, pouze autentizaci a nebo autentizaci i šifrování. Tím se výrazně zvětšila škála využitelnosti jedné konkrétní verze protokolu pro různé požadavky správců sítě.

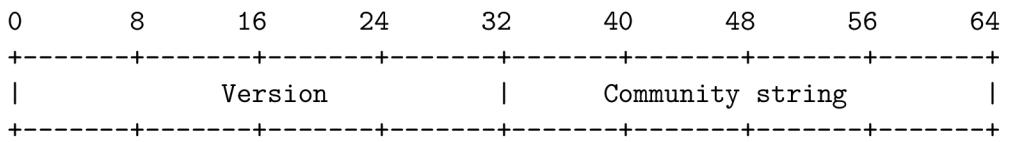
2.3.3 Formát SNMP zprávy

Struktura SNMP zprávy se skládá z hlavičky zprávy a těla, kde se nachází přenášená data. Největší rozdíl ve formátu zprávy byl mezi verzemi SNMPv2c a SNMPv3. A to v hlavičce, kde se pole pro Communtity string, definované v SNMPv2c, nahradilo dvěma políčky msgGlobalData a msgSecurityParameters. Za hlavičkou následuje datová jednotka, nazývaná PDU. Podrobnější popis formátování SNMP zprávy i s názornými příklady popisuje Douglas Bruey ve článku „*SNMP: Simple? Network Management Protocol*“[2], ze kterého jsem čerpal informace pro tuto podsekci.



Obrázek 2.2: Formát SNMP zprávy

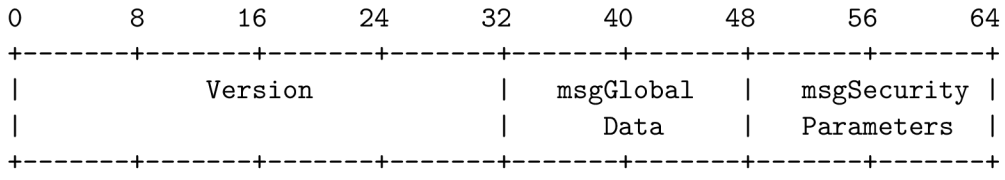
Hlavička zprávy



Obrázek 2.3: Hlavička SNMPv2 zprávy

- **Verze** Použitá verze SNMP protokolu, která odpovídá i jeho číselné hodnotě v názvu. Pouze verze SNMPv1 je identifikována číslem 0.
- **Community string** Textový řetězec určený pro autentizaci (více o zabezpečení viz podsekce 2.3.2).

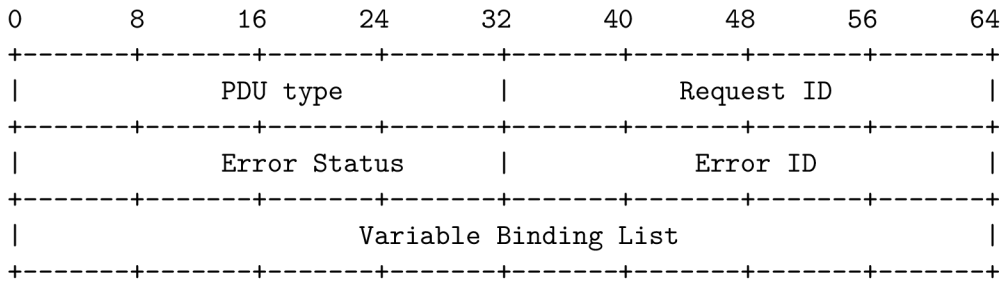
- **msgGlobalData** a **msgSecurityParameters** jsou pole určená pro autentizační algoritmy MD5 a SHA.



Obrázek 2.4: Hlavička SNMPv3 zprávy

Tělo zprávy

Tělo SNMP zprávy je tvořeno datovou jednotkou PDU (Protocol Data Unit). Ta také obsahuje hlavičku, kde jsou údaje o typu PDU, ID žádosti a poté následuje seznam vazeb proměnných (variable binding list), který obsahuje vazby (variable bindigs), skládající se z identifikátoru objektu OID (Object Identifier) a hodnoty proměnné tohoto objektu.



Obrázek 2.5: Hlavička PDU

- **PDU type** Typ PDU. V tomto poli je číselně identifikován typ zprávy, které dané PDU odpovídá. Číselné identifikátory jsou náledující:

  0 GetRequest

  1 GetNextRequest

  2 GetResponse

  3 SetResponse

  4 *nepoužívané*

  5 GetBulkRequest

  6 InformRequest

  7 Trap

- **Request ID** Číselný identifikátor požadavku/odpovědi. Jeho funkcí je správné přiřazení odpovědi k požadavku. Zařízení, které odpovídá na požadavek, pouze zkopíruje

hodnotu tohoto pole do odpovědi, na základě čehož dotazující zařízení rozpozná, které požadavky již byly úspešně zodpovězeny a které je třeba poslat znovu například z důvodu ztráty paketu při použití transportního protokolu UDP.

- **Error Status** Identifikátor chybového stavu PDU. Při žádosti je hodnota tohoto pole nastavena na 0. Zařízení, které odpovídá, poté tuto hodnotu v odpovědi nastaví dle chybového stavu, který nastal. Příklad číselných identifikátorů a jim náležících chybových stavů:

  0 noError – žádná chyba nenastala

  1 tooBig – Hodnota proměnné je příliš velká

  2 noSuchName – Proměnná požadovaného jména neexistuje

  3 badValue – Špatný datový typ požadované proměnné

  4 readOnly – Proměnná je pouze pro čtení

  5 genError – značí výskyt chybového stavu, který neodpovídá definovaným stavům.

- **Error ID** v případě výskytu chybového stavu je v tomto poli hodnota identifikátoru proměnné, která daný stav vyvolala.

- **Variable Binding List** Seznam vazeb proměnných. Jeho obsahem jsou dvojice identifikátorů OID a hodnot proměnných. Délka tohoto seznamu je proměnlivá, není tedy pevně určena ve specifikaci protokolu, ale je pro každý seznam přesně určena přímo ve zprávě.

2.3.4 MIB – Management Information Base

Popis MIB databáze, notace ASN.1 a kódování BER, které si popíšeme v této podsekci, je podrobněji uveden v knize „*LAN Mangement with SNMP and RMON*“ [10] od Gilberta Helda, ze které jsem primárně čerpal informace pro celou tuto podsekci.

MIB je databází informačních prvků. Struktura i prvky této databáze jsou standardizované. Má hierarchickou strukturu objektů, přičemž každý z těchto objektů je identifikován pomocí OID (Object Identifier), který je ve tvaru `kořen.uzel.uzel.....uzel.list`. Čím níže se budeme v této hierarchii pohybovat, tím delší bude výsledný identifikátor. Například identifikátor sloupce pro název rozhraní je

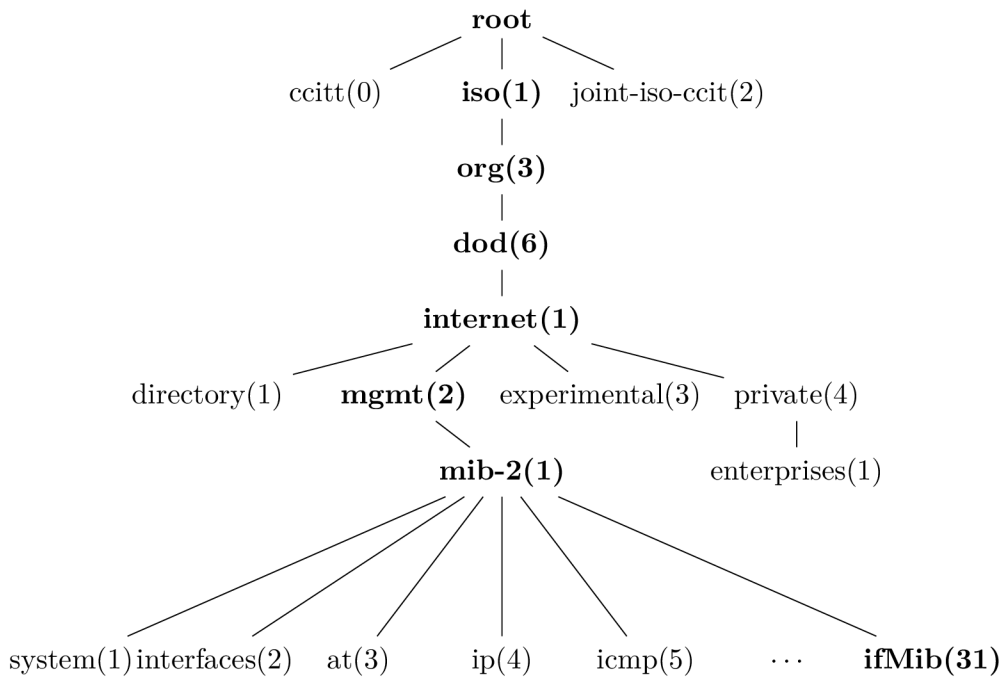
```
1.3.6.1.2.1.31.1.1.1.18.
```

Jelikož z těchto číselných identifikátorů na první pohled nepoznáme, o kterou informaci se jedná, je každému uzlu ve struktuře MIB přiřazen i textový název. Předchozí identifikátor lze tedy vyjádřit i jako identifikátor

```
iso.org.dod.internet.mgmt.mib-2.ifMIB.ifMIBObjects.ifXTable.ifXEntry.ifAlias
```

kde již vidíme dle názvu listového uzlu, že se jedná o jméno rozhraní. Pro získání konkrétní hodnoty je třeba k těmto identifikátorům ještě přidat index konkrétního záznamu. Grafické znázornění hierarchie MIB databáze lze vidět na obrázku 2.6.

Protokol SNMP používá MIB k vyčítání uložených hodnot na agentech. Manažer obsahuje popis té části MIB, kde se nachází popis informací, o které má zájem, a tím pádem může odesílat identifikátory proměnných v dotazech na agenta. Na agentovi nebývá popsána celá MIB, ale pouze její části, dle dostupnosti monitorovaných objektů. Pro každý



Obrázek 2.6: Grafické znázornění cesty k uzlu *ifMib* v hierarchii MIB

objekt má interně uloženou proměnnou, která je spjata se identifikátorem hodnoty objektu. Při obdržení požadavku od manažera tedy agent postupně rozpozná, o kterou proměnnou se manažer zajímá, zkontroluje, zdali má pro tento identifikátor definovanou proměnnou s hodnotou, a v případě že má, zasílá odpověď s identifikátorem a hodnotou.

ASN.1

Databáze MIB je standardně popsána jazykem ASN.1 (angl *Abstract Syntax Notation One*). Tento jazyk slouží k obecnému popisu datových struktur, přičemž při návrhu se nebere v potaz implementace. Díky separaci popisu datových struktur a jejich implementace můžeme použít notaci ASN.1 pro popis datových struktur nezávisle na protokolu. Ukázku této notace můžeme vidět na obrázku 2.7.

