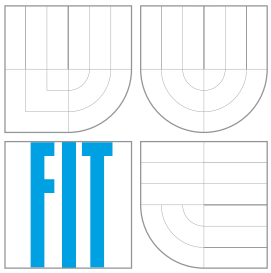
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

# SONDA PRO MONITOROVÁNÍ APLIKAČNÍCH PROTOKOLŮ

PROBE FOR THE APPLICATION PROTOCOLS MONITORING

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. TOMÁŠ FUKAČ

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. JAN VIKTORIN

BRNO 2016

*Zadání diplomové práce/18242/2015/xfukac00*

**Vysoké učení technické v Brně - Fakulta informačních technologií**

Ústav počítačových systémů Akademický rok 2015/2016

# Zadání diplomové práce

Řešitel: **Fukač Tomáš, Bc.**
Obor: Počítačové a vestavěné systémy
Téma: **Sonda pro monitorování aplikačních protokolů**
**Probe for the Application Protocols Monitoring**
Kategorie: Počítačová architektura

Pokyny:

1. Seznamte se s prototypem mikrosondy pro monitorování IPv6 provozu vyvinutého v rámci projektu Moderní prostředky pro boj s kybernetickou kriminalitou na Internetu nové generace (VG20102015022).
2. Seznamte se s možnostmi monitorováni aplikačních protokolů. Zvolte vhodnou sadu několika těchto protokolů (např. SIP, POP3, FTP, apod.). Vybranou sadu konzultujte s konzultantem práce.
3. Navrhněte do mikrosondy úpravy a rozšíření pro monitorování vybraných aplikačních protokolů.
4. Navržené úpravy a rozšíření implementujte.
5. Otestujte funkčnost vaší implementace v testovacím prostředí.
6. Diskutujte dosažené výsledky.

Literatura:

- Dle pokynů vedoucího.

Při obhajobě semestrální části projektu je požadováno:

- Splnění bodů 1 a 2 zadání.

Podrobné závazné pokyny pro vypracování diplomové práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva diplomové práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap, které byly vyřešeny v rámci dřívějších projektů (30 až 40% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Viktorin Jan, Ing.**, UPSY FIT VUT
Konzultant: Korček Pavol, Ing., UPSY FIT VUT
Datum zadání: 1. listopadu 2015
Datum odevzdání: 25. května 2016

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačových systémů a sítí
612 66 Brno, Božetěchova 2

doc. Ing. Zdeněk Kotásek, CSc.
*vedoucí ústavu*

# Abstrakt

Tato práce se zabývá rozšířením funkcionality Mikrosondy o detekci a filtrování aplikačních protokolů. Mikrosonda je vestavěný systém, který je určen pro monitorování síťových linek o rychlosti 1 Gb/s. Detekce aplikačních protokolů vyžaduje použití technik pro vyhledávání řetězců a vzorů definovaných regulárním výrazem, což jsou operace náročné na výpočetní výkon zařízení. Na základě studia vybraných protokolů (SMTP, POP3, FTP, SIP) a stávající aplikace Mikrosonda byl vytvořen návrh rozšíření, které rozděluje funkcionalitu mezi FPGA a procesor. V FPGA probíhá předzpracování síťového provozu, které spočívá v hledání požadovaných identifikátorů uživatelů a vzorů specifických pro daný protokol. Na procesoru je následně ověřeno, zda se jedná o požadovanou komunikaci. Procesor tedy nemusí zpracovávat celý síťový provoz, ale jen část předvybranou v FPGA. Softwarová část je rozšířena o modul pro analýzu SMTP komunikace, který umožňuje zpracovávat více než 5 000 síťových toků za sekundu. Podporu dalších protokolů lze přidat pouhým rozšířením softwarové části.



# Abstract

This work describes an extension of the Microprobe functionality for detection and filtering of application protocols. The Microprobe is an embedded system designed for monitoring network links at speed 1 Gb/s without loosing any packets. The detection of application protocols requires using of computationally expensive operations, especially string lookup (usually based on regular expressions). Based on the study of several protocols (SMTP, POP3, FTP, SIP) a draft of a new architecture has been created. The new architecture splits this functionality between programmable logic FPGA and processor. The FPGA performs preprocessing of network traffic consisting of a lookup for user identifiers and protocol-specific patterns. The processor verifies that it is the requested communication. The processor does not need to process the entire network traffic but only the part prefiltered in the FPGA. The software part is extended by a module for the analysis of SMTP which allows processing of more than 5,000 network flows per second. Support for other protocols can be added by an extension of the software part.



# Klíčová slova

Mikrosonda, FPGA, SMTP, POP3, FTP, SIP, NetModule ZE7000

# Keywords

Microprobe, FPGA, SMTP, POP3, FTP, SIP, NetModule ZE7000


# Citace

FUKAČ, Tomáš. *Sonda pro monitorování aplikačních protokolů*. Brno, 2016. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Viktorin Jan.

# Sonda pro monitorování aplikačních protokolů

## Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Jana Viktorina a uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Tomáš Fukač
23. května 2016


## Poděkování

Rád bych poděkoval Ing. Janu Viktorinovi za cenné rady, věcné připomínky a vstřícnost při konzultacích. Jsem rád, že byl jako vedoucí mé diplomové práce důsledný a pečlivý. Tato práce vnikla za podpory projektu *Sondy pro analýzu a filtraci provozu na úrovni aplikačních protokolů*.

# Obsah

# Kapitola 1

# Úvod

Monitorování síťových linek o rychlosti 1 Gb/s při plné rychlosti bez ztráty jediného paketu vyžaduje velký výpočetní výkon. Pokud však chceme pro tento účel vytvořit malé vestavěné zařízení s nízkou spotřebou, nebude možné využít dostatečně výkonný procesor. Řešením je využití vestavěného systému, který obsahuje FPGA (Field Programmable Gate Array). FPGA nabízí dostatečný výkon pro analýzu paketů na transportní vrstvě. Pro realizaci analýzy aplikačních protokolů však není dostatečně flexibilní a implementace analyzátorů aplikačních protokolů v jazycích pro popis hardwaru je obtížná.

Pro monitorování sítových linek o rychlosti 1 Gb/s byla vytvořena aplikace Mikrosonda. Ta slouží pro klasifikaci a filtrování provozu u menších poskytovatelů internetového připojení. V této síti umožňuje monitorování veškeré komunikace a sběr požadovaných dat, která jsou následně doručena do sběrného místa. Klasifikace a filtrování jsou časově náročné úkony. Z tohoto důvodu jsou akcelerovány v FPGA.

Mikrosonda je vybudována na platformě NetModule ZE7000, která obsahuje pět síťových rozhraní a čip Xilinx Zynq, který v sobě kombinuje programovatelné pole FPGA a procesor ARM Cortex-A9. Na jednom čipu lze tedy současně spouštět aplikace na procesoru ARM a akcelerovat náročné výpočty v FPGA.

Mikrosonda však umožňuje filtraci pouze předem daného síťového toku, protože klasifikace a filtrování probíhá na úrovni transportní vrstvy. Tato práce rozšiřuje Mikrosondu o možnost filtrace pouze konkrétního protokolu případně pouze komunikaci jednoho konkrétního uživatele.

Rozšířená implementace je rozdělena mezi FPGA a procesor. V FPGA je prováděno předzpracování síťového provozu, které spočívá ve vyhledávání konkrétních uživatelských identifikátorů nebo vzorů specifických pro požadovaný protokol. Do softwaru běžícím na procesoru jsou zaslány toky, které obsahují tyto identifikátory nebo vzory. Software následně zkontroluje, zda se jedná o požadovaný síťový tok.

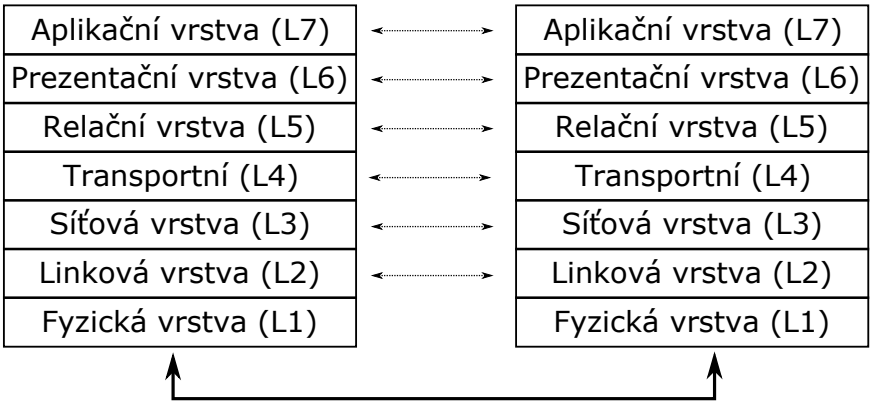
# Kapitola 2

# Monitorování aplikačních protokolů

V této kapitole je představen základní model vrstev síťové komunikace a několik základních aplikačních protokolů. Ke každému protokolu je následně popsán způsob jejich detekce a shromažďování požadovaných dat.

## 2.1 Referenční model ISO/OSI

Referenční model ISO/OSI [13] byl navržen mezinárodní organizací International Organization for Standardization (ISO) jako abstraktní model síťové komunikace. Model se skládá ze sedmi vrstev, které jsou znázorněny na obrázku 2.1 a popsány níže. Veškerá síťová komunikace probíhá mezi vrstvami na stejné úrovni s využitím nižších vrstev.



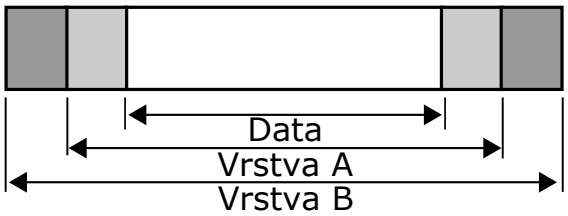

Obrázek 2.1: Vrstvy referenčního modelu ISO/OSI [13, s. 32]

- **Fyzická vrstva** – zabezpečuje přenos jednotlivých bitů mezi síťovými uzly, definuje všechny fyzikální a elektrické vlastnosti zařízení, specifikuje přenosové médium.
- **Linková vrstva** – seskupuje data z fyzické vrstvy do větších logických celků (rámců), které jsou opatřeny fyzickou adresou uzlů, mechanismy pro detekci a případně i opravu chyb vniklých na fyzické vrstvě.

- **Síťová vrstva** – má na starosti síťové adresování a směrování v síti, poskytuje funkce pro přenos dat různé délky mezi různými sítěmi při zachování požadované kvality služby.
- **Transportní vrstva** – první vrstva, která zajišťuje přenos dat pouze mezi koncovými uzly, kvalita tohoto přenosu je následně specifikována vyššími vrstvami.
- **Relační vrstva** – umožňuje vytvořit, synchronizovat, ukončit a obnovit spojení, uchovávat k němu informace.
- **Prezentační vrstva** – slouží pro převod dat do tvaru, který je používán na koncovém uzlu (pořadí bajtů, šifrování, komprimace, konvertování, atd.).
- **Aplikační vrstva** – účelem vrstvy je poskytovat aplikacím vzájemnou komunikaci.

Každá z uvedených vrstev plní přesně definované funkce, pro které využívá služby nižší vrstvy. Zdroj tedy vytvoří aplikační požadavek, který je předán prezentační vrstvě, následně je zpracován relační vrstvou atd. Tímto způsobem je požadavek předávám k nižším vrstvám a to až k vrstvě fyzické, která zprostředkovává přístup k přenosovému médiu.

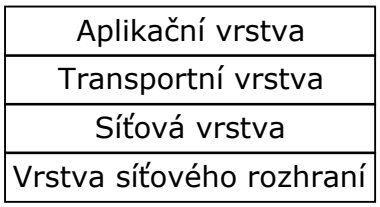
Při předávání aplikačních dat nižším vrstvám se k nim přidávají hlavičky (případně i patičky) specifické pro tuto vrstvu, čímž dochází k opakovanému zapouzdření dat. Při postupu od nižších vrstev k vyšším jsou využívány řídicí informace z hlaviček (patiček), které jsou postupně odstraňovány.




Obrázek 2.2: Zapouzdření dat

## 2.2 Rodina protokolů TCP/IP

Architektura protokolů rodiny TCP/IP [3] je velmi podobná referenčnímu modelu ISO/OSI, přestože vnikala v době, kdy tento model ještě neexistoval. Protokoly rodiny TCP/IP využívají vrstvový model podobně jako referenční model ISO/OSI, oproti kterému mají pouze čtyři vrstvy. Tyto vrstvy jsou naznačeny na obrázku 2.3 a popsány níže.




Obrázek 2.3: Vrstvy modelu TCP/IP

- **Vrstva síťového rozhraní** – je nejnižší vrstva, má přístup k fyzickému přenosovému médiu a je specifická pro použitou síť. Příkladem sítí jsou Ethernet, Token ring, FDDI, 100BaseVG, X.25, SMDS a další. V referenčním modelu ISO/OSI odpovídá fyzické a linkové vrstvě. Z pohledu vyšších vrstev tato vrstva přenáší data po blocích (rámcích).
- **Síťová vrstva** – zajišťuje především adresaci a směrování v síti. Vrstva odpovídá síťové vrstvě v referenčním modelu ISO/OSI a je součástí implementace všech síťových prvků (směrovacích i koncových). Příkladem protokolů této vrstvy jsou IP, ARP, RARP, ICMP, IGMP, IGRP, IPSEC.
- **Transportní vrstva** – je implementována pouze v koncových zařízeních. V modelu ISO/OSI odpovídá transportní vrstvě a obsahuje některé služby relační vrstvy (udržováni stavových informací o konkrétním spojení). Protokoly této vrstvy jsou spolehlivý protokol TCP a nespolehlivý protokol UDP.
- **Aplikační vrstva** – zajišťuje služby relační vrstvy (nepokryty předchozí vrstvou), prezentační vrstvy a aplikační vrstvy referenčního modelu ISO/OSI. Aplikační vrstva tedy musí zajistit kromě komunikace mezi aplikacemi také převod dat do správného tvaru.

Strukturu přenášených dat, způsob navazování spojení a způsob ukončování spojení na dané vrstvě určují síťové protokoly. V dalších kapitolách budou popsány nejpoužívanější protokoly síťové a transportní vrstvy (IP, TCP, UDP), které identifikují síťový tok, a několik vybraných aplikačních protokolů, které by mohli být z pohledu zákonných odposlechů zajímavé.

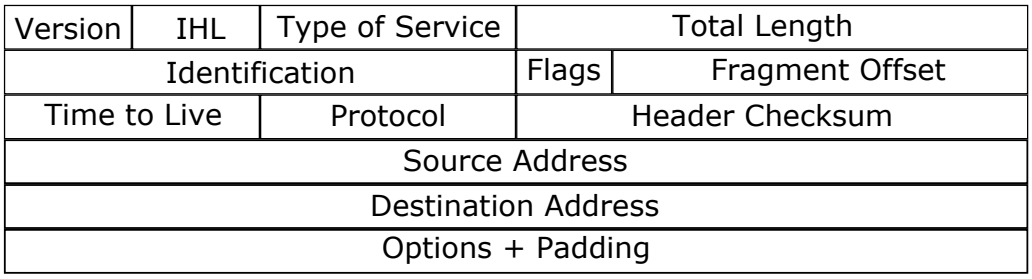
### 2.2.1 Internet Protocol

Internet Protocol [3][1] je základní protokol síťové vrstvy, který poskytuje vyšším vrstvám nespojovanou síťovou službu. Přenos každého datagramu je nezávislý na ostatních a je prováděn na základě síťových IP adres (identifikace zdroje a cíle), které se nacházejí v hlavičce tohoto protokolu. IP protokol nezajišťuje spolehlivé doručení ani doručení datagramů v pořadí, ve kterém byly odeslány. Tyto služby musí zajistit vyšší vrstvy.

Tento protokol dále umožňuje protokolům vyšších vrstev provádět segmentaci dat, která je důležitá pro přenos datagramů, jejichž velikost přesahuje maximální povolenou velikost dat pro přenos nižšími vrstvami. [1, s. 1]

Aktuálně se používá Internet Protocol verze 4 (IPv4) a jeho následník označený jako verze 6 (IPv6). Hlavička protokolu IPv4 je znázorněna na obrázku 2.4 a obsahuje níže uvedené položky. [1, s. 11–13]

- **Version** – obsahuje číslo verze protokolu, v tomto případě tedy 4 (u novější verze potom 6).
- **Internet Header Length (IHL)** – určuje velikost hlavičky jako počet 32-bitových slov.
- **Type of Service** – tato položka slouží pro určení typu služby a tak zajištění požadované kvality přenosu.
- **Total Length** – udává celkový počet oktetů (bytů) dat včetně hlavičky.



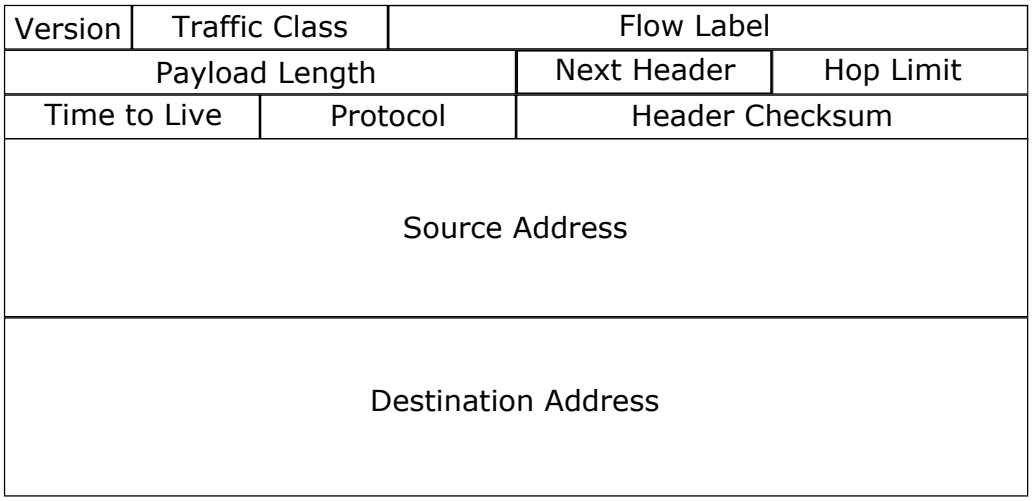

Obrázek 2.4: Formát hlavičky IPv4 [1, s. 11]

- **Identification** – obsahuje identifikační hodnotu fragmentované části datagramu.
- **Flags** – tři bity používané při fragmentaci, kterými je možné povolit nebo zakázat další fragmentaci, identifikovat poslední fragment nebo skutečnost, že bude následovat fragment.
- **Fragment Offset** – označuje pozici fragmentu ve výsledných datech.
- **Time To Live** – hodnota určuje maximální dobu, po kterou se může datagram nacházet na síti. Při každém průchodu směrovačem je hodnota snížena o jedna, pokud hodnota klesne na nulu, datagram je zahozen a zdroji je zaslána chybová zpráva. Tento mechanismus zamezuje nekonečnému předávání datagramu mezi směrovači v kruhu.
- **Protocol** – identifikuje protokol, který je přenášen (většinou se jedná o protokol vyšší vrstvy).
- **Header Chechsum** – obsahuje kontrolní součet hlavičky.
- **Source (Destination) Address** – udává 32-bitová IP adresa zdrojového (cílového) uzlu.
- **Options + Padding** – volitelné položky hlavičky, kterými je například požadovaná cesta datagramu a zabezpečení, padding potom značí zarovnání na 32-bitů.

Především kvůli nedostačujícímu počtu IP adres byla navržena nová verze IP protokolu, která byla označena jako verze 6. Původní 32-bitové IP adresy byly nahrazeny za 128-bitové adresy, byl zvýšen důraz na bezpečnost, byla přidána možnost přenášet datagramy o větší velikosti (označované jako tvz. jumbogramy) atd. [3][8]

Tvar hlavičky je značně odlišný od předchozí verze (viz obrázek 2.5) což znemožňuje zpětnou kompatibilitu. IPv6 hlavička obsahuje oproti předchozí verzi níže uvedené položky. [8, s. 4–5]

- **Traffic Class** – slouží pro odlišení různých tříd či priorit.
- **Flow Label** – umožňuje zdroji označit datagramy, které požadují specifické zacházení při směrování, lze ji tedy využít pro zajištění kvality služby.
- **Payload Length** – udává počet oktetů (bytů) přenášených dat, oproti předchozí verzi již není započtena velikost IP hlavičky. Pokud je pole nastaveno na nulu, jedná se o zmíněný jumbogram.




Obrázek 2.5: Formát hlavičky IPv6 [8, s. 4]

- **Next Header** – pole udává typ hlavičky, která bude následovat po IP hlavičce. Odpovídá poli Protocol v IPv4, jehož hodnoty jsou rozšířeny o speciální hlavičky.
- **Hop Limit** – jeho význam je shodný s polem Time to Live u IPv4.
- **Source (Destination) Address** – 128-bitová IP adresa zdrojového (cílového) uzlu.

Za IP hlavičkou následují data vyšších protokolů (např. protokoly TCP, UDP).

### 2.2.2 User Datagram Protokol

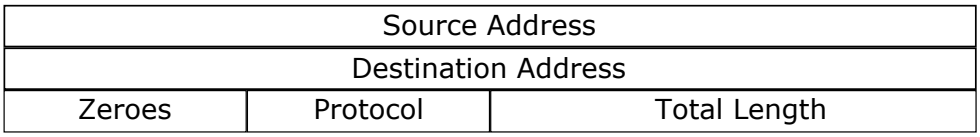
User Datagram Protocol (UDP) poskytuje nespolehlivé nespojované služby transportní vrstvy TCP/IP modelu. Protokol je využíván v aplikacích, které nevyžadují spolehlivost přenosu nebo ji implementují na vyšších vrstvách nebo které vyžadují nízké zpoždění ale respektují jistou úroveň ztrátovosti (audiovizuální aplikaci apod). [3, 19]

Tvar UDP hlavičky je znázorněna na obrázku 2.6 a obsahuje následující položky:

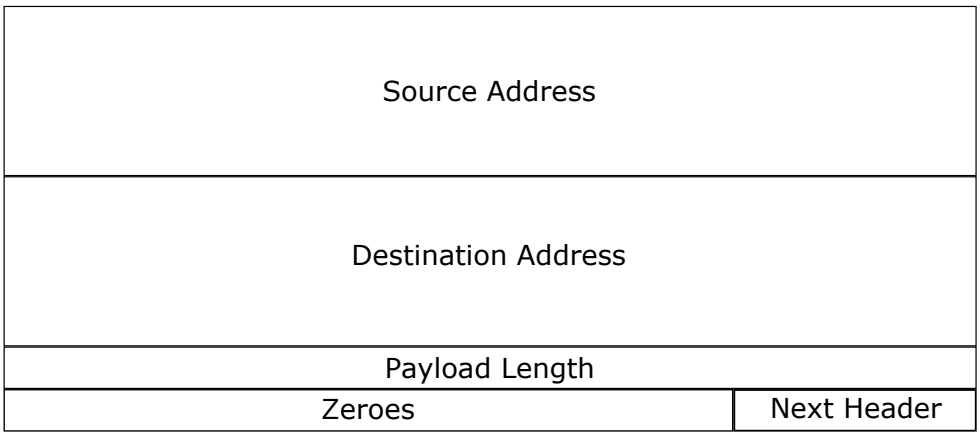

Obrázek 2.6: Formát hlavičky protokolu UDP [19, s. 1]

- **Source (Destination) Port** – obsahuje číslo portu, které jednoznačně identifikuje zdrojovou (cílovou) aplikaci.
- **Length** – určuje počet bytů následujících dat včetně velikosti UDP hlavičky.
- **Checksum** – obsahuje kontrolní součet IP pseudohlavičky, UDP hlavičky a dat.

Pseudohlavička protokolu IP není skutečně přenášená hlavička, využívá se pouze pro výpočet kontrolního součtu. Obsahuje vybrané položky IP hlavičky. Její tvar se liší podle veze protokolu, na obrázku 2.7 je znázorněn tvar pro IPv4, na obrázku 2.8 tvar pro IPv6.