Při popisu datových struktur se používá několik základních jednoduchých a strukturovaných datových typů, ze kterých se dají vytvářet nové, specifičtější datové typy. Ukázku názvů a čísel základních jednoduchých a strukturovaných datových typů můžeme vidět v tabulce 2.1.

```
-- top-level message

            Message ::=
                    SEQUENCE {
                        version        -- version-1 for this RFC
                            INTEGER {
                                version-1(0)
                            },

                        community      -- community name
                            OCTET STRING,

                        data           -- e.g., PDUs if trivial
                            ANY        -- authentication is being used
                    }
```

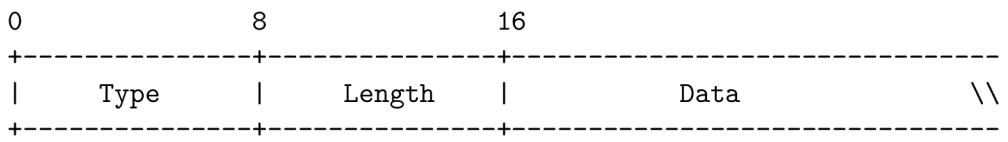
Obrázek 2.7: Popis hlavičky SNMP v notaci ASN.1[3]

Název	Číslo tagu
BOOLEAN	1
INTEGER	2
OCTET STRING	4
NULL	5
OBJECT IDENTIFIER	6
SEQUENCE	16
SET	17

Tabulka 2.1: Ukázka několika základních datových typů notace ASN.1

BER – Basic Encoding Rules

Tímto názvem je popsána sada pravidel pro formátování dat, která jsou proměnlivé délky. Formát BER využívá tzv. TLV kódování (Type Length Value). Skládá se tedy ze tří částí, přičemž v první je specifikován typ dat, ve druhé jejich délka a ve třetí části jsou již samotná data. Formát BER se používá k formátování abstraktních informací jazyka ASN.1. Celá SNMP zpráva využívá toto formátování, jelikož i formát SNMP zprávy je popsán jazykem ASN.1, především se však používá k formátování hodnot proměnných pro přenos po síti.



Obrázek 2.8: BER formát dat

- **Type** Označení typu dat. Samotný oktet pro typ dat se skládá z dalších tří částí. V prvních dvou bitech se nachází informace, zdali je typ univerzální, privátní, či specifický pro aplikaci nebo v určitém kontextu. V dalším bitu je obsažena informace o tom, jestli se jedná o typ strukturovaný, či jednoduchý a v posledních 5 bitech je již číselné označení datového typu[5] neboli tag. Příklad kódování si uvedeme na strukturovaném datovém typu SEQUENCE, užívaném k přenášení sekvence hodnot. Ten má číselné označení 16 dekadicky. V prvních dvou bitech označíme, že se jedná o univerzální datový typ (hodnota 0), bit informující o strukturovaném datovém typu nastavíme na hodnotu 1 a číselnou hodnotu typu nastavíme na hodnotu 16. Tím nám vznikne výsledná hodnota označující typ dat, která se rovná 0011000 binárně nebo 30 hexadecimálně.
- **Length** Délka přenášených dat v bajtech. První dva bajty (tedy předchozí bajt pro typ dat a tento bajt pro délku dat) nejsou započítány do výsledné hodnoty.
- **Data** Samotná přenášená data.

[5]Číselná označení univerzálních datových typů jsou popsána například na této stránce `http://luca.ntop.org/Teaching/Appunti/asn1.html`

Kapitola 3

Monitorování datových toků

Představme si situaci, kdy jsme správci velké sítě čítající stovky zařízení a chceme mít přehled o dostupnosti služeb, které naše síť nabízí. Častým periodickým dotazováním na stav všech elementů v síti bychom mohli zvýšit provoz a vytížení sítě, což by mohlo mít negativní efekt právě na dostupnost služeb. Měli bychom sice přehled o tom, že určitý server je zapnutý, nepřehřívá se a služba na něm funguje, ale zbytečně často bychom zabírali přenosové pásmo, čímž bychom mohli zpomalit připojení uživatelů. I za tímto účelem vzniklo monitorování datových toků. Díky tomu, že do strategicky zvolených míst na síti rozmístíme „sondy“, které sledují procházející provoz daným místem, získáme informace například o tom, které servery komunikují s kým, pomocí jakého protokolu nebo jaký objem dat zasílají. Tak můžeme vidět, že servery jsou aktivní a interval, ve kterém se dotazujeme na aktuální stav zařízení, můžeme prodloužit.

Dále nám toto monitorování pomáhá k analýze chování uživatelů, a to jak interních, tak i externích. V případě interních uživatelů, například zaměstnanců ve firmě, můžeme zjišťovat z nasbíraných dat například, kolik stanic se připojuje na externí webové servery nebo stahuje soubory ze vzdáleného serveru pomocí FTP protokolu. Tyto informace můžeme poté použít například k nastavení pravidel na firewallu. V případě chování externích uživatelů stojí nejvíce za zmínku ochrana před útoky. Tím, že budeme monitorovat správná místa v naší síti, přes která prochází všechen provoz k určitému serveru, můžeme zabránit potencionálním útočníkům v útoku na tento server. Pokud budeme analyzovat nasbíraná data, z čehož si vyvodíme běžné chování uživatelů, můžeme být upozorněni v případě anomálií a ochránit síťové prvky před výpadkem.

V této kapitole se proto zaměříme na **pasivní** monitorování. Popíšeme si princip monitorování datových toků a protokoly NetFlow a IPFIX, které se k tomuto účelu využívají. Zaměříme se především na protokol IPFIX, rozebereme si formát jeho zpráv a způsob používání. Podrobnější informace o monitorování datových toků poskytuje článek „*Flow Monitoring Explained: From Packet Capture to Data Analysis with NetFlow and IPFIX*“[11], který byl hlavním zdrojem informací pro tuto kapitolu.

3.1 Architektura systému

Předtím, než začneme s popisem jednotlivých komponent, je vhodné si vysvětlit, co je to datový tok. Ten je popsán jako „*Set IP paketů procházející určitým bodem v určitý čas, přičemž všechny pakety patřící k jednomu datovému toku mají stejné, předem definované parametry.*“[11]. Pod pojmem „stejné parametry“ je myšleno například IP adresa a port příjemce a odesílatele a další metadata z hlaviček paketů.

Pro monitorování datových toků potřebujeme tři hlavní komponenty. První z nich byla již zmíněna v abstraktu této kapitoly, tedy **sonda**. Sondy se nasazují na místa, kde má smysl sledovat procházející datové toky. Taková místa se nazývají bod pozorování (angl. *Observation Point*) a sonda může být v tomto bodě nasazena přímo do cesty datových toků nebo na ni mohou být odesílány kopie aktuálního provozu. Jedním z takových míst je například hraniční směrovač (angl. *Border Router*), který spojuje lokální síť s dalšími sítěmi. Sonda pro svoji klíčovou roli nemusí provádět DPI (angl. *Deep Packet Inspection*), tedy nemusí nahlížet do přenášených dat, pouze do hlaviček paketů a z nich extrahuje potřebné informace k identifikaci datového toku. Tím se monitorováním nezasahuje do soukromí uživatelů tolik jako v případě DPI. Sonda je též označována jako exportér a to z toho důvodu, že exportuje nasbírané datové toky na kolektor.

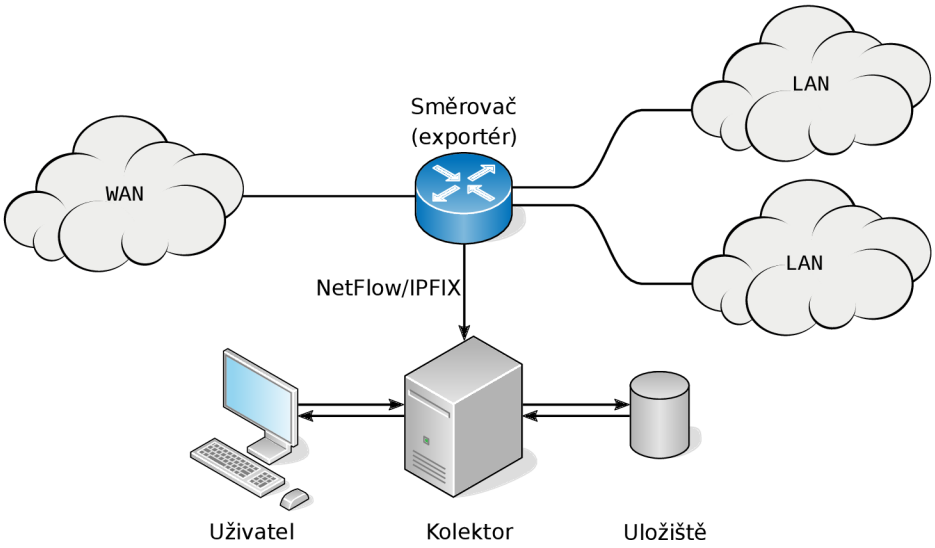
Poté, co sonda úspěšně přiřadí zkoumané pakety do datového toku a tok je považován za ukončený, jsou informace ohledně datového toku zaslány na další zařízení v tomto systému, čímž je **kolektor**. Ten je centrálním bodem pro sběr informací o datových tocích. Tyto informace jsou definovány jako „Informace, které byly o datovém toku nasbírány v určitém pozorovacím bodě“. Těmi mohou být nejen informace popsané u definice sondy, tedy informace z hlaviček paketů, ale i naměřené informace jako třeba počet paketů patřících k datovému toku. Kolektor přijaté informace ukládá k následné analýze.

Poslední nezbytnou částí je **protokol**, který umožňuje efektivní a standardizovaný způsob přenosu informací o datových tocích ze sondy na kolektor. Efektivní způsob přenosu je vyžadován kvůli tomu, že bod pozorování se může vyskytovat v rámci vysokorychlostní sítě, a tudíž má potenciál přenášet velké množství informací o datových tocích. Pro tento účel byly vyvinuty a standardizovány protokoly, mezi které patří například NetFlow od firmy Cisco, nebo IPFIX, který byl vyvinut organizací IETF.

Informace přenášené protokolem mohou být různorodé a jinak strukturované. Proto jsou standardem[5] definované tzv. *Informační elementy* (*Information Elements*), pomocí kterých se identifikují přenášená data ve smyslu jejich významu a datového typu. Každý informační element je definován pomocí jména, unikátního identifikačního čísla, datového typu, popisu a stavu platnosti. O databázi informačních elementů se stará organizace IANA (Internet Assigned Numbers Authority). V případě potřeby přenášet z exportéru jiné informace než ty popsané pomocí informačních elementů je možnost zažádat organizaci IANA o vlastní rozsah definovaný privátním číslem organizace PEN (Private Enterprise Number). Díky privátnímu rozsahu si můžeme nadefinovat vlastní informační elementy, a to bez procesu schvalování každého z elementů.

3.2 Protokoly pro monitorování datových toků

Vývoj protokolů pro monitorování datových toků začal roku 1991[11], kdy bylo poprvé popsáno shromažďování paketů do datových toků na základě informací z hlaviček paketů. Po nějaký čas však vývoj stagnoval. Jednak kvůli malému zájmu od výrobců síťových



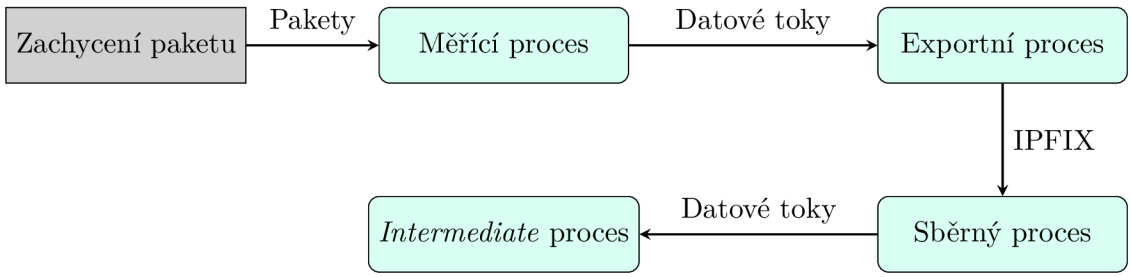
Obrázek 3.1: Architektura systému pro monitorování datových toků [12]

zařízení a také kvůli názoru, že Internet by měl být volný v tom významu, že by se v něm nemělo odehrávat sledování síťového provozu jak za účelem monitorování, tak i za účelem zpoplatnění služeb. Následovaly další pokusy o popsání monitorování síťového provozu za účelem analýzy, které ovšem opět selhaly kvůli nezájmu výrobců.