Obrázek 2.7: Formát pseudohlavičky protokolu IPv4 [19, s. 2]

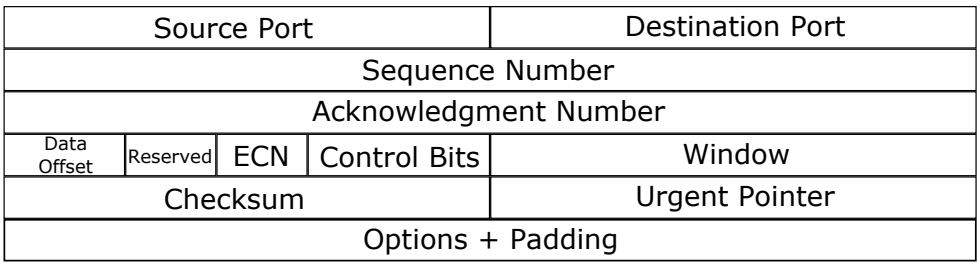



Obrázek 2.8: Formát pseudohlavičky protokolu IPv6 [8, s. 27]

### 2.2.3 Transmission Control Protocol

Transmission Control Protocol (TCP) je spojovaný spolehlivý protokol transportní vrstvy TCP/IP modelu. Jedná se o stavový protokol (hlavička obsahuje informace k příslušnému spojení), využívá klouzavé okno a pozitivní potvrzování doručení. Tyto principy umožňují zabránění zahlcení sítě a seřazení datagramů do pořadí, ve kterém byly odeslány. [3, 2]

Formát hlavičky protokolu je zobrazen na obrázku 2.9. Níže jsou popsány položky hlavičky. [2, s. 15–16]




Obrázek 2.9: Formát hlavičky protokolu TCP [2, s. 15]

- **Source (Destination) Port** – obsahuje číslo portu, které jednoznačně identifikuje zdrojovou (cílovou) aplikaci.
- **Sequence Number** – pokud není nastaven příznak SYN, pole obsahuje sekvenční číslo, pokud je příznak SYN nastaven, pole obsahuje inicializační hodnotu.
- **Acknowledgement Number** – pokud je nastaven příznak ACK pole obsahuje sekvenční číslo, které je očekáváno příjemcem (zároveň jsou potvrzena všechna nižší sekvenční čísla).

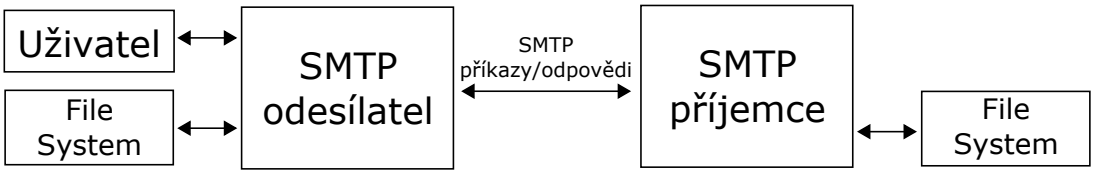
- **Data Offset** – udává velikost TCP hlavičky jako počet 32-bitových slov.
- **Explicit Congestion Notification** – políčko slouží pro signalizaci zahlcení sítě. Toto rozšíření bylo přidáno později v RFC 3168 [22].
- **Control Bits** – obsahuje kontrolní bity:
  - URG – určuje, zda je platná položka Urgent Pointer,
  - ACK – udává, zda je platná položka Acknowledgement Number,
  - PSH – využití tzv. push funkce,
  - RST – reset spojení,
  - SYN – synchronizace sekvenčních čísel,
  - FIN – ukončení spojení odesílatelem.
- **Window** – udává počet oktetů, které je schopen odesílatel odeslat (aniž by pro ně přijal potvrzení).
- **Urgent Pointer** – je využito, pokud je nastaven příznak URG a obsahuje sekvenční číslo posledního urgentního oktetu dat.
- **Options + Padding** – volitelné položky hlavičky, které jsou zarovnány na 32 bitů.

Za hlavičkou UDP (TCP) následují data aplikační vrstvy. Protokoly IP a UDP (TCP) následně specifikují síťový tok, který je definován jako datagramy (pakety), které prochází daným bodem sítě a které mají shodnou zdrojovou a cílovou IP adresu, zdrojový a cílový port a protokol. Síťový tok tedy specifikuje konkrétní komunikaci aplikačních vrstev.

V dalších kapitolách se budeme zabývat konkrétními protokoly aplikační vrstvy modelu TCP/IP.

## 2.3 Simple Mail Transfer Protocol

Elektronická pošta (e-mail) je jedna z nejpoužívanějších síťových služeb. Většina služeb pro zasílání pošty využívá protokol Simple Mail Transfer Protocol (SMTP). Tento protokol je využit pro přenos e-mailů mezi SMTP servery. Klient přistupuje k poště ve své poštovní schránce na serveru buď pomocí protokolu Post Office Protocol (POP), který je popsán v kapitole 2.4 nebo pomocí protokolu Internet Message Access Protocol (IMAP). [24, s. 1]




Obrázek 2.10: Model SMTP [21, s. 2]

SMTP je protokol aplikační vrstvy, který pro spolehlivý přenos pošty využívá protokoly TCP a IP. Model komunikace vypadá tak, že na základě uživatelova požadavku zaslat e-mail SMTP odesílatel (software zabezpečující odesílání e-mailů) inicializuje TCP spojení s cílovým SMTP serverem a pokusí se odeslat poštu prostřednictvím tohoto spojení. Na cílovém SMTP serveru je spuštěn SMTP příjemce, který zajišťuje příjem e-mailů. SMTP odesílatel inicializuje spojení na stanoveném portu, na kterém SMTP příjemce naslouchá. Po úspěšném navázání spojení začne dialog, kdy SMTP odesílatel zasílá požadavky (příkazy) a od SMTP příjemce přijímá odpovědi. Příkazy jsou zasílány v ASCII kódování. [21, s. 2–3]

### 2.3.1 Formát zprávy

Zpráva začíná hlavičkou e-mailu, která se skládá z klíčových slov a hodnot, kterými lze specifikovat čas odeslání, odesilatele atd. Hlavička e-mailu má tvar seznamu řádků, které se skládají z názvu položky, oddělovače (dvojtečka), bílého znaku (tabulátor nebo mezera) a hodnoty proměnné. Pokud má položka více argumentů, jsou další hodnoty uvedeny na následujících řádcích začínajících bílým znakem: [24, s. 4]

jméno-položky: hodnota-položky<CRLF>
hodnota-položky<CRLF>

Názvy položek a jejich význam je následující:

- DATE – specifikuje čas vytvoření zprávy,
- TO – primární příjemci zprávy,
- CC – sekundární příjemci zprávy (kopie zprávy),
- FROM – identifikuje odesílatele zprávy,
- REPLY-TO – identifikuje příjemce odpovědi na zprávu,
- SUBJECT – specifikuje předmět zprávy (shrnutí obsahu zprávy).

Za hlavičkou je vložen oddělovat (<CRLF>), který je následován textem zprávy. Zpráva je ukončena kombinací <CRLF>.<CRLF>. Příklad hlavičky a zprávy e-mailu může vypadat například následovně: [21, s. 23]

```
Date: 27 Oct 81 15:01:01 PST
From: JOE@ABC.ARPA
Subject: Improved Mailing System Installed
To: SAM@JKL.ARPA

Text zprávy ...
.
```

Na prvních čtyřech řádcích se nachází hlavička e-mailu. Ta popisuje datum a čas odeslání, odesilatele, předmět a příjemce e-mailu. Za hlavičkou následuje text zprávy.

### 2.3.2 Příkazy protokolu

Syntaxe a význam vybraných příkazů, které zasílá SMTP odesílatel, je následující: [21, s. 19–26]

- EHLO <SMTP odesílatel> – tímto příkazem se příjemci specifikuje odesílatel, u starších verzí protokolu se objevuje jako příkaz HELO případně HELLO.
- MAIL FROM: <adresa odesílatele> – specifikuje poštovní schránku uživatele odesílajícího zprávu (např. John.Smith@USC-ISI.ARPA).
- RCPT TO: <adresa příjemce> – slouží pro specifikaci cílové poštovní schránky.
- DATA <CRLF> – za tímto příkazem následuje obsah zprávy, jejíž tvar byl popsán v předchozí kapitole.
- RSET <CRLF> – slouží pro zrušení transakce.
- VRFY <string> <CRLF> – příkazem je možné ověřit, zda argument specifikuje poštovní schránku.
- QUIT <CRLF> – po zaslání tohoto příkazu příjemce odpoví zprávou OK a dojde k ukončení spojení.

Na každý zaslaný příkaz je očekávána odpověď. Odpověď obsahuje tříciferné číslo, které udává sémantiku. Za tímto číslem je vložena mezera (oddělovač), která je následována doprovodným textem. Tento text má pouze informativní účel pro uživatele. Odpověď je zakončena znaky <CRLF>. [21, s. 34]

Každá číslice odpovědi má speciální význam. První z číslic udává, zda je odpověď pozitivní, negativní nebo příslušný příkaz není dokončen. Číslice může nabývat jedné z pěti hodnot: [21, s. 48–49]

- 1yz – předběžná pozitivní odpověď (příkaz byl přijat ale byl pozastaven, protože je nutné ho dalším příkazem potvrdit),
- 2yz – pozitivní odpověď pro dokončený příkaz,
- 3yz – předběžná pozitivní odpověď (příkaz byl přijat, ale byl pozastaven, protože jsou vyžadovány další informace),
- 4yz – předběžná negativní odpověď (příkaz nebyl přijat ani vykonán, jedná se o dočasnou chybu),
- 5yz – negativní odpověď.

Druhá číslice udává kategorii, které odpověď přísluší. Číslice může nabývat jedné ze tří hodnot: [21, s. 49]

- x1z – syntaktická,
- x2z – informační,
- x5z – systémová.

Třetí číslice následně udává konkrétní odpověď z příslušné kategorie. Níže je uveden výčet několika nejdůležitějších odpovědí: [21, s. 35–36]

- 220 <doména> – služba je připravena,
- 421 <doména> – služba není k dispozici,
- 250 – příkaz byl v pořádku dokončen,
- 354 – SMTP příjemce očekává obsah zprávy,
- 500 – syntaktická chyba, příkaz nebyl rozeznán,
- 554 – transakce skočila neúspěchem.