S prvním úspěšným protokolem přišla až firma CISCO, která vyvíjela privátní protokol NetFlow. Velmi oblíbenou verzí tohoto protokolu se stala verze NetFlow v5, která se stala v roce 2002 volně dostupná[11]. V této verzi protokolu byla přesně definována pole informačních elementů, které bylo možné pomocí protokolu přenášet a pro základní monitorování datových toků to bylo dostatečné množství. Problém nastal až v situaci, kdy správce potřeboval přenést i jiné informace než ty popsané v protokolu. V tomto směru byl protokol NetFlow v5 omezený.

Po protokolu NetFlow v5 následovalo několik dalších verzí, ale právoplatným nástupcem se stal protokol NetFlow v9, a to v roce 2004[11]. Nová verze protokolu NetFlow již nabízela větší flexibilitu pomocí použití šablon pro přenos informačních elementů. Správce si tím pádem mohl na exportéru zvolit, které dostupné informace bude chtít přenášet ze sondy na kolektor. Exportér poté vytváří šablonu pro přenos těchto nasbíraných informací a její definici periodicky zasílá kolektoru. Kolektor používá definice šablon pro parsovaní přijatých dat, která mají poznačené, která ze šablon byla použita k jejich přenosu.

Za účelem standardizace začala i organizace IETF vyvíjet protokol pro monitorování datových toků. Základem pro nový protokol se stal protokol NetFlow v9 od firmy CISCO. Nový protokol nazývaný IPFIX (IP Flow Information eXport) poprvé vyšel v roce 2008[4] a stal se internetovým standardem v roce 2013[6]. Oproti NetFlow v9 nabízí další výhody, jako například flexibilní definici datových toků, kdy datový tok může být definován na základě i dalších parametrů, než je pouze IP adresa a port zdroje a odesílatele a použitého protokolu. IPFIX tedy nabízí širokou škálu použitelnosti jak v aktuálních síťových architekturách, tak i v budoucnu.



Obrázek 3.2: Jednoduché uspořádání procesů v architektuře IPFIX

3.3 Protokol IPFIX

Architektura systému, ve kterém je použit protokol IPFIX, je popsána v RFC 5470 [15]. Skládá se ze základních komponent, které jsou popsány v obecné definici monitorování datových toků. Definovány jsou především tři typy procesů, a to **měřící** procesy (angl. *Metering Processes*), určené ke generování datových toků. Procesy **exportní** (ang. *Exporting Processes*), určené k zasílání zpráv ze sond na kolektor, a **sběrné** procesy (angl. *Collecting Processes*), k účelu sbírání zpráv na kolektoru od sond. Vztahy těchto procesů jsou 1:N, tedy například 1 exportní proces může přenášet informace o několika naměřených datových tocích na několik kolektorů.

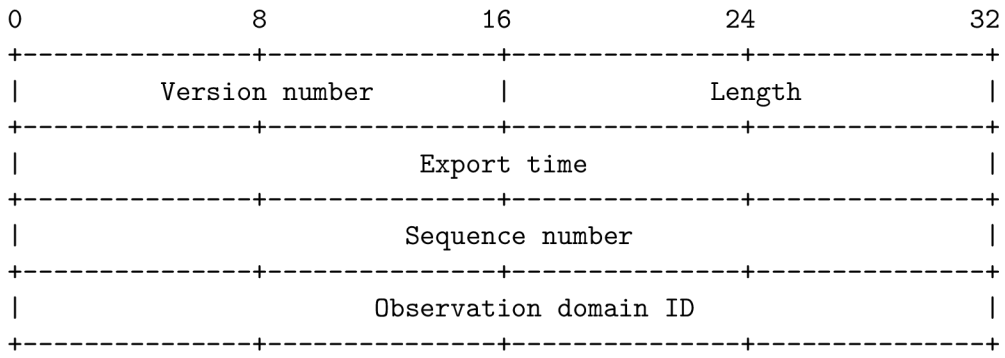
Po dalším vývoji IPFIX protoklu byl definován ještě čtvrtý typ procesů, a to procesy **pokročilé** (angl. *Intermediate Processes*). Ty se používají k úpravě informací o datových tocích, například k jejich anonymizaci či agregaci. Jelikož by český překlad názvu typu těchto procesů mohl být matoucí, budeme dále využívat anglický název, tedy *Intermediate* procesy. Názornou ukázku uspořádání všech čtyř typů procesů můžeme vidět na obrázku 3.2.

IPFIX zprávy a šablony

Zpráva protokolu IPFIX se skládá z hlavičky zprávy, kde jsou obsaženy informace o verzi protokolu, délce zprávy, času, kdy byla zpráva exportována ze sondy, sekvenční číslo pro správné pořadí exportovaných zpráv a číslo pozorovací domény. Ukázku hlavičky IPFIX zprávy můžeme vidět na obrázku 3.3. Pro protokol IPFIX je číslo verze rovno hodnotě 10 a kvůli tomu bývá také protokol IPFIX někdy nesprávně označován za NetFlow v10[1]. Dále za zmínku stojí i číslo pozorovací domény neboli ODID (angl. *Observation Domain ID*). Na sondách může totiž běžet několik měřících procesů, které měří jiné oblasti sítě. Představme si například směrovač, ke kterému jsou připojeny dvě různé oblasti monitorované sítě. Na směrovači tedy běží pouze jeden exportní proces, zatímco pro určité porty běží odlišné měřící procesy. Tyto měřící procesy jsou rozlišeny právě na základě ODID.

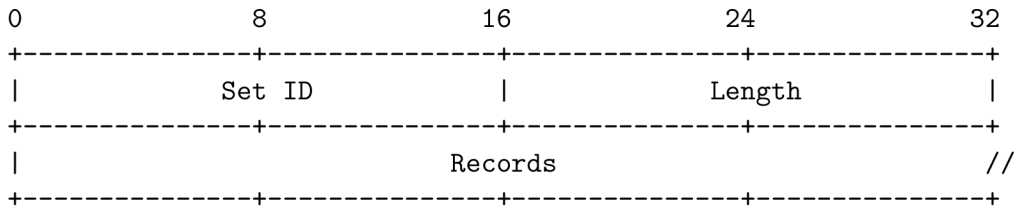
Za hlavičkou zprávy následuje seznam setů neboli sad. Každá z těchto sad obsahuje hlavičku, kde jsou informace o identifikátoru sady a její velikosti. Ukázka hlavičky sady je na obrázku 3.4. Hlavním prvkem v hlavičce sady je právě identifikátor, pomocí kterého můžeme identifikovat, zdali se jedná o sadu s definicí šablon, nebo o sadu se záznamy nesoucími informace o datových tocích. V případě sady, ve které se nachází definice šablon, je identifikátor sady roven hodnotě 2. V těle sady je poté seznam definic šablon, přičemž každá z nich musí obsahovat číslo šablony větší než hodnota 255. V případě sady obsa-

[1]NetFlow v5 má číslo verze 5 a NetFlow v9 má číslo verze 9.



Obrázek 3.3: Formát hlavičky IPFIX zprávy

hující šablony nastavení je identifikátor roven hodnotě 3. V případě identifikátoru většího než hodnota 255 se jedná o sady obsahující datové záznamy, přičemž identifikátor setu se v tomto případě rovná i identifikátoru šablony, pomocí které byla data naformátována do zprávy. Identifikátory z rozsahu 4–255 se momentálně nepoužívají a jsou určeny pro budoucí využití. Identifikátory 0–1 již byly využity u protokolu NetFlow v9.



Obrázek 3.4: Formát IPFIX setu

Kapitola 4

Kolektor IPFIXcol2

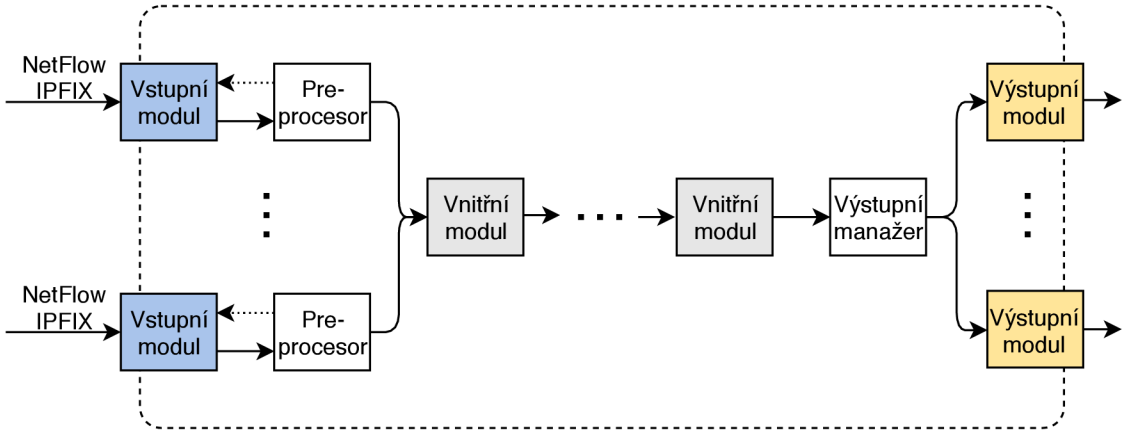
Firmy, které se zaměřují na poskytování monitorovacích komponent, dnes nabízí kolektory, které mají různé vlastnosti a schopnosti. V této kapitole se však zaměříme na konkrétní kolektor a to *IPFIXcol2*, který je vyvíjen organizací CESNET. Důvodem je jeho volná dostupnost a také fakt, že autor této práce se spolupodílí na jeho vývoji, a tudíž je se strukturou kolektoru obeznámen. Jeho předchůdcem byl kolektor *IPFIXcol*, který ovšem trpěl nedostatky v jeho rozšiřitelnosti, malým počtem testů a slabou dokumentací[12]. Proto vznikl kolektor nový, jehož hlavní myšlenkou při vývoji bylo heslo „*Znovu a lépe*“. Tento přístup se osvědčil a aktuální architektura kolektoru poskytuje rozhraní pro jednoduchou rozšiřitelnost. Mimo jiné je i ke kolektoru vytvořena knihovna *libfds* neboli *Flow data storage library*, která je přímo určená k jednoduché práci s datovými strukturami používanými v kolektoru *IPFIXcol2*. Hlavním zdrojem informací pro tuto kapitolu byla diplomová práce zabývající se architekturou nového kolektoru [12].

4.1 Architektura kolektoru

Kolektor je rozdělen do několika modulů, které jsou rozděleny podle jejich funkce. Modularita umožňuje jednoduché rozšíření o další funkce kolektoru a také o dynamickou konfiguraci. Momentálně jsou moduly rozděleny do tří kategorií, a to moduly **vstupní** pro příjem IPFIX zpráv, moduly **vnitřní** pro další transformaci dat v přijatých zprávách, a moduly **výstupní** pro transformaci dat do jiných formátů. Moduly jsou v kolektoru uspořádány ve zmíněném pořadí a tím tvoří zřetězenou linku (angl. *pipeline*) pro zprávy, které si mezi sebou zasílají. Zřetězenou linku můžeme vidět na obrázku 4.1, kde vstupní moduly jsou vyznačeny modrou barvou, vnitřní moduly oranžovou a výstupní moduly jsou vyznačeny červeně. Sestavení modulů probíhá pomocí konfiguračních souborů ve formátu XML, kde jsou specifikovány moduly, které si uživatel přeje používat při běhu programu. Tímto se šetří výpočetní výkon zařízení, na kterém kolektor běží, díky tomu, že nepotřebné moduly nejsou vůbec spuštěny.