Pro zaslání pošty je nutné, aby SMTP odesílatel ustanovit spojení s cílovým SMTP serverem (a jeho SMTP příjemcem). SMTP příjemce zašle odpověď 220, kterou SMTP odesilateli potvrdí navázání spojení a kterou se zároveň i identifikuje. SMTP odesilatel se následně identifikuje příkazem EHLO, na který dostává kladnou odpověď. SMTP odesilatel dále zašle příkaz MAIL, kterým identifikuje poštovní schránku uživatele odesílajícího poštu. Pokud taková schránka na serveru existuje, SMTP příjemce odpoví pozitivně. Následně odesílatel příkazem RCPT identifikuje cílové poštovní schránky uživatelů, kterých může být více. Pokud se na serveru nachází schránka pro tohoto uživatele, je to SMTP odesilateli oznámeno pozitivní odpovědí, v opačném případě je SMTP odesílateli zaslána chybová odpověď (ta platí pouze pro tuto poštovní schránku, nikoli na celou transakci). Opakovaným posíláním příkazu RCPT je tedy možné specifikovat více poštovních schránek, do kterých bude zaslána stejná zpráva. V rámci jednoho spojení lze však specifikovat pouze schránky, které se nachází na daném serveru. Pro schránky nacházející se na různých doménách je nutné navázat více spojení (každé s požadovaným serverem). Po specifikování všech uživatelů SMTP odesílatel pošle příkaz DATA, na který od SMTP příjemce dostává odpověď 354. SMTP příjemce nyní očekává zprávu, která je ukončena speciální sekvencí znaků <CRLF>.<CRLF>. Pokud SMTP příjemce úspěšně zprávu zpracuje, pozitivně odpoví. Zpráva je příjemcům odeslána po ukončení spojení příkazem QUIT. [21, s. 3–5]

Komunikace může vypadat např. následovně ($S$ značí SMTP odesílatele a $R$ SMTP příjemce): [21, s. 16, 63]

```
R: 220 HOSTW.ARPA Simple Mail Transfer Service Ready
S: HELO HOSTY.ARPA
R: 250 HOSTW.ARPA
S: MAIL FROM:<SMTP@HOSTY.ARPA>
R: 250 ok
S: RCPT TO:<@HOSTX.ARPA:JOE@HOSTW.ARPA>
R: 250 ok
S: DATA
R: 354 send the mail data, end with .
S: Date: 23 Oct 81 11:22:33
S: From: SMTP@HOSTY.ARPA
S: To: JOE@HOSTW.ARPA
S: Subject: Mail System Problem
S:
```

```
S:   Sorry JOE, your message to SAM@HOSTZ.ARPA lost.
S:   HOSTZ.ARPA said this:
S:    "550 No Such User"
S: .
R: 250 ok
```

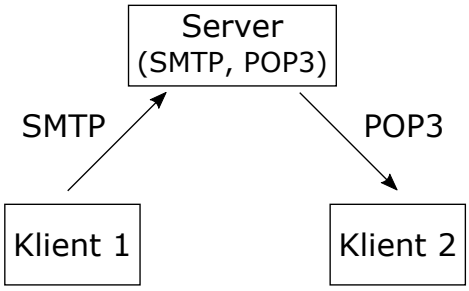
### 2.3.3 Detekce protokolu

Z předcházejícího textu vyplývá, že detekci SMTP komunikace lze provádět na základě přítomnosti kombinace konkrétních SMTP příkazů (např. `MAIL FROM` a `RCPT TO`) a identifikátoru e-mailové schránky. Pro účely této práce bude uvažován následující algoritmus detekce SMTP provozu s přihlédnutím k potenciálnímu rozdělení úlohy mezi hardwarovou a softwarovou část:

1. Nejprve je v síťovém provozu vyhledávána požadovaná e-mailová adresa (akcelerováno hardwarem).
2. Pokud je adresa nalezena, daný síťový tok je považován za potenciálně zájmový, a proto je předán k dalšímu zpracování.
3. Protože mohla být e-mailová adresa nalezena v kontextu, kde se nejedná o SMTP (webová stránka atd.), lze dodatečně ověřit, zda je v prvním exportovaném paketu přítomen příkaz `MAIL FROM` nebo `RCPT TO` včetně požadované e-mailové adresy.
4. Po ukončení vybraného TCP spojení je export tohoto toku zastaven.

## 2.4 Post Office Protocol

Pro běžné koncové uzly je provoz SMTP serveru velmi nepraktický a drahý, protože pro doručování pošty pouze pomocí protokolu SMTP by musel být uzel neustále v provozu a nepřetržitě připojen k síti. Z tohoto důvodu byl navržen protokol Post Office Protocol (POP). POP je aplikační protokol, který se využívá pro stažení e-mailových zpráv ze vzdáleného serveru, kde jsou shromažďovány e-mailové zprávy, na lokální počítač poštovním klientem. Na lokální počítač je obvykle stažena veškerá pošta, která je následně ze serveru smazána. V současné době se téměř výhradně používá protokol POP verze 3 (POP3). [16, s. 2–3]




Obrázek 2.11: Model komunikace s e-mailovým serverem

### 2.4.1 Architektura protokolu

Model zaslání zprávy (obrázek 2.11) z jednoho klienta pomocí protokolu SMTP a stažení pomocí protokolu POP3 do druhého klienta vypadá tak, že se první klient spojí s e-mailovým serverem, na kterém běží služby SMTP a POP3. Pomocí protokolu SMTP odešle zprávu na tento server. Druhý klient se později na tentýž server připojí a stáhne tuto zprávu protokolem POP3. [16, s. 2]

Hostitelský server tedy nejdříve spustí službu POP3, čímž započne naslouchat na TCP portu číslo 110 (pokud není stanoven jiný). Klient, který chce využít tuto službu, naváže TCP spojení s tímto serverem na odpovídajícím portu. Jakmile je spojení navázáno, server odešle klientovi pozitivní odpověď (popsáno níže). Následně klient odesílá příkazy, na které server odpovídá.

### 2.4.2 Příkazy a odpovědi protokolu

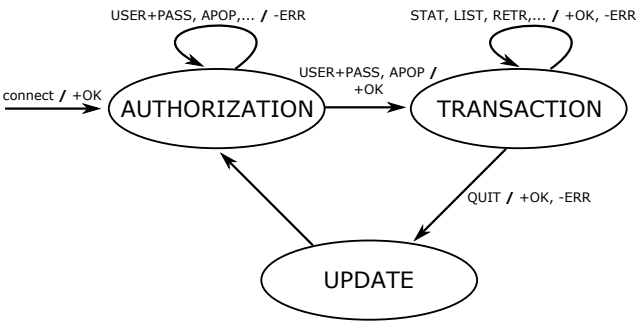
Příkazy protokolu POP3 se skládají z klíčového slova, po kterém následuje jeden nebo více argumentů. Příkaz je následně zakončen znaky <CRLF>. Klíčová slova i argumenty se skládají z tisknutelných ASCII znaků.

POP3 odpověď se skládá z identifikátoru statusu (+ nebo -), klíčového slova (OK nebo ERR) a případně další informace. Odpověď začíná jedním ze dvou statusů:

- $+OK$ – pozitivní status,
- $-ERR$ – negativní status.

Každá odpověď je opět zakončena znaky <CRLF>. [16, s. 3]

POP3 sezení přechází během své existence mezi několika stavy. Jakmile je navázáno TCP spojení se serverem a server pošle pozitivní odpověď, relace přejde do stavu AUTHORIZATION. V tomto stavu se klient musí identifikovat serveru. Po úspěšné autorizaci přejde sezení do stavu TRANSACTION, kdy klient může provádět akce na straně serveru. Po zaslaní příkazu pro ukončení spojení, server vrací pozitivní (v případě chyby negativní) odpověď, sezení přejde do stavu UPDATE, ve kterém server uvolní všechny prostředky (viz. obrázek 2.12).




Obrázek 2.12: Graf přechodů FTP sezení

Po úspěšném navázání spojení s POP3 serverem, je klientovy zaslána odpověď, která může vypadat např. následovně:

```
+OK POP3 server ready
```

POP3 sezení se nyní nachází ve stavu AUTHORIZATION. Klient se nyní musí identifikovat serveru pomocí kombinace příkazů příkazů USER a PASS nebo příkazem APOP (podrobněji popsáno níže). Po úspěšném ověření klienta se server pokusí získat exkluzivní přístup (zámek) k požadované poštovní schránce. Po úspěšném získání přístupu ke schránce je klientovy poslána pozitivní odpověď a relace přechází do stavu TRANSACTION. [16, s. 3–4]

Ověření uživatele je možné pomocí dvou postupů. Prvním z nich je použití příkazu USER pro specifikaci poštovní schránky a následně příkaz PASS pro zadání hesla. Každý z příkazů může odpovědět pozitivně nebo negativně (uživatel neexistuje, heslo je nesprávné): [16, s. 13–14]

```
C: USER mrose
S: +OK mrose is a real hoopy frood
C: PASS secret
S: +OK mrose's maildrop has 2 messages (320 octets)
```

V tomto případě se na síti objevuje heslo v čitelné podobě.

Tuto skutečnost je možné řešit příkazem APOP. Server při připojení klienta zašle klientovy v odpovědi řetězec tvaru:

```
<process-ID.clock@hostname>
```

Tento řetězec obsahuje identifikaci procesu POP3 serveru, aktuální čas a doménové jméno serveru. Argumenty příkazu APOP jsou identifikace schránky a speciální řetězec. Tento řetězec je vytvořen pomocí MD5 hash funkce. Na vstup hash funkce je přiveden řetězec z prvotní odpovědi serveru spolu s předem domluveným sdíleným tajemstvím (heslem). Příklad tohoto způsobu ověření je následující:

```
S: +OK POP3 server ready <1896.697170952@dbc.mtview.ca.us>
C: APOP mrose c4c9334bac560ecc979e58001b3e22fb
S: +OK maildrop has 1 message (369 octets)
```

Vstupem MD5 algoritmu je řetězec („tanstaaf“ je sdílené tajemství):

```
<1896.697170952@dbc.mtview.ca.us>tanstaaf
```

a výstupem je následně: [16, s. 15–16]

```
c4c9334bac560ecc979e58001b3e22fb
```

Každé zprávě, která se na serveru nachází, je přiřazen identifikátor (kladné číslo). Příkazem STAT může klient zjistit stav schránky. Odpověď je tvaru (*num* značí počet zpráv a *size* celkovou velikost všech zpráv):

```
+OK num size
```

Volání příkazu a odpověď může vypadat následovně ($C$ značí klienta a $S$ server):

```
C: STAT
S: +OK 2 320
```

Příkazem LIST je možné zjistit informace o všech zprávách. Pokud se ve schránce nenachází žádná zpráva, server odpoví negativně. V opačném případě odpoví pozitivně a následně pošle seznam dvojic (číslo zprávy, velikost). Za tento příkaz je možné také přidat argument, který specifikuje číslo konkrétní zprávy. V tomto případě odpověď obsahuje informace pouze o této zprávě.

Příklad použití tohoto příkazu může být následující: [16, s. 5–6]

```
C: LIST
S: +OK 2 messages (320 octets)
S: 1 120
S: 2 200

C: LIST 2
S: +OK 2 200

C: LIST 3
S: -ERR no such message, only 2 messages in maildrop
```

Výpis zprávy se následně vyvolá příkazem RETR. Argumentem příkazu je číslo zprávy. Pokud je zpráva nalezena, server odpoví pozitivně a následně zašle text zprávy, který je zakončen znaky <CRLF>.<CRLF>. Pokud některý řádek zprávy začíná tečkou, za níž se nevyskytují znaky <CRLF>, je tečka klientem odstraněna, v opačném případě se jedná o ukončující sekvenci. Tečku na samostatném řádku lze tedy zapsat pomocí dvou teček na samostatném řádku (první z nich bude odstraněna). Příklad vypadá následovně: [16, s. 8]

```
C: RETR 1
S: +OK 120 octets
S: < Text zprávy >
S: .
```

Zprávu ze schránky lze smazat příkazem DELE, jehož argumentem specifikuje číslo zprávy. Server odpoví pozitivně, pokud smazání zprávy bylo úspěšné, v opačném případě odpoví negativně. Příkazem RSET lze smazané zprávy (smazané v tomto sezení) obnovit. Trvalé smazání nastane až po ukončení spojení příkazem QUIT.

Pokud klient nepotřebuje stahovat celý obsah zprávy, ale stačí mu pouze jeho část (hlavička e-mailu), je možné využít příkaz TOP, v jehož argumentech specifikujeme číslo zprávy a počet řádků, o které máme zájem. [16, s. 8–10]

Komplexní příklad komunikace s POP3 serverem vypadá např. následovně: [16, s. 19]

```
S: < čekání na TCP spojení TCP na portu 110 >
C: < navázání spojení >
S:    +OK POP3 server ready <1896.697170952@dbc.mtview.ca.us>
C:    APOP mrose c4c9334bac560ecc979e58001b3e22fb
S:    +OK mrose's maildrop has 2 messages (320 octets)
C:    STAT
S:    +OK 2 320
C:    LIST
S:    +OK 2 messages (320 octets)
S:    1 120
S:    2 200
S:    .
C:    RETR 1
S:    +OK 120 octets
S:    < Text zprávy 1 >
S:    .
```

```
C:    DELE 1
S:    +OK message 1 deleted
C:    RETR 2
S:    +OK 200 octets
S:    < Text zprávy 2 >
S:    .
C:    DELE 2
S:    +OK message 2 deleted
C:    QUIT
S:    +OK dewey POP3 server signing off (maildrop empty)
C:  < uzavření spojení >
S:  < čekání na další spojení >
```

### 2.4.3 Detekce protokolu

Detekce komunikace aplikačním protokolem POP3 je velmi obdobná detekci protokolu SMTP. Z předchozího textu vyplývá, že lze detekci protokolu POP3 provádět zkoumáním přítomnosti POP3 příkazů (především `USER` a `APOP`) a položek hlavičky e-mailu (např. *From* a *To*). Pro účely této práce je uvažován následující algoritmu detekce provozu POP3, který přihlíží na případné rozdělení problému mezi hardwarovou a softwarovou část:

1. V síťovém provozu je vyhledávána e-mailová adresa nebo identifikátor uživatele.
2. Pokud je v síťovém provozu e-mailová adresa nebo identifikátor uživatele nalezen, je odpovídající síťový tok předán pro další zpracování, protože se jedná o potenciálně zájmový provoz.
3. Protože mohla být adresa nebo identifikátor nalezen v kontextu jiného protokolu (nejedná se o POP3), lze dodatečně ověřit, zde se v prvním exportovaném paketu nachází:
   - příkaz `USER` nebo `APOP` včetně identifikátoru uživatele,
   - položka hlavičky e-mailu *From* a *To* včetně e-mailové adresy.
4. Po ukončení vybraného TCP spojení je export tohoto toku zastaven.