4.1.1 Vnitřní zprávy

Zprávy „proudí“ v kolektoru pouze jedním směrem a to od vstupních modulů k vnitřním modulům a končí u výstupních modulů. Typů zpráv je několik, ale hlavním typem je zpráva datová. Ta je vytvořena vstupním modulem, který v ní obsáhne nezpracovaná data. O kontrolu jejich formátu a vyplnění struktury datové zprávy se stará preprocesor IPFIX zpráv, který následuje po každém vstupním modulu. Zpráva tedy obsahuje především zaobalenou



Obrázek 4.1: Architektura IPFIXcol2 [12]

IPFIX zprávu, v případě konkrétních datových záznamů obsahuje i ukazatel na šablonu, podle které byla data poskládána do zprávy, a také identifikaci exportního procesu, který zprávu zaslal.

Dalším typem je zpráva relační. Relační zprávy jsou generovány při navázání nebo ukončení spojení sondy s kolektorem a nachází se v nich informace z třetí vrstvy referenčního modelu TCP/IP jako je například IP adresa a port sondy.

Dále existují i zprávy „odpadní“ a zprávy „terminační“. Ty jsou v kolektoru využity především ke správné funkčnosti kolektoru, a jelikož jejich recepce a zpracování provádí jádro kolektoru a vývojář jednotlivých modulů nemá k těmto zprávám přístup, nebudeme si tyto zprávy hlouběji probírat.

4.1.2 Moduly kolektoru

První kategorií modulů jsou moduly **vstupní**. Ty pracují paralelně a jsou implementovány především pro různé transportní protokoly, pomocí kterých jsou IPFIX zprávy přenášeny. Momentálně existují dva vstupní moduly, které názvem odpovídají transportním protokolům, pro které jsou určeny, tedy moduly UDP a TCP. Ihned po vstupních modulech následuje preprocesor IPFIX zpráv, který se stará o korektní formát přijatých dat. Špatně naformátované zprávy zahazuje a na špatně naformátované záznamy ve zprávách reaguje požadavkem na ukončení spojení v případě TCP nebo zahozením šablony špatné zprávy a neparsováním dat v případě UDP, aby další moduly nepracovaly s nekorektními daty. Správně naformátovaná data zabalí do datové zprávy, kterou propaguje vnitřním modulům. Další funkcí preprocesoru je zkoumání používaných šablon v komunikaci s kolektorem. Pro tento účel je všem modulům dostupný *Template Manager*, kde jsou uloženy čísla a definice šablon pro parsování dat uvnitř datových záznamů. Vstupní moduly ještě generují zprávy relační. Touto zprávou oznamují vnitřním modulům informaci o tom, že sonda navázala či ukončila spojení s kolektorem.

Druhým typem jsou moduly **vnitřní** (angl. *Intermediate*). Oproti vstupním modulům jsou zřetězeny sériově. Jejich hlavním účelem je transformace informací o datových tocích, dříve než se zprávy dostanou k výstupním modulům. Momentálně je součástí kolektoru jediný vnitřní modul pro anonymizaci IP adres datových toků. Do budoucna je však možnost rozšířit kolektor o další vnitřní moduly, jako je například modul pro filtrování. Tento

typ modudlů se využívá k implementaci *Intermediate* procesů, popsaných v architektuře protokolu IPFIX.

Posledním typem modulů jsou moduly **výstupní**. Ty opět pracují paralelně, nezávisle jeden na druhém. O propagaci zprávy k těmto modulům se stará manažer výstupních modulů, který každému z výstupních modulů zasílá odkaz na datovou zprávu a nastaví počítadlo těchto odkazů. Každý výstupní modul poté po ukončení práce dekrementuje počítadlo. Až hodnota počítadla dosáhne hodnoty 0, je zpráva uvolněna z paměti.

4.2 Provozní parametry

Obecný popis parametrů, které lze sledovat u protokolu IPFIX, je popsán v RFC 6615 [8], a to jak parametry pro kolektor, tak i pro sondu. V této sekci se ale zaměříme na parametry pro kolektor. Ve zmíněném RFC je hned v úvodu zmíněno, že ne všechny popsané parametry musí být sledovány, což se odvíjí od způsobu implementace kolektoru. Aktuální implementace kolektoru IPFIXcol2 neumožňuje sledování všech definovaných parametrů. Proč tomu tak je, si rozebereme u konkrétních parametrů.

Parametry jsou popsány pomocí jazyka ASN.1 a jejich struktura odpovídá dvěma podstromům MIB databáze. Přesněji řečeno dvěma modulům MIB databáze, přičemž druhý modul **IPFIX SELECTOR MIB** je využit především pro sondy a popisuje parametry a definice vzorkovacích funkcí měřícího procesu. Měřící proces se na kolektoru nenachází, proto se zaměříme na první modul, a to **IPFIX MIB**. Tento modul se skládá z tabulek parametrů, které jsou rozdělené do dvou podstromů. První podstrom slouží pro hlavní parametry kolektoru a v druhém podstromu nalezneme statistické údaje k hlavním parametrům v prvním podstromu. V obou podstromech se opět nachází tabulky, které obsahují parametry pro sondy. Těmito se nebudeme momentálně zabírat a zaměříme se na parametry kolektoru.

První kategorií parametrů jsou relační údaje mezi kolektorem a sondami. Zde se sledují informace jako cílová a zdrojová adresa, cílový a zdrojový port, transportní protokol, typ aktuálního zařízení (sonda či kolektor) a aktivita spojení. Pro transportní protokol UDP navíc informace o časovém limitu, kdy vyprchá platnost šablon, a pro transportní protokol SCTP navíc informace o tzv. *Association ID*, které je ovšem v aktuální implementaci kolektoru nedostupné. Vstupní modul pro SCTP ho nezahrnuje do relačních zpráv, které generuje. Pro relační údaje lze sledovat statistiky ohledně aktuální rychlosti přenosu zpráv, která je měřena v jednotkách B/s, a počítadla přijatých zpráv, přijatých bajtů, počtu záznamů v přijatých zprávách a počtu šablon používaných v komunikaci. RFC 6615 navíc definuje i počítadla zahozených zpráv a počtu paketů. Tyto tvě počítadla jsou v aktuální implementaci problémem, jelikož v kolektoru aktuálně neexistují zprávy, které by signalizovaly špatně naformátované zprávy. Preprocesor zpráv je zahodí bez další notifikace. A počet paketů je také problém z toho důvodu, že pokud je IPFIX zpráva fragmentována na několik paketů kvůli maximální velikosti segmentu transportní vrstvy, systém se postará o její poskládání dohromady dříve, než samotná zpráva doputuje ke kolektoru.

Dalšími sledovanými parametry jsou údaje o šablonách. Pro každou relaci lze sledovat, jaké šablony jsou v komunikaci používány a jaké informační elementy tyto šablony obsahují. Jak již bylo zmíněno v sekci 3.3, exportéry mohou mít několik měřících procesů, které jsou rozlišeny pomocí ODID. Proto i šablony se dělí nejen na základě relací mezi kolektorem a exportérem, ale i na základě ODID. Dalším parametrem u šablon je čas posledního výskytu definice šablony v komunikaci. V případě transportního protokolu UDP se tento čas dá

použít i k rozpoznání toho, kdy vyprší interval používání pro danou šablonu, a tím se stane šablona neplatná. U informačních elementů se dají uchovávat údaje o identifikátoru informačního elementu, jeho datovém typu, velikosti dat a číslu privátního rozsahu PEN. Statistické údaje, které lze sledovat u šablon, jsou především počítadla datových záznamů, které danou šablonu použily pro přenos informací z exportéru na kolektor.

Jednou z dalších informací, kterou obsahují statistické tabulky IPFIX-MIB modulu, je sloupec nazvaný *Discontinuity Time*. V tomto sloupci se uchovává časová informace, která signalizuje tzv. „přetečení“ jednoho z počítadel. Zajímavostí je, že například ve statistické tabulce pro relace je těchto počítadel několik, ale pro všechny je jeden společný časový údaj o přetečení. Aplikace, která má k těmto statistickým údajům přístup, musí tedy sama určit, které počítadlo překročilo maximální hodnotu, a to na základě předchozích nasbíraných hodnot.

Kapitola 5

Návrh a realizace statistického modulu

Nový statistický modul je řešením absence monitorování kolektoru. Momentálně neexistuje jednoduchý způsob, jak sledovat provozní parametry. Jedna z možností by mohla být analýza výstupních dat kolektoru, ze které bychom dokázali zjistit informace jako například používané šablony a počet jejich použití, ale další informace by bylo možné zjistit jen těžko nebo vůbec (např. přenosová rychlost v relaci mezi kolektorem a exportérem). Analýza výstupních dat se tedy jeví jako neefektivní způsob, a proto je žádoucí za tímto účelem vytvořit speciální modul.

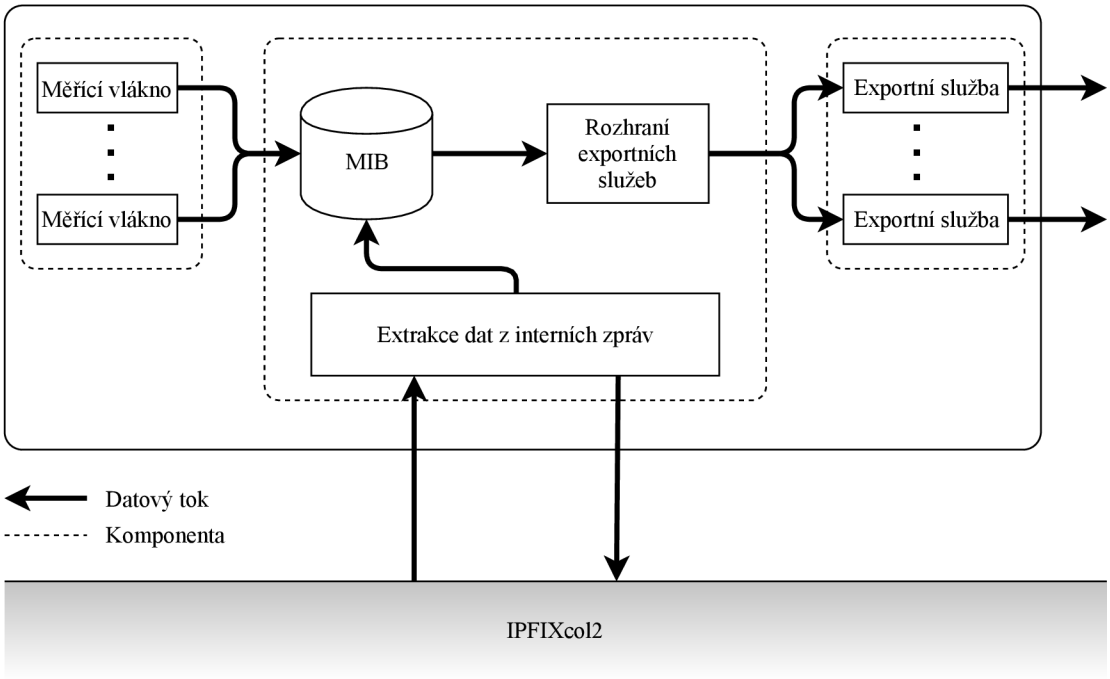
Statistický modul je tedy jedním z přídavných zásuvných modulů kolektoru. Jelikož existují tři typy těchto modulů, jak bylo popsáno v sekci 4.1.2, je třeba si v první řadě určit typ tohoto modulu. Ten se svou funkčností neřadí mezi vstupní moduly, takže bylo třeba učinit rozhodnutí mezi modulem vnitřním a výstupním. Požadavkem statistického modulu není transformace či úprava informací o datových tocích, ale pouze jejich sledování, ukládání a export. Proto by se nabízelo implementovat modul jako modul výstupní. V tomto bodě plánování však bylo třeba zvážit, jaké informace se k modulu dostanou. Pokud by byl v kolektoru použit vnitřní modul, který by upravoval data potřebná pro statistický modul, neměli bychom přístup ke všem informacím a získané statistiky by mohly být nepřesné. Proto je statistický modul vnitřním modulem kolektoru. Už jen tímto rozhodnutím se naskýtá první požadavek na modul a tím je jeho rychlost. Vnitřní moduly jsou zapojeny sériově, a proto by pomalý vnitřní modul mohl zpomalit zpracování zpráv v kolektoru. K řešení tohoto problému se dostaneme v sekci 5.1 zabývající se architekturou modulu a v sekci 5.2, kde je popsáno konkrétní řešení.

5.1 Architektura modulu

Stejně jako u celého kolektoru *IPFIXcol2* byla jedním z cílů snadná rozšiřitelnost implementace, tak i u návrhu samotného modulu jsem se řídil tímto cílem. Výsledný zásuvný modul je tedy poskládán z dílčích komponent, přičemž každá komponenta představuje jednu specifickou úlohu v zásuvném modulu. Zároveň činnost komponenty (nebo činnost jejích částí) probíhá odděleně ve vlákně, takže dohromady pracují paralelně. Komunikace pak probíhá na základě sdílených proměnných.

Hlavní rozšiřitelnost se nachází v části pro export provozních parametrů, kde se do budoucna předpokládá implementace nových exportních služeb, jelikož momentálně modul

dokáže exportovat informace pouze pomocí protokolu SNMP. Jedním z možných rozšíření může být například export parametrů ve formátu JSON.



Obrázek 5.1: Schéma uspořádání jednotlivých komponent zásuvného statistického modulu

5.1.1 Popis komponent

Extrakce dat a komunikace s kolektorem

Hlavní částí zásuvného modulu je komponenta pro komunikaci s kolektorem a zpracování dat. Jeho primárním cílem je komunikovat s ostatními moduly kolektoru a dále extrahovat a ukládat potřebné informace z interních zpráv a tyto zprávy dále posílat dalším modulům. Právě tato část je z pohledu rychlosti klíčová, právě kvůli komunikaci s ostatními moduly, a proto je její činnost separována do hlavního vlákna modulu. Extrakce a ukládání dat probíhá pouze v míře, která je potřebná pro získání potřebných informací. Při extrakci dat se tedy neprochází všechny datové struktury ve zprávách, ale činí se pouze průchody nezbytně nutné.

V původním návrhu bylo plánem separovat i zpracování zpráv do asynchronního vlákna, ovšem naskytl se problém s interními zprávami, které procházejí kolektorem. Data uvnitř interních zpráv jsou platná pouze při práci hlavního vlákna zásuvného modulu, které je odpovědné za příjem zprávy od předchozího modulu a za její přeposlání modulu následujícímu. Pokud bychom tedy chtěli zpracovávat data v jiném než hlavním vláknu, muselo by být zaručeno, že se tak stane po přijmu zprávy a před jejím odesláním. Tím by vzrostla režie komunikace mezi vlákny zásuvného modulu, přičemž doba zpracování by byla stejná či ještě větší. Proto je extrakce a ukládání součástí zásuvného modulu, kde je jistota, že zpracovávaná data ze zprávy jsou platná.

Měřící vlákna

Měřící vlákna provádí statistická měření závislá na čase. Jejich oddělení bylo nezbytné právě kvůli přesnosti měření a rychlosti modulu. Pokud by měření probíhalo pouze na základě časových razítek, docházelo by k nepřesnostem kvůli tomu, že výsledný modul je spouštěn pouze v případě příchozí zprávy. Pro příklad si představme situaci, kdy exportér zašle jednou za 10 sekund zprávu o velikosti 500 bajtů. Zásuvný modul by tedy také pracoval pouze 1krát za 10 sekund a při použití časových razítek bychom jako výslednou hodnotu rychlosti přenosu za posledních 10 sekund dostali 50 B/s. To ovšem nevystihuje realitu, kdy exportér byl po většinu času intervalu neaktivní. Z tohoto důvodu jsou měřící vlákna oddělena a provádí úpravu dat pokaždé po uplynutí intervalu, ve kterém provádí měření. Aktuálně je třeba měřit v čase dvě položky IPFIX-MIB databáze, a to rychlost přenosu v relaci a aktivita relace. O to se stará jedno měřící vlákno, ale do budoucna se nevylučuje možnost rozšíření o další měrné statistiky, čímž by se zvýšil i počet měřících vláken.

Exportní služby

Poslední komponentou jsou exportní služby a jejich správa. Pomocí služeb se přistupuje k uloženým provozním parametrům. Tyto služby by však opět mohly svou činností zpomalovat celý zásuvný modul, a proto jsou opět oddělené do samostatně běžících vláken. O jejich správu se stará speciální rozhraní. I přesto, že aktuální návrh popisuje pouze jednu výstupní službu, a to službu protokolu SNMP, je rozhraní navrženo pro další rozšíření služeb, jako například zmíněná služba pro export parametrů ve formátu JSON. V tomto rozhraní mohou pracovat další notifikační vlákna, která by všechny exportní služby informovala o různých událostech, které v kolektory nastaly nad daty (nová příchozí zpráva, překročení nastavených prahů pro hodnoty atd.). Služba protokolu SNMP však primárně nevyžaduje tyto notifikace, a proto jsou notifikační vlákna pouze návrhem pro další rozšíření v případě nových exportních služeb.

5.1.2 Exportní služba SNMP

Předtím, než byla navržena architektura exportní služby protokolu SNMP, bylo zapotřebí vybrat způsob implementace této služby. Prvním řešením by bylo implementovat celý proces obsluhy bez použití externích knihoven. Tím by se sice modul oprostil od závislostí, ale složitost implementace samotné obsluhy by byla pro účel služby až příliš velká. Tím pádem jsem se rozhodl použít jednu z dostupných knihoven, kde jsou mechanismy obsluhy již implementovány a otestovány vývojáři knihovny. Hlavním kritériem výběru byla jednoduchá instalace, aby byl zásuvný modul lehce použitelný.

Zvolena byla knihovna NetSNMP[1], která jako jediná splňuje hlavní kritérium. Dostupné jsou i další Open Source knihovny, jako například SNMP++[2], kde je však zapotřebí manuální překlad kódu a instalace, což by pro některé uživatele mohlo být překážkou k používání statistického modulu. Oproti tomu je knihovna NetSNMP lehce dostupná pro širokou škálu linuxových distribucí skrze jejich oficiální repozitáře balíčků. K tomu nabízí i podporu AgentX protokolu, jehož princip si vysvětlíme v sekci 5.2.

Dalším krokem návrhu bylo zjišťování možností knihovny Net-SNMP. Především bylo nutné navrhnout způsob práce nad interními daty. Bez použití synchronizačních prostředků by hlavní vlákno modulu mohlo exportní službě upravovat data při obsluze požadavků.

[1] `http://www.net-snmp.org/`

[2] `https://agentpp.com/api/cpp/snmp_pp.html`

A v případě implementace zámku sdílených datových struktur by naopak exportní služba mohla svou častou činností zpomalovat hlavní vlákno, a tím i rychlost zpracování zpráv v celém kolektoru. Tento problém řeší systém mezipaměťí (angl. „cache“) pro každou implementovanou tabulku IPFIX-MIB databáze. Cache každé tabulky funguje na základě expirace, tedy každé cache paměti je nastaven časový limit, po který jsou data v cache validní. Po uplynutí tohoto limitu však nedochází k přímému obnovení dat, ale cache je pouze označena jako expirovaná. Obnova dat přichází až při příchozím SNMP požadavku, kdy je nejprve zkontrolován stav cache paměti, poté jsou data v ní obnovena a následně je obsloužen požadavek. Tímto se zamezí zbytečným přístupům k interním datům či jejich inkonzistenci. Kompromisem je neúplná aktuálnost dat v mezipamětích, která závisí na délce expirace. Pokud však uvažujeme typ dat, ke kterým služba přistupuje, je tento nedostatek zanedbatelný. Cache mechanismus je podporován přímo knihovnou Net-SNMP.

Dále je třeba vysvětlit způsob obsluhy příchozích požadavků. Pokud by si exportní služba vytvořila síťový *socket* na standartních portech 161 nebo 162, musela by začít obsluhovat všechny příchozí SNMP požadavky, a to i pro ty části MIB databáze, kterou neimplementuje. Řešením tohoto problému je protokol AgentX, ve kterém se definuje role hlavního agenta (*master agent*) a role podagenta (*subagent*). Hlavní agent si vytvoří síťové sockety pro obsluhu SNMP požadavků a podagenti se poté připojují k hlavnímu agentovi, přičemž mu definují, které části MIB databáze implementují. Při příchozím SNMP požadavku si tedy hlavní agent nejdříve vyhledá podagenta, kterému požadavek patří, a tomu ho dále zasílá. Hlavní agent a podagenti spolu komunikují pomocí souborového socketu [7].

V případě této architektury je také potřeba hlavního agenta na zařízení, na kterém běží kolektor. Přesněji systémovou službu `snmpd`, která je spuštěna jako hlavní agent. K němu se exportní služba (statistický modul) připojuje a tím obsluhuje pouze implementovanou část IPFIX-MIB databáze.

5.2 Realizace modulu

Modul je implementován v jazyce C++, a to jak kvůli kompatibilitě s kolektorem, tak i kvůli rychlosti zpracování dat ve výsledném programu. Využitá knihovna Net-SNMP je zároveň s kolektorem nezbytná pro překlad výsledného kódu. Spolu s knihovnou jsou distribuovány i linuxové nástroje vhodné pro testování a vývoj agenta. Při implementaci byl využit třídní přístup.

Rozhraní zásuvných modulů v kolektoru má předem definovanou strukturu a také funkce, které jádro kolektoru vyžaduje pro správnou komunikaci s modulem. Implementace zásuvného modulu tedy byla závislá na této struktuře tohoto rozhraní. Nezbytnými prvky pro implementaci modulu jsou tři funkce, a to inicializace modulu, funkce zpracovávající příchozí interní zprávy a funkce pro ukončení modulu.