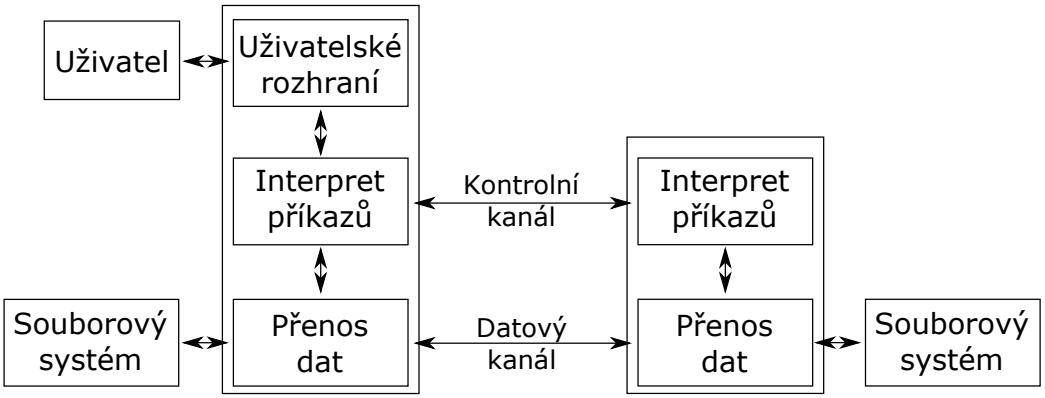
## 2.5 File Transfer Protocol

File Transfer Protocol (FTP) je jeden z nejstarších internetových protokolů vůbec a slouží pro přenos souborů mezi počítači v síti. Pro komunikaci využívá transportní protokol TCP. [14, s. 345]

### 2.5.1 Architektura protokolu

Architektura protokolu je naznačena na obrázku 2.13. Komunikace mezi uživatelem a hostem (serverem) probíhá pomocí kontrolního a datového kanálu (dva různé síťové toky).

FPT klienta lze rozdělit na tří logické části, kterými jsou uživatelské rozhraní, interpret příkazů a část obsluhující přenos dat. Komunikaci s hostem inicializuje uživatelův interpret příkazů na základě podnětu od uživatelského rozhraní. Interpret příkazů uživatele naváže

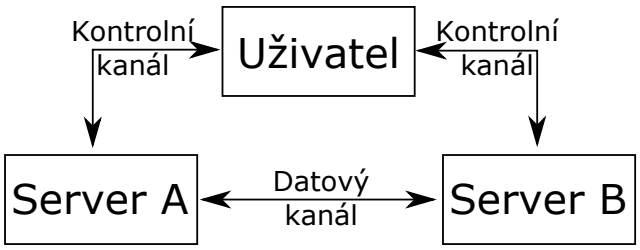



Obrázek 2.13: Architektura protokolu

TCP spojení s interpretem příkazů hosta a vytvoří tak kontrolní kanál. Po tomto kanálu jsou následně zasílány FTP příkazy a odpovědi na ně. [14, s. 345–346]

Parametry datového kanálu jsou domluveny pomocí příkazů na kontrolním kanálu, přenos dat se pak řídí těmito domluvenými parametry. Z pohledu navazování datového kanálu lze rozlišovat tzv. aktivní a pasivní režim komunikace. Při aktivním režimu komunikace protokolu je datový kanál vytvářen hostem (směrem od hosta ke klientovi). Tento způsob navazování datového kanálu může být problematický, protože datový tok může být filtrován firewallem. Další komplikaci může způsobit překlad síťových adres na síťových zařízeních (Network Address Translation). Z tohoto důvodu může být výhodné využít pasivní režim komunikace. V tomto režimu protokolu vytváří uživatel s hostem jak kontrolní, tak i datový kanál. [14, s. 346]

Datový kanál nemusí být nutně navázán mezi klientem a hostem. Pokud klient požaduje přenos dat mezi dvěma hosty, naváže s oběma kontrolní kanál a vyjedná datový kanál mezi hosty (jak je naznačeno na obrázku 2.14). [20, s. 8–9]




Obrázek 2.14: Přenos souborů mezi dvěma hosty [20, s. 9]

### 2.5.2 Formát dat

Při přenosu dat z odesílatele k příjemci je často nutné provádět transformaci dat, protože reprezentace dat na těchto systémech může být odlišná. Například formát NVT-ASCII může mít na různých systémech různou reprezentaci. Odesílací a přijímací strana tedy bude muset provést transformaci mezi standardní reprezentací a jeho interní reprezentací. Další problém může vzniknout přenosem binárních dat mezi systémy s různou délkou slova (např. přenos dat ze systému s 32-bitovými slovy na systém s 64-bitovým slovy a opačně). [20, s. 10]

Reprezentace dat v FTP protokolu se řídí podle uživatele, který specifikuje typ reprezentace. Ten může být buď implicitní (ASCII nebo EBCDIC) nebo explicitní, kdy je definována velikost bytu přenášených dat (tzv. velikost logického bytu). Tato velikost však nemá nic společného s velikostí bytu použitého pro přenos dat po síti. [20, s. 11]

Každá implementace FTP musí povinně akceptovat datový typ ASCII. Alternativou pak může být kódování EBCDIC, které může být výhodné použít, pokud oba koncové systémy používají tento typ reprezentace dat. Pro přenos dat odesílatel tedy převede data z vnitřní reprezentace do smluvené (např. do standardní 8-bitové NVT-ASCII reprezentace), příjemce následně provádí opačný proces, kdy převádí data do vlastní reprezentace dat. [20, s. 11–12]

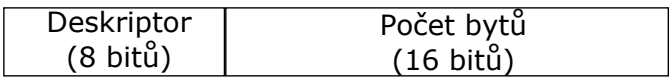
Dalším typem, který je možné přenášet je tzv. IMAGE typ. Pomocí tohoto typu je možné přenášet především binární data. Data jsou odesílána jako kontinuální proud bitů, které jsou seskupovány do 8-bitových bytů. Protože binární data nemusí být zarovnána na byty (velikost logického bytu může být odlišná), součástí posledního přeneseného bytu může být zarovnání (nulové bity). [20, s. 12]

Jak již bylo zmíněno, pro přenos lze definovat vlastní velikost bytu (logický byt). Jednotlivé logické byty jsou skládány za sebe. Protože jsou data opět přenášena po bytech (osm bitů), může se v posledním přenášeném bytu objevit zarovnání. Přijímající strana pak musí doručená data správným způsobem převést na logický byte dohodnuté velikosti. [20, s. 12–13]

FTP protokol kromě typu reprezentace dat umožňuje specifikovat typ struktury přenášeného souboru. FTP rozlišuje následující struktury: [20, s. 14–17]

- **Souborová struktura** – soubor neobsahuje žádnou vnitřní strukturu a je považován za sled bitů.
- **Struktura záznam** – soubor se skládá ze záznamů, které jsou odděleny znakem konce řádku.
- **Stránková struktura** – soubor je tvořen nezávislými stránkami různé velikosti, každá stránka tedy obsahuje index (číslo) stránky, velikost a typ, který může nabývat následujících hodnot:
  - 0 = poslední stránka – označení konce stránkové struktury,
  - 1 = obyčejná stránka – normální typ stránky,
  - 2 = popisovač stránky – používá se pro přenos informací popisujících,
  - 3 = stránka řízení přístupu – slouží pro řízení přístupu na úrovní stránek.

Dalším parametrem přenosu dat protokolem FTP je mód přenosu. Uživatel má na výběr jeden ze tří módů. Prvním z nich je tzv. streamový přenos, kdy jsou data přenášena jako proud bitů. Konec souboru je indikován uzavřením datového kanálu odesílatelem. Další variantou je blokový mód přenosu, kdy jsou data přenášená po blocích. Každý blok se skládá z hlavičky a dat. Hlavička obsahuje deskriptor a velikost dat v bytech, které blok obsahuje (viz obrázek 2.15). [20, s. 20–22]

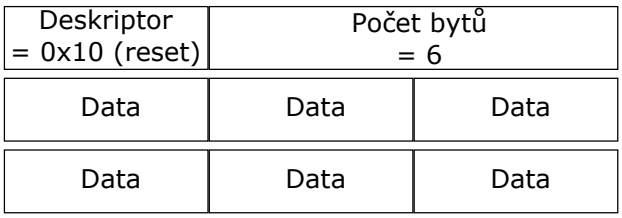



Obrázek 2.15: Tvar hlavičky bloku [20, s. 22]

Hodnota deskriptoru indikuje následující příznaky: [20, s. 22]

- poslední blok souboru (bit 7),
- poslední blok záznamu (bit 6),
- podezření na chybu v bloku dat (bit 5),
- značku restartu (bit 4).

Příklad struktury přenášeného bloku obsahující šest bytů je naznačen na obrázku. 2.16.

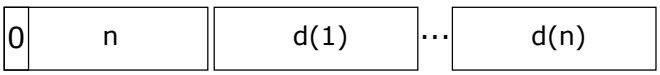



Obrázek 2.16: Příklad blokového módu [20, s. 22]

Poslední módem přenosu je kompresní mód. V tomto módu existují tři druhy přenášených dat: [20, s. 23]

- nekomprimovaná data,
- komprimovaná data (přenos několika stejných bytů),
- řídicí informace.

První byte přenosu obsahuje indikaci, o jaký druh dat se jedná. Pokud je první bit roven nule, jedná se o přenos nekomprimovaných dat. Zbylé bity prvního bytu určují, kolik bytů dat (nekomprimovaných) bude následovat za tímto bytem (viz obrázek 2.17). [20, s. 23]




Obrázek 2.17: Tvar dat bez komprese [20, s. 23]

Pokud první byte na prvních bitech obsahuje *10*, jedná se přenos dat komprimovaných pomocí kódování délek sledů. Následující bity určují počet opakování bytu, který následuje za tímto bytem (viz obrázek 2.18).


Obrázek 2.18: Tvar dat s kompresí [20, s. 23]

Řídicí informace jsou indikovány dvojicí bitů *11* prvního bytu dat. Další bity následně indikují deskriptor kódu (konec záznamu, konec souboru atd.).

### 2.5.3 Příkazy protokolu

Příkazy, které jsou uživatelem zasílány hostovi prostřednictvím kontrolního kanálu, je možné rozdělit do několika kategorií podle jejich účelu.

První množinou jsou příkazy pro řízení přístupu. Těmi základními jsou: [20, s. 25–27]

- USER – argumentem příkazu je identifikován uživatel, k jehož souborovému systému budeme přistupovat.
- PASS – slouží pro specifikaci hesla uživatele a dokončení identifikaci uživatele.
- CWD – slouží pro změnu pracovního adresáře.
- QUIT – slouží pro odhlášení uživatele.

Další skupinou jsou příkazy určené pro vyjednání parametrů přenosu dat. Příkladem této skupiny jsou: [20, s. 27–28]

- PORT – příkazem se specifikuje IP adresa a port uživatele, který využije host při vytvoření datového kanálů (aktivní režim komunikace).
- PASV – stanovuje, že komunikace bude probíhat v pasivním režimu.
- TYPE – příkazem specifikujeme reprezentaci dat popsanou výše.
- STRU – určuje typ struktury souboru (popsáno výše).
- MODE – specifikuje mód přenosu dat (viz. výše).

Poslední skupinou jsou příkazy pro práci se soubory. Vybranými příkazy jsou: [20, s. 28–33]

- RETR – příkaz slouží pro přenos souboru specifikovaného v argumentu z hosta k uživateli nebo k hostovi, který se nachází na druhém konci datového kanálu.
- STOR – příkaz slouží pro přijetí souboru hostem, v argumentu je uveden cíl přijatého souboru (existující soubor je přepsán).
- STOU – jako příkaz STOR, ale pokud soubor existuje, je generována chyba.
- APPE – podobné příkazu STOR, pokud soubor existuje, přijatá data budou k tomuto souboru připojena.
- RNFR, RNTO – příkazy je možné přejmenovat soubor kdy první z nich definuje soubor, který má být přejmenován a druhý specifikuje nové jméno souboru.
- DELE – příkaz slouží pro vymazání soubor specifikovaného v argumentu příkazu.
- RMD – slouží pro mazání adresáře.
- MKD – vytvoří nový adresář požadovaného jména.
- PWD – vypíše cestu k pracovnímu adresáři.
- LIST – příkaz vypíše obsah adresáře.

### 2.5.4 Odpovědi protokolu

Na každý příkaz uživatel obdrží odpověď, která se skládá z tříciferného čísla a dodatečné informace. První číslice udává, zda se jedná o pozitivní či negativní odpověď a může nabývat následujících hodnot: [20, s. 37]

- 1yz – předběžná pozitivní odpověď,
- 2yz – konečná pozitivní odpověď,
- 3yz – pozitivní odpověď na příkazy, které očekávají, že budou následovat další příkazy (např. příkaz RNFR, protože očekává příkaz RNTO),
- 4yz – přechodná negativní odpověď (akce může být zopakována, např. zadané chybné heslo),
- 5yz – trvale negativní odpověď (příkaz nebo kombinace příkazů není platná).

Druhá číslice určuje okruh, kterých se odpověď týká: [20, s. 38]

- x0z – syntaxe,
- x1z – informace,
- x2z – připojení,
- x3z – autentizace,
- x4z – nespecifikováno,
- x5z – souborový systém.

Třetí číslice následně specifikuje konkrétní odpověď z množiny určené předchozími číslicemi.

### 2.5.5 Detekce protokolu

Z předchozího textu vyplývá, že detekce protokolu FTP je oproti předchozím protokolům složitější. Hledáním identifikátoru uživatele v komunikaci jsme schopni detekovat pouze síťový tok, který obsahuje kontrolní kanál. Síťový toku datového kanálu je získán analýzou kontrolního kanálu. Detekce kontrolního kanálu protokolu FTP lze provádět na základe přítomnosti kombinace konkrétního FTP příkazů (`USER`) a identifikátoru uživatele. Po detekci datového kanálu protokolu FTP lze z konkrétního FTP příkazu (`PORT`) identifikovat sítový tok datového kanálu.

V této práci je uvažován následující algoritmus detekce protokolu FTP, který přihlíží na potenciální rozdělení úlohy mezi hardwarovou a softwarovou část:

1. V síťovém provozu je nejdříve vyhledáván identifikátor uživatele.
2. Pokud je identifikátor nalezen, daný síťový tok je předán k další zpracování.
3. Protože mohl být identifikátor nalezen v nesprávném kontextu (nejedná se o protokol FTP), lze dodatečně ověřit, zda se v prvním exportovaném paketu nachází příkaz `USER` s identifikátorem uživatele.

4. V následujících paketech je vyhledáván FTP příkaz `PORT`, který identifikuje síťový tok datového kanálu.
5. Po nalezení příkazu je zahájen export síťového toku datového kanálu specifikovaného argumentem příkazu.
6. Po ukončení TCP spojení datového kanálu je ukončen export odpovídajícího síťového toku.
7. Dále se pokračuje buď bodem 4 nebo 8.
8. Po ukončení TCP spojení kontrolního kanálu je ukončen export daného síťového toku.

## 2.6 Session Initiation Protocol

Session Initiation Protocol (SIP) [25] je signalizační protokol pro řízení multimediálních relací v IP sítích. Protokol definuje zprávy pro sestavení, ukončení a udržování telefonního spojení v IP telefonie. [28, s. 37]

### 2.6.1 Architektura protokolu

SIP je protokol pracující na aplikační vrstvě. Jde o textový protokol, který je svou strukturou velmi podobný např. již zmíněnému protokolu SMTP nebo protokolu HTTP (HyperText Transport Protocol). [7]

Jelikož SIP slouží pouze pro přenos signalizace, pro komunikaci dvou zařízení je nutné vytvořit další spojení pro přenos dat. Spojení a parametry spojení jsou vyjednány pomocí signalizace (podobně jako u protokolu FTP). Data jsou nejčastěji přenášena protokolem RTP (Real-time Transport Protocol), který slouží pro přenos multimediálních dat v reálném čase. Dalším důležitým protokolem je protokol SDP (Session Description Protocol), který slouží pro popis vlastností účastníků spojení. [28, s. 38]

Adresace v síti SIP je prováděna použitím SIP URI (Iniform Resource Identifier). SIP URI má následující formát:

```
sip:user@host:port
```

Skládá se z označení protokolu SIP, identifikace uživatele (user) a specifikace domény (host) Za dvojtečkou je potom možné uvést nestandardní číslo portu a další parametry. Doménová část URI je adresována pomocí systému pro překlad doménových jmen DNS (Domain Name System). [28, s. 39]

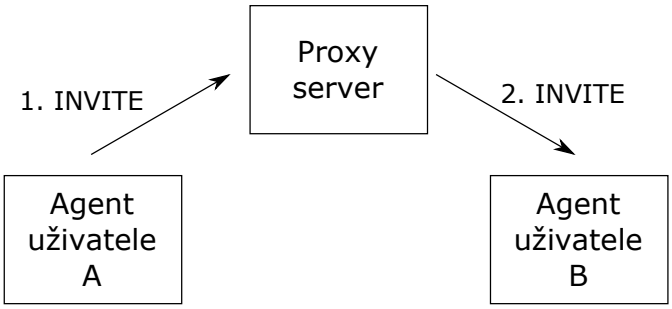
Síť SIP obsahuje následující prvky:

- Uživatelský agent,
- Směrovací server,
- Proxy server,
- Lokalizační server,
- Registrační server.

Uživatelskými agenty (UA) jsou nazývány koncové uzly sítě SIP. UA můžou být SIP telefony, mobilní telefony (podporující SIP), softwarové aplikace nebo brány pro připojení do klasické veřejné telefonní sítě PSTN (Public Switched Telephone Network). Každý UA se skládá z: [28, s. 39]

- UA serveru (UAS) – přijímá požadavky, na které odesílá odpovědi,
- UA klienta (UAC) – zasílá požadavky a přijímá odpovědi na ně.

Další prvkem, který může být součástí SIP sítě, jsou SIP proxy servery. Ty přijímají požadavky okolních prvků a směrují je do požadovaného cíle (viz. obrázek 2.19).




Obrázek 2.19: Proxy server a přesměrování hovoru [25, s. 12]

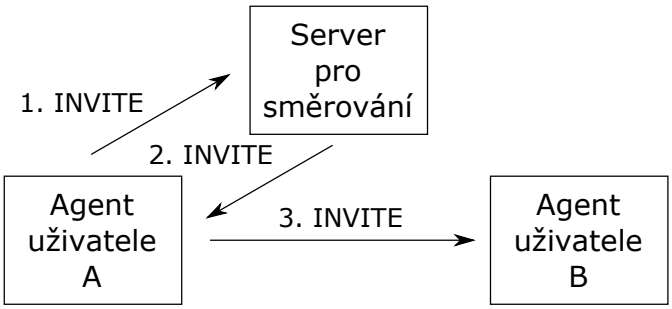
Lze rozlišovat dva typy SIP proxy serverů:

- bezstavový proxy server – pouze přeposílají požadavky,
- stavový proxy server – udržují si informaci o stavu.

Stavový proxy server je komplexnější než bezstavový. Udržuje si stav spojení po celou dobu své existence (což může limitovat výkon), umožňuje provádět komplikovanější metody nalezení uživatele (volání na alternativní zařízení v případě nedostupnosti atd.), účtování a další. [28, s. 41–42]

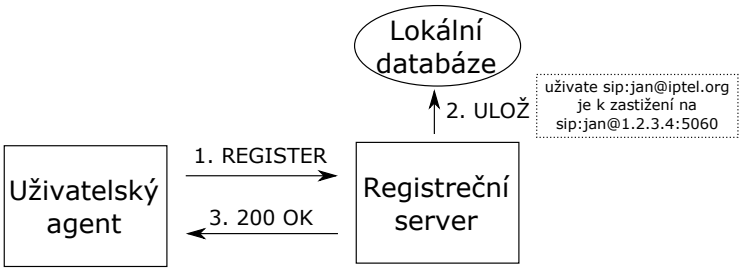
Lokalizační servery slouží jako zdroj informací o možném umístění volaného uživatele pro směrovací a proxy servery. [26]

Směrovací server se uplatní v situacích, kdy není známa IP adresa volaného. Server pomocí lokalizačních serverů nalezne žádanou adresu a předá ji zpět UA. Po obdržení adresy UA naváže spojení přímo s UA volaného (viz obr. 2.20). [26]




Obrázek 2.20: Směrovací server a přesměrování hovoru [28, s. 44]

Registrační server slouží pro registraci uživatele (jeho identifikaci a autentizaci). Registrace je prováděna žádostí REGISTER, která obsahuje adresu záznamu v lokální databázi (např. *sip:jan@iptel.org*) a kontaktní adresu (např. *sip:jan@1.2.3.4:5060*, kde 1.2.3.4 je IP adresa a 5060 port, na kterém je uživatel dostupný). Pokud registrace proběhla v pořádku, server zasílá potvrzující odpověď a do lokalizační tabulky si uloží informace o uživateli (IP adresa a port). Na obrázku 2.21 je znázorněn typický proces registrace. [28, s. 43]




Obrázek 2.21: Registrační server [28, s. 44]

### 2.6.2 Žádosti a odpovědi protokolu

Komunikace signalizace SIP je tvořena zprávami. Každá zpráva je přenášena v samostatném UDP datagramu a skládá se z hlavičky zprávy a obsahu. První řádek zprávy vždy identifikuje typ zprávy (žádosti nebo odpovědi). Žádostmi jsou: [28, s. 47]

- INVITE – žádost o inicializaci spojení nebo o změnu parametrů existujícího spojení,
- ACK – zpráva potvrzuje přijetí odpovědi OK na žádost INVITE (trojcestné potvrzení),
- BYE – žádost o ukončení spojení jednou ze stran,
- CANCEL – slouží pro zrušení sestavovaní spojení žádostí INVITE, pokud volaný ještě nepotvrdil žádost,
- REGISTER – slouží pro registraci uživatele, smyslem je oznámit aktuální polohu uživatele (IP adresa a port),
- OPTIONS – žádost o zaslání vlastností UAS.

Odpovědi na první řádku obsahuje verzi protokolu, kód odpovědi a zprávu. Kód odpovědi je celé číslo v rozmezí 100-699, první číslice udává typ odpovědi: [28, s. 49–50]

- 1xx – informační, výsledek žádosti není prozatím známí,
- 2xx – pozitivní finální odpověď,
- 3xx – odpovědi spojené s přesměrováním,
- 4xx – negativní konečné odpovědi,
- 5xx – indikují problém na straně serveru,
- 6xx – žádost nemůže být splněna na žádném serveru (globální chyba).