5.2.1 Inicializace

V inicializační funkci se především parsuje konfigurace modulu z konfiguračního souboru. V něm se dá definovat, které exportní služby mají být spuštěny se spuštěním modulu. Jak již bylo několikrát zmíněno, aktuálně existuje pouze služba protokolu SNMP, a jelikož modul nelze spustit bez exportní služby, je zároveň i nezbytné v konfiguraci definovat použití této služby. Základní konfiguraci modulu můžeme vidět na obrázku 5.2. Dále může konfigurace modulu obsahovat *timeout* hodnoty, a to jak pro celý modul, tak i pro exportní služby. Pro celý modul lze specifikovat timeout pro rozeznání aktivního či neaktivního spojení. Bez

```
<intermediate>
  <name>Collector stats</name>
  <plugin>statistics</plugin>
  <params>
    <outputs>
      <snmp></snmp>
    </outputs>
  </params>
</intermediate>
```

Obrázek 5.2: Základní konfigurace statistického modulu ve formátu XML bez specifikace volitelných položek

specifikace je spojení označeno za neaktivní, pokud za posledních 10 sekund neproběhl ve spojení přenos dat. Právě položkou v konfiguraci lze tento časový interval upravit. Ukázku konfigurace modulu, která obsahuje všechny volitelné položky, můžeme vidět v příloze B.

Konfigurace služby SNMP dovoluje specifikovat timeout pro cache jednotlivých tabulek. Bez nastavení pomocí konfiguračního souboru je hodnota timeout nastavena na 1 sekundu, a to z toho důvodu, že se jedná o nejmenší použitelnou hodnotu. Existuje i možnost nastavení hodnoty na 0 sekund, což znamená, že cache by byla stále v expirovaném stavu a obnova by probíhala při každém požadavku. To by však mohlo zapříčinit problémy, pokud by manažerská aplikace SNMP začala posílat velké množství požadavků najednou. Exportní modul by stále přistupoval k interní struktuře s daty a tím by zpomalil chod kolektoru. Nemusí se jednat pouze o manažerskou aplikaci, ale i případný útočník by mohl přehltit kolektor požadavky a tím by mohlo vzniknout zpoždění při sběru dat. Při použití intervalu 1 sekundy se tedy eliminují zmíněná nebezpečí a správce bude mít stále přístup k relativně aktuálním datům (viz sekce 5.1).

Po úspěšném načtení konfigurace následuje inicializace jednotlivých komponent modulu. Úložiště si nejdříve vytváří potřebné interní struktury a následovně spouští měřící vlákna, o jejichž běh se stará. Rozhraní exportních služeb při inicializaci vytvoří třídy exportních služeb, kterým následně signalizuje požadavek na spuštění. Rozdělení inicializace a spuštění bylo implementováno proto, aby i každá ze služeb měla možnost si nejdříve nastavit interní parametry před tím, než se spustí.

Inicializace služby SNMP však probíhá až při jejím spuštění. Je to z důvodu omezení knihovny Net-SNMP pro její použití pouze v jediném vlákně aplikace. V tomto případě se vlákno spouští až při signalizaci spuštění. Takže nejdříve proběhne inicializace interních struktur knihovny včetně cache pamětí a následně vlákno přechází do stavu, kdy naslouchá příchozím požadavkům. V rámci interní inicializace knihovny probíhá i připojení k hlavnímu agentovi `snmpd`. Pokud by agent na zařízení nebyl spuštěn, nespustí se ani exportní služba a celý kolektor končí s chybovým kódem a příslušnou hláškou. Toto obecně platí pro všechny exportní služby statistického modulu. Pokud by některá nebyla úspěšně spuštěna, nespustí se ani kolektor.

5.2.2 Zpracování dat

Při běžném provozu modul dostává interní zprávy relační a datové, přičemž se volá druhá nezbytná funkce modulu, a to právě pro zpracování vnitřních zpráv. Na základě relačních

zpráv se vytváří a odstraňují záznamy v mapovací tabulce modulu, kde se mapují textové identifikátory na číselné identifikátory pro rychlejší vyhledávání. Textový identifikátor obsahuje IP adresu a port příjemce i odesílatele a název transportního protokolu použitého v relaci. Číselný identifikátor je přidělen při zprávě oznamující novou relaci a tento identifikátor je pro dané spojení stejný po celou dobu běhu kolektoru. Pokud by se exportér odpojil, je záznam z mapovací tabulky přenesen do tabulky ukončených spojení. Po případném opětovném připojení je nejdříve provedeno vyhledání číselného identifikátoru v této tabulce. Tím dostane exportér po opětovném připojení stejný identifikátor jaký měl během své předchozí relace. Pouze při prvotním připojení je vyhledání v tabulce ukončených spojení neúspěšné a pro spojení je vytvořen nový číselný identifikátor.

Co se týče datových zpráv, nejdříve je opět sestaven textový identifikátor z relačních údajů které zpráva obsahuje, a poté se provede vyhledání číselného identifikátoru v mapovací tabulce aktivních spojení. Při úspěšném vyhledání se poté extrahují potřebná data ze zprávy a ukládají se do interního úložiště. Aby byla uložená data konzistentní, je nad interním úložištěm implementovaný synchronizační prostředek *spinlock*, který zamezí čtení dat exportními službami při jejich úpravě (tedy synchronizačnímu problému známému jako *data race*). Ten byl upřednostněn před prostředkem zvaným *mutex* hlavně kvůli menšímu nároku na výpočetní výkon z důsledku režie běhu vlákna. Úpravu dat provádí hlavní vlákno modulu a měřící vlákna. Čtení dat provádí exportní služby.

Aktuálně jediné měřící vlákno provádí měření na základě oddělené činnosti nad daty s použitím časových razítek. Proces měření tedy probíhá tak, že vlákno je uspáno po určitý časový interval, během kterého hlavní vlákno sbírá data. Interval je aktuálně zvolen na délku 1 sekundy, a to z důvodu měření rychlosti, která je měřena v jednotkách B/s. První činností měření je tedy měření rychlosti přenosu v relaci, které probíhá tak, že hlavní vlákno během doby sbírání dat inkrementuje interní počítadlo přenesených bajtů pro konkrétní relaci při nové příchozí datové zprávě. Po uplynutí intervalu měřící vlákno kopíruje hodnoty počítadel všech relací do náležitých záznamů interní databáze a počítadla vynuluje.

Druhou měřící činností je kontrola aktivity relací. Tato činnost využívá zmíněných časových razítek. Při nové datové zprávě hlavní vlákno poznačuje u relace čas, kdy byla naposledy přijata data od exportéru, a měřící vlákno po probuzení kontroluje časový rozdíl, zdali není větší než nastavená hodnota. Pokud ano, označuje relaci za neaktivní.

5.2.3 Obsluha SNMP požadavků

O parsování přijatých požadavků a sestavení odpovědí se v exportní službě SNMP stará především knihovna Net-SNMP. Ta k tomu ale využívá uživatelem definované funkce pro inicializaci a destrukci tabulek, naplnění a vyprázdnění cache paměti a obsluhu příchozího požadavku pro každou tabulku. Zde je zajímavé podotknout, že knihovna Net-SNMP obsahuje i skript pro vygenerování těchto funkcí v programovacím jazyce C na základě popisu MIB modulu v jazyce ASN.1. Výsledné vygenerované funkce však mají k použitelnosti ještě daleko, jelikož jejich kompilace je po vygenerování nemožná a neobsahují kontext aplikace, které mají být součástí, takže celková úprava je nezbytná.

Podstatnou částí obsluhy požadavků je právě funkce pro samotnou obsluhu. Zde je třeba převést data z formátu, ve kterém byla uložena v interním úložišti, do formátu, který lze přenést ve výsledném paketu. Pro základní datové typy, jako například *integer*, se o konverzi postarají knihovní funkce, ale u složitějších a specifických formátů jako například formát časové známky indikující přetečení počítadla je třeba výsledná data konvertovat manuálně.

Kapitola 6

Využití a měření výsledků

V této kapitole budeme měřit a testovat modul a jeho exportní službu protokolu SNMP. Našim cílem bude dokázat jeho nízký dopad na propustnost kolektoru a to v různých případech využití modulu. Statistický modul je kvůli závislosti na externí knihovně dostupný jako přídavný modul do kolektoru *IPFIXcol2*. Jeho překlad tedy neprobíhá zároveň s celým kolektorem, ale je třeba ho provést samostatně, zároveň s instalací. Modul je poté integrován do kolektoru pomocí konfiguračního souboru. Právě na základě různých konfigurací kolektoru budeme provádět testování modulu.

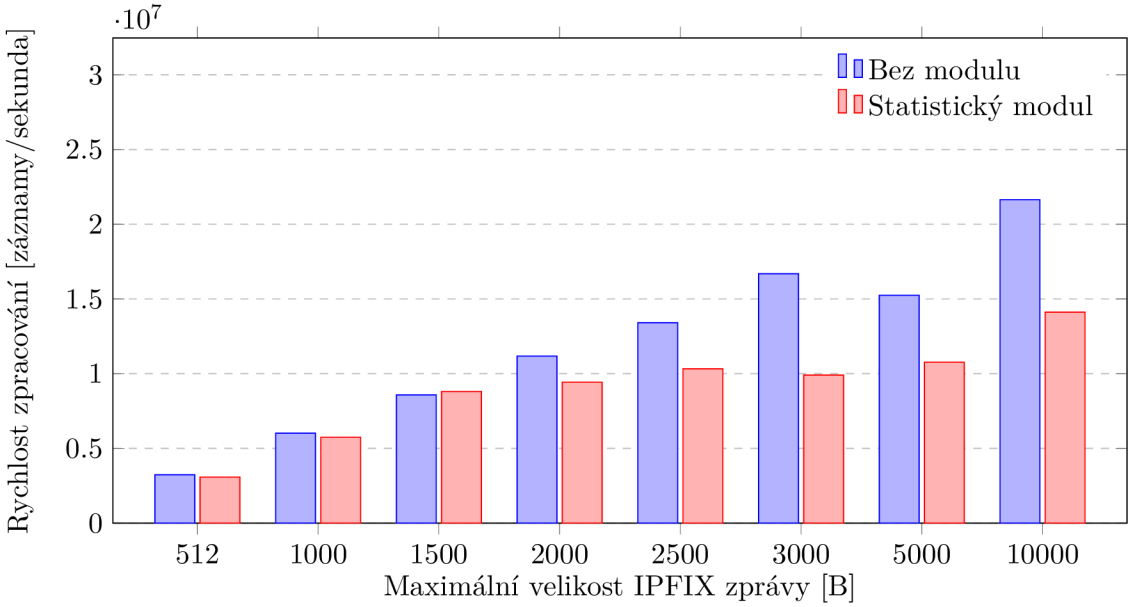
6.1 Testování a měření výsledků

Prvním testovaným faktorem byla propustnost kolektoru při využití statistického modulu. Pro porovnání jsem si vybral propustnost kolektoru, který žádný vnitřní modul neobsahuje. Cílem testování tedy bylo určit, zdali se statistický modul řadí, na základě naměřených výsledků, mezi pomalejší zásuvné moduly, či jeho použití výrazně neovlivní výkon kolektoru.