Tělo zprávy je dále tvořeno dvojicemi názvů polí a jejich hodnot níže uvedeného tvaru: [7]

```
<název> : <hodnota> <CRLF>
```

SIP žádost obsahuje jedno nebo více polí *Via*, která jsou používaná pro záznam cesty žádosti a ke směrování odpovědi. Pole *From* a *To* identifikuje iniciátora a příjemce hovoru. Pro rozeznání, k jakému hovoru žádost náleží, se využívá pole *Call-ID*, pro zajištění správného pořadí vykonání požadavků je využito sekvenční číslo *CSeq*. Pole *Contact* obsahuje IP adresu a port, na kterém je odesilatel dostupný (a kde očekává žádosti). Velikost a obsah dat následujících za hlavičkou jsou popsány položkami *Content-Length* a *Content-Type*. [28, s. 46]

Tělo zprávy je od hlavičky odděleno prázdným řádkem. V těle se nejčastěji objevuje SDP protokol, kterým odesílatel určuje protistraně, jaké parametry datového spojení upřednostňuje. [28, s. 37]

Položky protokolu SDP jsou tvaru *x=hodnota*, kde x zastupuje písmeno parametru. Důležitými parametry jsou především: [12, s. 8–9]

- v – verze protokolu,
- o – zakladatel a identifikace sezení,
- s – název sezení,
- m – název média (video, audio) a a transportní adresa,
- t – doba trvání sezení,
- c – informace o spojení,
- a – atributy medií.

Žádost tedy může vypadat následovně: [28, s. 51]

```
INVITE sip:UserB@there.com SIP/2.0
Via : SIP/2.0/UDP here.com:5060
From : AG <sip:UserA@here.com>;tag=123
To : BG <sip:UserB@there.com>
Call-ID : 12345600@here.com
Cseq : 1 INVITE
Content : AG <sip:UserA@here.com>
Content-Type : application/sdp
Content-Length : 147

V=0
O=UserA 28908 28908 IN IP4 here.com
S=Session SDP
C=IN IP4 100.101.102.103
T=0 0
M=audio 49 172 RTP/AVP 0
A=rtpmap:0 PCMU/8000
```

Na tomto příkladu zle vidět žádost o inicializaci spojení uživatele identifikovaného kontaktní adresou *sip:UserA@here.com* s uživatelem identifikovaným kontaktní adresou *sip:UserB@there.com*. V těle zprávy se pak nachází protokol SDP, kterým se domlouvá jeden RTP kanál pro přenos audia (směr od volaného). Odpověď na tuto žádost může vypadat například následovně: [28, s. 51]

```
SIP/2.0 200 OK
Via : SIP/2.0/UDP here.com:5060
From : AG <sip:UserA@here.com>;tag=123
To : BG <sip:UserB@there.com>
Call-ID : 12345600@here.com
Cseq : 1 INVITE
Content : BG <sip:UserB@here.com>
Content-Type : application/sdp
Content-Length : 134

V=0
O=UserB 28908 28908 IN IP4 here.com
S=Session SDP
C=IN IP4 110.111.112.113
T=0 0
M=audio 3456 172 RTP/AVP 0
A=rtpmap:0 PCMU/8000
```

Z příkladu lze vidět, že odpověď je pozitivní, spojení se podařilo tedy v pořádku inicializovat. V těle zprávy se nachází protokol SDP, kterým se domluví RTP kanál přenášející audio (směr k volanému).

### 2.6.3 Detekce protokolu

Detekce protokolu SIP je podobná detekci protokolu FTP. Vyhledáním identifikátoru uživatele (SIP URI) jsme schopni nalézt pouze síťový tok obsahující signalizaci. Tento tok lze dle předchozího textu detekovat přítomností konkrétního SIP požadavku (INVITE) a identifikátoru uživatele. Tento tok je dále analyzován, protože požadavek INVITE a odpověď na něj obsahují popis vytvářeného RTP spojení.

Pro účely této práce je dále uvažován následující algoritmus detekce SIP komunikace, který přihlíží k potenciálnímu rozdělení úlohy na hardwarovou a softwarovou část:

1. V provozu je vyhledáván identifikátor uživatele.
2. Po nalezení identifikátoru je odpovídající síťový tok předán pro další zpracování.
3. Protože mohl být identifikátor nalezen v nevyhovujícím kontextu (nejedná se o SIP), lze dodatečně ověřit, zda je v prvním exportovaném paketu přítomen SIP požadavek `INVITE`, který v těle obsahuje obsahuje položku `From` nebo `To` a identifikátor uživatele.
4. Z těla požadavku `INVITE` je získána identifikace vytvářeného RTP spojení.
5. Je zahájen export síťového toku odpovídajícího RTP spojení.
6. Po ukončení TCP spojení je ukončen export obou síťových toků.

# Kapitola 3

# Architektura mikrosondy

Mikrosonda je vestavěný systém vytvořený v rámci projektu pro zákonné odposlechy *Moderní prostředky pro boj s kybernetickou kriminalitou na Internetu nové generace* [4]. Slouží pro monitorování komunikace na síťových linkách o rychlosti 1 Gb/s a je navržena na hardwarové platformě NetModule ZE7000[1].

## 3.1 Stávající architektura

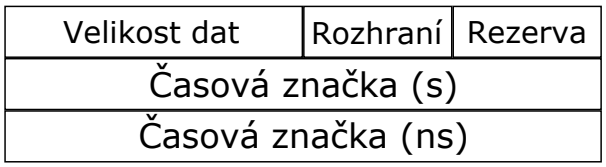
Architektura Mikrosondy byla navržena v bakalářské práci *Mikrosonda na platformě Xilinx Zynq* [11], na kterou tato práce navazuje.

Mikrosonda je vytvořena na čipu Xilinx Zynq[2], který kromě programovatelného pole FPGA (Field Programmable Gate Array) obsahuje především plnohodnotný dvoujádrový procesor ARM Cortex-A9. Aplikaci Mikrosonda lze rozdělit na FPGA firmware a software pro procesor ARM. [17, s. 2]

### 3.1.1 FPGA firmware

V FPGA je implementována funkcionalita, která se zabývá klasifikací a filtrováním paketů. Lze ji rozdělit na vstupní, procesní a výstupní část.

Úkolem vstupní části je přijímat pakety ze sítě a označit je přesnou časovou značkou. Protože je možné přijímat pakety z více vstupních rozhraní, je do paketu vložena informace o tom, ze kterého rozhraní byl paket přijat, dále je do paketu vložena informace o celkové velikost přijatých dat. Informace jsou vloženy do hlavičky, která je předřazena přijatému paketu (viz. obrázek 3.1). Všechny takto rozšířené pakety jsou následně sloučeny v jeden datový tok a předány procesní části. [17, s. 4]



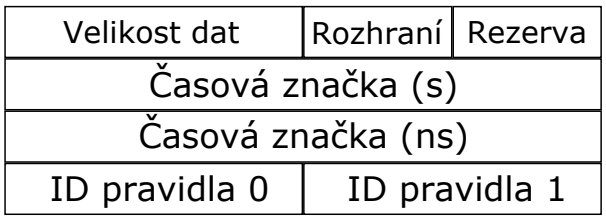

Obrázek 3.1: Tvar hlavičky přidané vstupní částí Mikrosondy

[1] http://www.netmodule.com/technologies/platforms/zynq-platform.html
[2] http://www.xilinx.com/products/silicon-devices/soc/zynq-7000.html

Procesní část se skládá z klasifikační jednotky a filtru. Klasifikační část provádí analýzu paketu a filtru předává potřebné parametry (zdrojovou a cílovou IP adresu). Filtr na základě tabulky pravidel a jejího obsahu provádí filtrování paketů. Výstupem procesní části jsou pakety a identifikátory pravidel, na základě kterých proběhlo propuštění paketu. [17, s. 4]

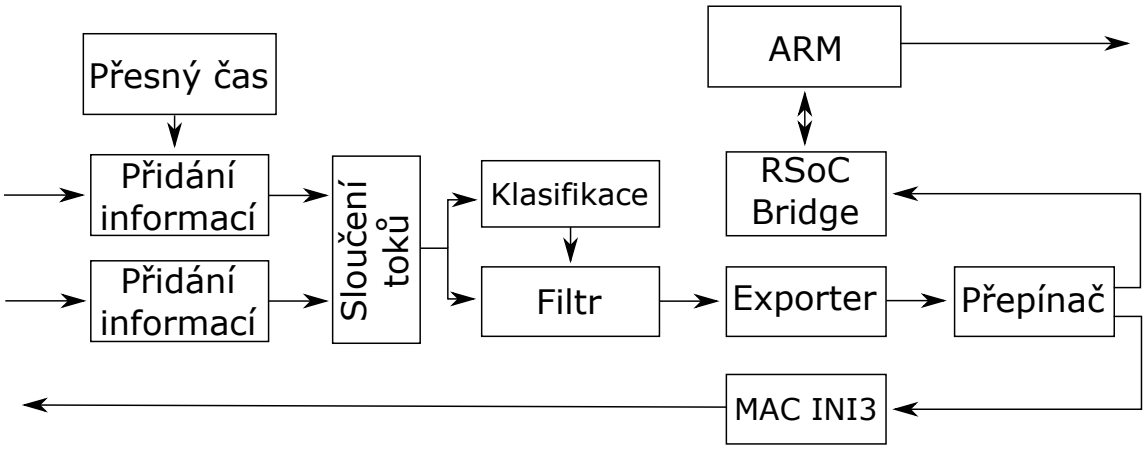
Úkolem výstupní části je kompletace hlavičky protokolu Internal Network Interface 3 (INI3), která je předřazena před každým přijatým paketem. Její tvar je uveden na obrázku 3.2. Paket s hlavičkou INI3 je následně výstupní částí odeslán sítí na určené místo. Aktuální implementace výstupní část byla oproti uvedené bakalářské práci upravena.

Původní exportér dat, který umožňoval volbu způsobu exportu (protokoly IPv4 či IPv6 a UDP) pouze před spuštěním syntézy obvodu, byl nahrazen variantou, která umožňuje volbu protokolu či odpojení exportéru za běhu. Do firmwaru byl přidán konfigurovatelný přepínač, který umožňuje volbu portu, na který se bude exportovat. Přepínač ve firmwaru umožňuje volbu mezi exportem do softwaru, kde mohou být data dále zpracovávána, nebo odesláním paketu pomocí hardwaru do sítě. Jelikož přidané hlavičky vnáší do paketu nadbytečná data, není možné exportovat přijatá data na plné rychlosti rozhraní (hlavičky vnáší nadbytečnou režii a ubírají šířku pásma). Tento problém řeší další komponenta kompromisem, který spočívá v nahrazení INI3 hlavičky a MAC adres původního paketu (pro odposlech jsou nezajímavé) modifikovanou INI3 hlavičkou (MAC_INI3). V tomto případě je však již nutné sběrný bod připojit přímo k sondě. [18, s. 3–5]




Obrázek 3.2: Tvar INI3 hlavičky [18, s. 3]

Pro export do softwaru a komunikaci s komponentami se používá RSoC Bridge, který je součástí RSoC Frameworku. Celý obsah firmwaru je naznačen na obrázku 3.3.

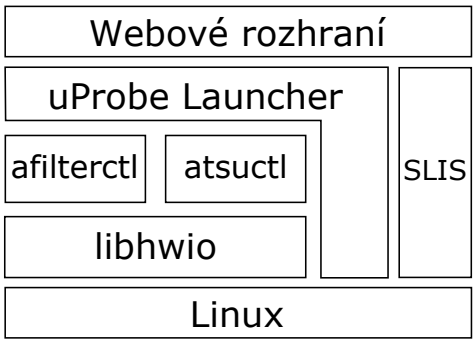



Obrázek 3.3: Firmware Mikrosondy [17, s. 4]

Z výše uvedeného je tedy vidět, že Mikrosonda provádí klasifikaci a filtrování pouze na transportní vrstvě. Klasifikace konkrétního aplikačního protokolu by musela být prováděna v softwaru. Výkon procesoru ARM však není dostačující pro klasifikaci celého objemu příchozích dat. Řešením je akcelerace části funkcionality klasifikace v hardwaru. V hardwaru je prováděno předzpracování síťového provozu, které spočívá ve výběru potenciálně zajímavých síťových toků. Software následně provádí klasifikaci na mnohem menším objemu dat.

### 3.1.2 Software

Softwarové vybavení Mikrosondy slouží především pro konfiguraci hardwaru (nastavování filtračních pravidel, typ exportu atd.) a pro softwarový export dat protokolem TCP. Skládá se z mnoha softwarových vrstev, které jsou naznačeny na obrázku 3.4.