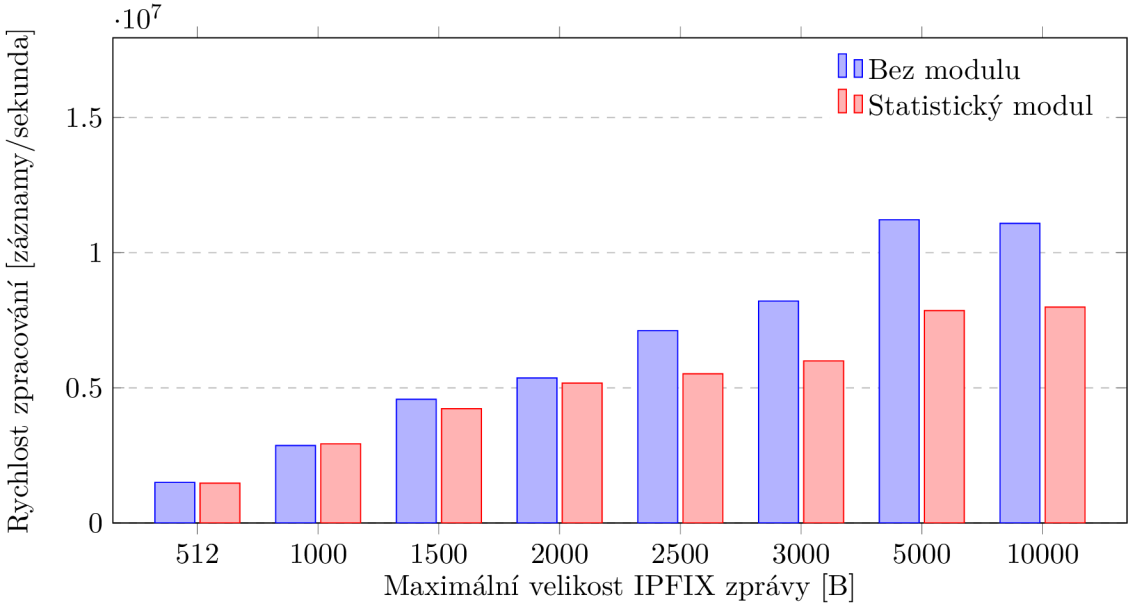
Při testování jsem používal stejný postup, který byl použit pro testování jednotlivých modulů kolektoru při jeho vývoji. Na spuštěný kolektor jsem si pomocí nástroje *IPFIXsend2* přehrával dříve nasbírané IPFIX zprávy, čímž jsem simuloval exportní proces. Použité testovací sady jsou totožné těm sadám, které byly využity při testování celého kolektoru během jeho vývoje. Sady se liší se v typu dat obsažených ve zprávách, kde prvním typem jsou sady obsahující pouze základní informace o datových tocích a ve druhém případě sady obsahují navíc informace z aplikační vrstvy. Dále byl zvolen různý limit maximální velikosti IPFIX paketu pro každou sadu stejného typu. Jediným rozdílem bylo testovací zařízení, které svým výkonem neodpovídalo použitému zařízení při testování kolektoru. Naměřené hodnoty jsou proto rozdílné než v případě výsledku testování kolektoru, které bylo provedeno v rámci diplomové práce [12]. Parametry zařízení, ne kterém byl modul testován, jsou uvedeny v tabulce 6.1.

CPU	Intel Core i5-6200U (2.30GHz, TDP 15W)
RAM	1x 8192MB (2133MHz)
Operační systém	Ubuntu 18.04
Nástroje pro překlad	gcc 7.3.0, cmake 3.10.2
Parametry překladu	bez ladících symbolů, optimalizace O2

Tabulka 6.1: Parametry testovacího zařízení



Obrázek 6.1: Porovnání rychlostí zpracování běžných dat ve zprávách při využití vnitřního statistického modulu, a rychlostí bez využití jakéhokoli vnitřního modulu.

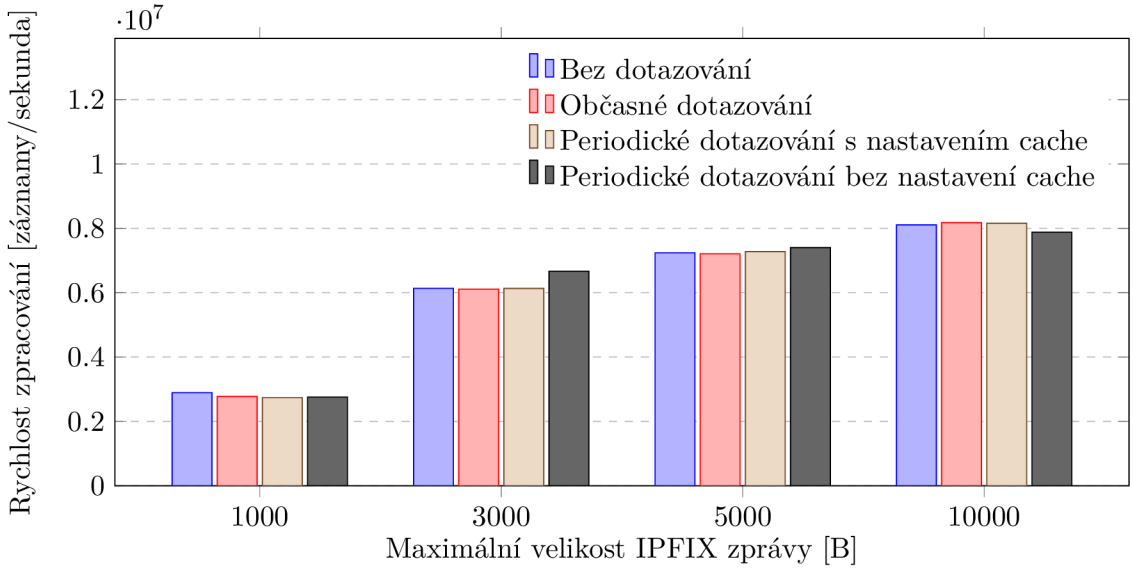


Obrázek 6.2: Porovnání rychlostí zpracování aplikačních dat ve zprávách při využití vnitřního statistického modulu, a rychlostí bez využití jakéhokoli vnitřního modulu.

Z naměřených výsledků v grafech 6.2 a 6.1 je patrné, že statistický modul ovlivní propustnost kolektoru v přijatelné míře a to především při zpracování zpráv, které mají maximální velikost do 2500 bajtů. Při větších velikostech vzniká větší režie při extrakci dat, což zapříčiní útlum růstu rychlosti zpracování. Podobný vývoj lze sledovat jak při zpracování běžně sledovaných parametrů o datových tocích, tak při zpracování aplikačních dat.

V druhé fázi testování jsem uvažoval tři příklady použití modulu. V prvním případě je to nasazení statistického modulu pro občasnou kontrolu. V takové situaci předpokládáme, že administrátor jednou za čas zašle požadavek na aktuální data, vyhodnotí je a tím kontrola končí. Nejedná se tedy o periodické dotazování, ale o dotazování náhodné. Oproti tomu v druhém případě uvažuji periodické dotazování na aktuální data, a to při specificky nastaveném statistickém modulu. Při tomto stavu probíhá časté dotazování každou 1 sekundu na všechny hodnoty, ale cache paměti jednotlivých tabulek mají nastavenou dobu expirace dle typu dat, která se v nich nachází. Například tabulka relací mezi kolektorem a exportérem obsahuje data, která se spíše nemění, proto je doba expirace nastavena na 15 sekund. Oproti tomu tabulka statistik těchto relací obsahuje například aktuální rychlost přenosu v relaci, tedy doba expirace je nastavena na 1 sekundu. Posledním a nejvíce náročným případem je stav, kdy probíhá dotaz na všechny hodnoty v tom nejkratším možném intervalu, což je 1 sekunda. Tím se snažíme reflektovat stav, kdy správce chce mít v manažerské aplikaci co nejaktuálnější data při použití výchozích hodnot expirací cache pamětí. V tomto případě probíhá daleko častější přístup do interní databáze, než ve předešlých dvou případech.

U všech tří případů by na konci testování mělo platit, že dotazování na hodnoty výrazně nesnižuje propustnost kolektoru. Tímto zjistíme závislost hlavního vlákna na činnosti exportní služby protokolu SNMP. Kvůli absenci modulu podobného charakteru v aktuální implementaci kolektoru *IPFIXcol2* nebude možné porovnávat výsledky s jiným modulem. Porovnání tedy bude prováděno s propustností kolektoru se statistickým modulem bez jakéhokoli dotazování se na hodnoty pomocí SNMP. Jak můžeme vidět z výsledků druhé fáze



Obrázek 6.3: Porovnání rychlosti zpracování aplikačních dat ve statistickém modulu při různém způsobu dotazování se na data pomocí protokolu SNMP

měření, zobrazených v obrázku 6.3, tak ani jeden ze tří uvažovaných případů využití nijak

výrazně neovlivňuje propustnost kolektoru. Rozdíly, které lze vidět oproti stavu bez dotazování, jsou pouze v rámci chyby měření. Lze tedy usoudit, že častější zamykání sdílené interní databáze, z důvodu obnovy dat v cache paměti, má zanedbatelný efekt na činnost hlavního vlákna statistického modulu.

6.2 Využití v praxi

Výhodou statistického modulu je, že aktuálně používá standardní protokol SNMP, který využívají již existující manažerské aplikace. Není tedy třeba vyvíjet vlastní aplikaci pro sledování stavu kolektoru, ale můžeme si vybrat již existující nástroj, který podporuje protokol SNMP do nějž lze přidávat vlastní sledované elementy (pokud tedy již neobsahuje definici IPFIX-MIB modulu). Prvním takovým nástrojem je MibBrowser[1] od společnosti Manage Engine, který je volně dostupný. Pomocí něj se můžeme připojit ke sledovanému zařízení, lokálně si importovat popis určitého modulu a na zařízení zasílat SNMP požadavky s identifikátory elementů. Tento nástroj je vhodný převážně pro občasné vyčítání hodnot, tedy v situaci, kdy administrátor nechce shromažďovat monitorovací data, ale pouze jednou za čas provede kontrolu stavu. Nástroj dále nabízí i základní vytváření grafu z hodnot, které dokáže sbírat po určitý čas, či přehledné tabulkové zobrazení pro lepší orientaci.

Pokud se ale chceme dotazovat na stav kolektoru periodicky, uchovávat si monitorovací data a vytvářet si pravidla pro nasbírané hodnoty, existují i sofistikovanější nástroje, jako například volně dostupný nástroj Zabbix[2] od společnosti Zabbix LLC. Ten nabízí kompletní uživatelské rozhraní nejen pro monitorování pomocí SNMP, ale nabízí i podporu dalších protokolů. V tomto nástroji lze definovat vlastní elementy, kterým se přiřadí identifikátor prvku v MIB databázi a specifikují se údaje potřebné k připojení na monitorované zařízení. Nástroj Zabbix se poté v definovaných časových intervalech dotazuje na hodnoty pomocí poskytnutých údajů, ty shromažďuje a nabízí například nastavení prahů hodnot pro určité elementy, na základě kterých dokáže uživatele upozornit na nastalou událost. Příkladem použití je nastavení prahu pro hodnoty rychlosti přenosu. Klesne-li hodnota rychlosti pod nastavený práh, může to signalizovat událost, kdy například exportér přestává zasílat zprávy, a o této události nástroj Zabbix informuje správce tzv. alarmem.

Jedním z příkladů nasazení modulu v praxi je jeho použití na testovacím kolektoru organizace CESNET. Tento kolektor sbírá metadata o aktuálním provozu v určitých částech monitorované sítě. Nejedná se tedy o pouhé přehrávání a sběr již dříve nasbíraných zpráv, ale jedná se o zprávy obsahující metadata o „živém“ provozu. V tomto prostředí se otestuje nejen správná funkce implementovaného modulu, ale dále se může použít právě ke sledování aktivity testovacího kolektoru.

Po důkladném otestování modulu na tomto kolektoru se statistický modul může stát součástí volitelných přídavných modulů, které jsou distribuovány s nejaktuálnější verzí *IPFIXcol2*.

6.3 Budoucí rozvoj

I přesto že většina částí IPFIX-MIB modulu je aktuálně implementována, stále existují položky, jejichž hodnota je v aktuální implementaci kolektoru neznámá. Těmito položkami jsou:

[1] `https://www.manageengine.com/products/mibbrowser-free-tool/`
[2] `https://www.zabbix.com/`

- **SCTP Association ID**, kvůli absenci vstupního SCTP modulu.
- **Počet zpráv, po kterém je očekávaná zpráva s opětnou definicí šablony** v případě použití UDP protokolu. Momentálně se pro tento účel využívá pouze časových údajů.
- **Počet zahozených zpráv**, jelikož preprocesory zpráv špatně naformátované zprávy zahazují bez zaslání upozornění ostatním modulům.
- **Počet obdržených paketů**, a to proto, že pokud je IPFIX zpráva fragmentována na více paketů, tak ty jsou sjednoceny dříve, než se v systému dostanou ke kolektoru.

Jedná se tedy o položky, pro jejichž získání bychom museli zasáhnout do hlavní struktury celého kolektoru.

Dalším plánovaným rozšířením je zmiňované přidání služeb pro export dat ze statistického modulu. V úvahu připadá formát JSON, který lze například využít při použití webové aplikace pro monitorování, ale do budoucna je možnost modul dále rozšiřovat i o služby jiných protokolů. Na základě rozšiřování o další exportní služby je plánováno i rozšíření rozhraní pro správu exportních služeb, které by se rozšířilo o upozornění závislá na čase či na splnění určité podmínky.

Při úvodním návrhu byla zvažována i implementace mechanismu pro zasílání SNMP zpráv typu TRAP. V RFC6615 [8], které o monitorování kolektoru protokolu IPFIX pojednává, však není specifikováno, při jakých událostech by TRAP zpráva musela být odeslána. Navíc schopnosti moderních monitorovacích aplikací umožňují nastavování prahů přímo v aplikaci bez nutnosti příjmu TRAP zprávy, proto i tato možnost zůstává návrhem pro budoucí rozšířeni.

Kapitola 7

Závěr

Tématem této práce bylo monitorování stavu IPFIX kolektoru, což zahrnovalo nastudování problematiky obecného monitorování a konkrétněji i monitorování síťových toků. Dále bylo zapotřebí zjistit, jaké provozní parametry se při monitorování vyskytují a je-li vhodné sledovat jejich vývoj. Pro účel implementace modulu bylo zapotřebí stručně popsat aktuální stav kolektoru *IPFIXcol2* a analyzovat možnosti sledování provozních parametrů. Významná část práce se zabývá návrhem a implementací nového statistického modulu, který řeší absenci modulu takové funkcionality. Právě kvůli absenci podobného modulu je nakonec výsledný modul testován v porovnání s jiným vnitřním modulem kolektoru.

Jelikož práce kombinuje pojmy *monitorování síťových toků* a *monitorování síťových zařízení*, byl za účelem porozumění popsán význam obecného monitorování včetně používaných praktik a protokolů, kde se popisuje i souvislost mezi těmito dvěma pojmy. Zmíněn je i význam monitorování a možná rizika, která mohou nastat při jeho zanedbání. Konkrétněji je popsán protokol SNMP, jakožto nejpoužívanější protokol pro správu sítě, včetně architektury systému, kde se tento protokol využívá.

Dále se práce zabývá stručným vysvětlením základních pojmů monitorování síťových toků. V textu lze nalézt popis architektury systému pro monitorování síťových toků a také vývoj jednotlivých protokolů používaných v tomto systému. Hlouběji je probrán protokol IPFIX, včetně vysvětlení principu sběru metadat o datových tocích. V rámci formátu IPFIX zpráv je stručně vysvětlen i systém šablon. Následuje popis aktuálního stavu kolektoru *IPFIXcol2*, který vyvíjí organizace CESNET. Hlavním tématem popisu je jeho architektura včetně popisu principu modularity a interní komunikace.

Na základě analýzy aktuálního stavu kolektoru byl navržen zcela nový statistický modul tohoto kolektoru, určený pro monitorování jeho aktuálního stavu. Vysvětlena je architektura statistického modulu, jejíž popis obsahuje i důvody, které vedly k rozhodování o výsledném návrhu. Text obsahuje i popis jednotlivých komponent modulu, přičemž podrobněji je vysvětlen princip služby modulu, která se stará o export dat pomocí protokolu SNMP. V rámci implementace je zmíněno řešení jednotlivých komponent a také řešení problémů, se kterými návrh modulu nepočítal, ale vyskytly se během implementace.

V závěrečných kapitolách je věnován prostor testování a využití modulu. V rámci testování byla porovnána propustnost kolektoru spuštěným se statistickým modulem s propustností kolektoru bez vnitřních modulů. Na výsledcích měření lze pozorovat malý, až zanedbatelný vliv na propustnost kolektoru. Dále bylo provedeno testování několika případů použití modulu s aktivním dotazováním na data. To dokázalo, že dotazování v různých intervalech s různým nastavením mezipamětí nemá na propustnost kolektoru významný vliv. Na závěr je v práci uveden popis využití včetně doporučených technik a nástrojů používaných ke sledování stavu kolektoru.

Jelikož je modul součástí stále se rozvíjejícího projektu kolektoru *IPFIXcol2*, vývoj statistického modulu tímto nekončí. Hlavním plánovaným rozšířením je obohacení statistického modulu o nové způsoby exportu nasbíraných dat pro širší škálu využití v praxi.

Literatura

[1] Bejtlich, R.: The Self-Defeating Network. https://taosecurity.blogspot.com/2007/01/self-defeating-network.html, online; navštíveno: 2019-04-02.

[2] Bruey, D.: SNMP: Simple? Network Management Protocol. Prosinec 2005, [Online; navštíveno 8. 04. 2019].
URL https://www.rane.com/note161.html

[3] Case, J. D.; Fedor, M.; Schoffstall, M. L.; aj.: Simple network management protocol (SNMP). Technická zpráva, IETF, 1990.

[4] Claise, B.: Specification of the IP flow information export (IPFIX) protocol for the exchange of IP traffic flow information. Technická zpráva, IETF, 2008.
URL https://www.rfc-editor.org/rfc/rfc5101.txt

[5] Claise, B.; Trammell, B.: Information Model for IP Flow Information Export (IPFIX). Technická zpráva, IETF, 2013.

[6] Claise, B.; Trammell, B.; Aitken, P.: Specification of the IP flow information export (IPFIX) protocol for the exchange of flow information. Technická zpráva, IETF, 2013.
URL https://www.rfc-editor.org/rfc/rfc7011.txt

[7] Daniele, M.; Wijnen, B.; Ellison, M.; aj.: Agent extensibility (AgentX) protocol version 1. Technická zpráva, IETF, 1999.

[8] Dietz, T.; Kobayashi, A.; Claise, B.; aj.: *Definitions of Managed Objects for IP Flow Information Export*. IETF, Červen 2012, [Online; navštíveno 25.03.2019].
URL https://www.ietf.org/rfc/rfc6615.txt

[9] Galvin, J.; McCloghrie, K.: Security protocols for version 2 of the simple network management protocol (SNMPv2). Technická zpráva, IETF, 1993.

[10] Held, G.: *LAN management with SNMP and RMON*. New York: John Wiley, vyd. 1. vydání, 1996, ISBN 0-471-14736-2.

[11] Hofstede, R.; Čeleda, P.; Trammell, B.; aj.: Flow Monitoring Explained: From Packet Capture to Data Analysis With NetFlow and IPFIX. *IEEE Communications Surveys Tutorials*, ročník 16, č. 4, Fourthquarter 2014: s. 2037–2064, ISSN 1553-877X, doi:10.1109/COMST.2014.2321898.

[12] Huták, L.: *Nová generace IPFIX kolektoru*. Diplomová práce, Vysoké učení technické v Brně, Fakulta informačních technologií, 2018.
URL http://www.fit.vutbr.cz/study/DP/DP.php?id=20939

[13] Matoušek, P.: *Síťové služby a jejich architektura*. Publishing house of Brno University of Technology VUTIUM, 2014, ISBN 978-80-214-3766-1, 396 s.
URL http://www.fit.vutbr.cz/research/view_pub.php.cs.iso-8859-2?id=10567

[14] Nucci, A.; Papagiannaki, K.: *Design, measurement and management of large-scale IP networks: Bridging the gap between theory and practice*. Cambridge University Press, 2009.

[15] Sadasivan, G.; Brownlee, N.; Claise, B.; aj.: Architecture for IP flow information export. Technická zpráva, IETF, 2009.

[16] Stallings, W.: SNMP and SNMPv2: the infrastructure for network management. *IEEE Communications Magazine*, ročník 36, č. 3, March 1998: s. 37–43, ISSN 0163-6804, doi:10.1109/35.663326.

[17] Wijnen, B.; Harrington, D.; Presuhn, R.: *An architecture for describing SNMP management frameworks. IETF, Duben 1990, [Online; navštíveno 01.04.2019].*
URL https://www.rfc-editor.org/rfc/pdfrfc/rfc2571.txt.pdf

[18] Wikipedie: Lexikografické uspořádání — Wikipedie: Otevřená encyklopedie. 2018, [Online; navštíveno 4. 04. 2019].
URL https://cs.wikipedia.org/w/index.php?title=Lexikografick%C3%A9_uspo%C5%99%C3%A1d%C3%A1n%C3%AD&oldid=16695418

[19] Wikisofia: Teorie systémů —. 2019, [Online; navštíveno 1. 04. 2019].
URL https://wikisofia.cz/index.php?title=Teorie_syst%C3%A9m%C5%AF&oldid=57289

Příloha A

Obsah přiloženého paměťového média

Návod na instalaci nezbytných balíčků a překlad modulu lze nalézt ve složce `source` v souboru `README.rst`. Složka `data` obsahuje soubory, které lze využít při ověřování funkčnosti modulu. Popis souborů a jejich využití lze nalézt ve stejné složce v souboru `README.txt`.

```
Kořenový adresář
├── source/.........Zdrojové kódy společně s návodem pro jejich přeložení
├── data/...........Soubory, které lze použít pro ověření funkčnosti
├── doc/............Zdrojové texty bakalářské práce
└── thesis.pdf......Text bakalářské práce
```

Příloha B

Pokročilá konfigurace statistického modulu

Konfigurace na obrázku B.1 je doplněna o volitelně nastavitelné položky konfigurace. Jedná se převážně o *timeout* hodnoty pro expiraci cache pamětí jednotlivých tabulek uvnitř exportní služby protokolu SNMP. Tyto položky mohou obsahovat pouze celá kladná čísla. Dalším nastavitelným parametrem, tentokrát již celého modulu, je hodnota *timeout* pro rozpoznání aktivity relace. Stejně jako v předchozím případě, tak i tato položka může obsahovat pouze celá kladná čísla.

```
<intermediate>
  <name>Collector statistics</name>
  <plugin>statistics</plugin>
  <params>
    <sessionActivityTimeout>1</sessionActivityTimeout>
    <outputs>
      <snmp>
        <cacheTimeout table="ipfixTransportSessionTable">
          1
        </cacheTimeout>
        <cacheTimeout table="ipfixTemplateTable">
          1
        </cacheTimeout>
        <cacheTimeout table="ipfixTemplateDefinitionTable">
          1
        </cacheTimeout>
        <cacheTimeout table="ipfixTransportSessionStatsTable">
          1
        </cacheTimeout>
        <cacheTimeout table="ipfixTemplateStatsTable">
          1
        </cacheTimeout>
      </snmp>
    </outputs>
  </params>
</intermediate>
```

Obrázek B.1: Konfigurace statistického modulu ve formátu XML, kde jsou specifikovány všechny volitelné položky.