Obrázek 3.4: Softwarové vrstvy Mikrosondy [17, s. 5]

Vrstvou, která je nejblíže hardwaru, je operační systém Linux s verzí jádra 3.9.0 obsahující úpravy od společnosti Xilinx. Součástí operačního systému je ovladač RSoC Framework, který zprostředkovává komunikaci s komponentou RSoC Bridge, respektive s komponentami nacházející se v hardwaru. [17, s. 5]

Pro přístup ke komponentám, které se nachází ve firmwaru, je používána knihovna HWIO. Vytváří abstraktní vrstvu pro jednotný přístup k hardwaru nezávisle na softwarovému prostředí. Nachází se v ní také informace o komponentách umístěných v hardwaru. [17, s. 5]

Nad touto knihovnou jsou následně programovány uživatelské aplikace pracující s hardwarem v FPGA. Příkladem jsou naznačené aplikace afilterctl (sloužicí pro konfiguraci filtru), atsuctl (pro správu jednotky přesného času) a CCCID (Content of Communication Control Interface Daemon). Aplikace CCCID řídí filtraci (vkládá a odebírá pravidla z filtru) na základě požadavků mediačního zařízení (zařízení obsahující funkce pro vzdálenou konfiguraci a příjem zájmového provozu od Mikrosondy), kterým v současné době může být přímo i Mikrosonda. [17, s. 5]

Výše uvedené nástroje využívá aplikace uProbe Launcher, která na základě konfigurace, která se nachází na SD kartě, provede celkové nastavení Mikrosondy. [17, s. 5]

Další součástí softwarového vybavení je systém pro zákonné odposlechy (SLIS), který slouží pro správu odposlechů. SLIS přijímá požadavky na odposlechy z uživatelského rozhraní, na základě kterých spouští či ukončuje odposlech konfigurací hardwaru v předem definovaný čas. [17, s. 5–6]

Na nejvyšší softwarové vrstvě se nachází webové konfigurační rozhraní, pomoci kterého lze konfigurovat sondu, ukládat konfiguraci na SD kartu a spravovat odposlechy.

## 3.2 Limity ZE7000 a Zynq

Pracovní frekvence procesoru ARM Cortex-A9 MPCore, který se nachází na čipu Xilinx Zynq, je až 1 GHz. Výkonnost jednoho jádra při této frekvenci je přibližně 2,5 DMIPS. Na modulu Enclustra Mars ZX3 se však nachází čip Xilinx Zynq Z-7020 (speed grade -1), u kterého je maximální pracovní frekvence 667 MHz. Výkonost procesoru ARM tohoto konkrétního čipu je pak přibližně 1,67 DMIPS na jedno jádro. [10, s. 1] [6]

Limitujícím faktorem je také rychlost přenosu dat z PL do PS. Pro tento přenos je využit RSoC Framework, který v aktuální konfiguraci umožňuje přenášet data rychlostí až 165 MB/s (1320 Mb/s). [23]

Pro export dat z procesoru je využíváno Ethernetové rozhraní, které je připojeno přímo k procesoru ARM. Toto rozhraní je schopno exportovat data pouze rychlostí až 500 Mb/s. Rychlost přenosu reálného provozu je pak pouze asi 300 Mb/s. [18, s. 3] [9]

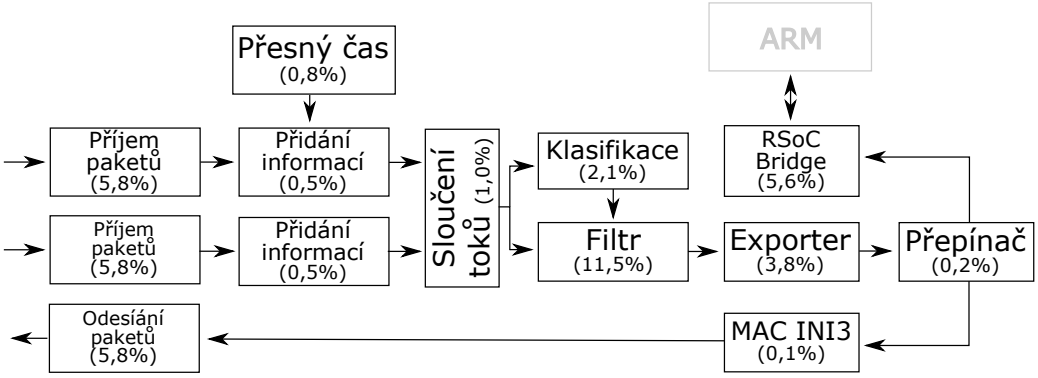
V tabulce 3.1 jsou uvedeny zdroje programovatelného hradlového pole čipu Xilinx Zynq a spotřebované zdroje stávající implementací firmwaru Mikrosondy.

| | Celkem | Využito | Využito [%] |
|---|---|---|---|
| Look-Up (LUT) | 53200 | 28347 | 53,3 |
| Flip-Flop (FF) | 106400 | 34901 | 32,8 |
| Block RAM 36 Kb (BRAM) | 140 | 78 | 55,7 |
| DSP | 220 | 2 | 0,9 |

Tabulka 3.1: Celkový počet zdrojů FPGA Zynq a spotřebované zdroje

Z tabulky je vidět, že pro implementaci stávajícího řešení je využita přibližně polovina dostupných zdrojů. Na obrázku 3.5 je znázorněno přibližné využití LUT hlavními komponentami. Implementaci hardwarového řešení monitorování aplikačních protokolů by vyžadovala do firmwaru vložit komponentu podobnou jednotce Pattern Match, která bude popsána v kapitole 5.1.3. Z tabulky 6.3 je patrné, že přestože komponenta umožňuje vyhledávat vzory velmi omezené délky, pro implementaci komponenty je využito značné množství zdrojů. Pro monitorování aplikačních protokolů by však délka vyhledávaného vzoru byla mnohonásobně větší. Počet nutných zdrojů by tak překročil dostupnou kapacitu FPGA. [30, s. 3]

Z těchto důvodů nebude čistě softwarová (hardwarová) varianta realizovatelná a funkcionalita monitorování aplikačních protokolů bude muset být rozdělena mezi hardware a software.




Obrázek 3.5: využití zdrojů jednotlivými komponentami

# Kapitola 4

# Software Defined Monitoring

Monitorování síťového provozu je jedním z klíčových úkolů v oblasti moderního inženýrství a bezpečnosti. S neustále se zvětšující rychlostí síťových linek (až 100 Gb/s) se zvyšují nároky na výkon monitorovacích zařízení. Dalším problémem je neustále větší množství síťových protokolů, které se na síti pochybuje a které je třeba monitorovat.

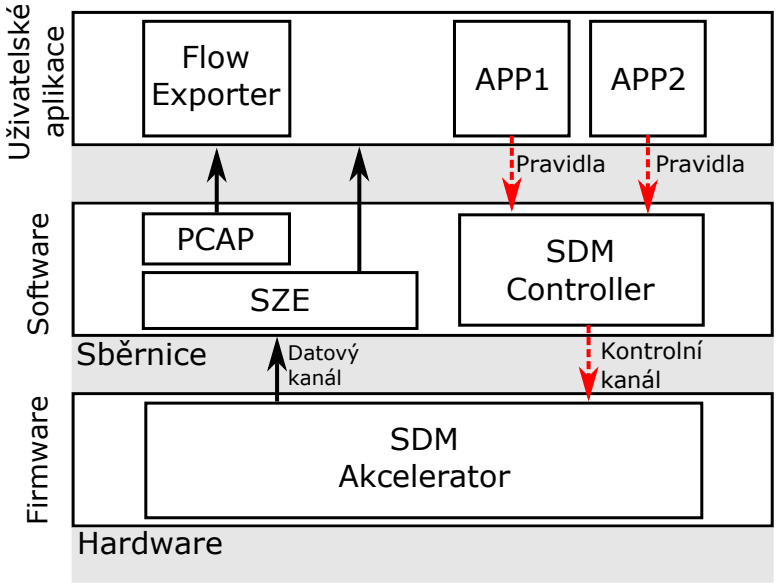
Čistě softwarová implementace monitorování síťových toků na aplikační úrovni může být omezena výkonností dostupných procesorů. Oproti tomu čistě hardwarová implementace není příliš flexibilní, realizace protokolového analyzátoru v jazycích pro popis hardwaru je obtížná. [15, s. 1]

## 4.1 Architektura SDM

Myšlenkou konceptu Software Defined Monitoring (SDM) je fakt, že síťový provoz obsahuje jen zlomek zájmové komunikace. SDM proto rozděluje proces monitorování na akcelerovanou část, která se nachází v hardwaru, a softwarový kontrolér s monitorovacími pluginy. Úkolem hardwarového akcelerátoru je snížení zátěže pro tyto softwarové pluginy. To je prováděno tak, že do softwaru jsou předány pouze vybrané síťové toky. Výběr těchto toků provádí jednotlivé monitorovací pluginy přes jednotné konfigurační rozhraní hardwarového akcelerátoru.

Základní myšlenkou akcelerace u SDM je regulace síťového provozu a jeho distribuce do softwaru, která je realizovaná hardwarovým předzpracováním paketů. Předzpracování je potom plně řízeno softwarovými aplikacemi. [15, s. 1–2]

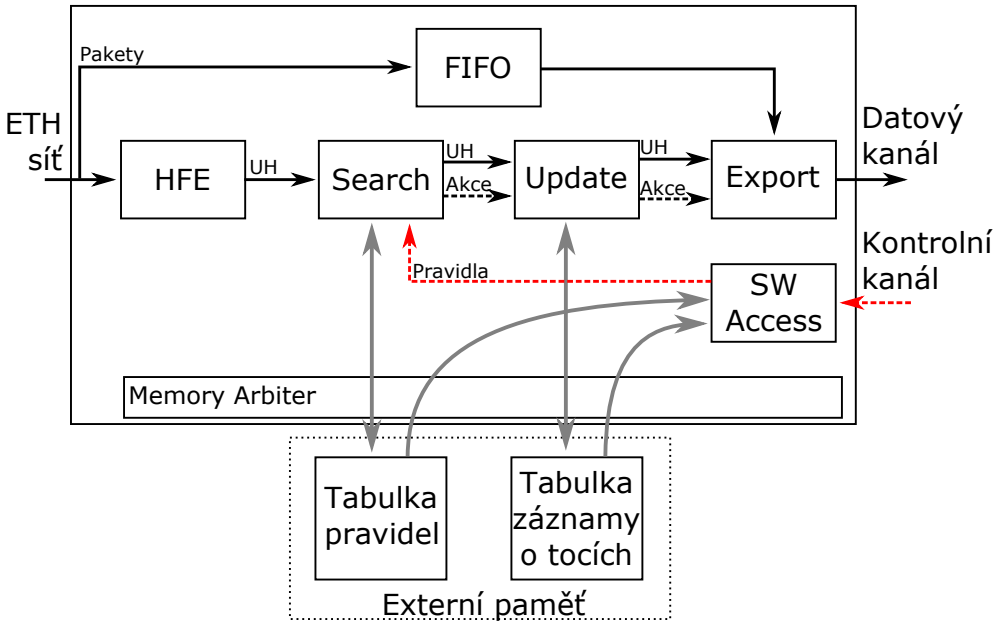
Schéma systému SDM je naznačeno na obrázku 4.1. Datové cesty jsou naznačeny plnými šipkami, kontrolní jsou naznačeny šipkami přerušovanými. Systém se skládá ze dvou hlavních částí: firmware a software. Tyto dvě části jsou následně propojeny sběrnicí (např. PCI Express). FPGA firmware obsahuje část SDM, která je schopna zpracovávat příchozí provoz na plné rychlosti linky (podrobněji popsáno v následující kapitole). Softwarová vrstva obsahuje prostředky pro konfiguraci SDM firmwaru a přenos síťových dat. Data z firmwaru mohou být přijata ve standartním PCAP formátu nebo proprietárním formátu SZE. Na nejvyšší úrovni systému SDM se nachází specifické uživatelské aplikace. [15, s. 2–3]




Obrázek 4.1: Architektura konceptu SDM [15, s. 3]

## 4.2 Firmware

Schéma implementace firmawaru SDM je naznačeno na obrázku 4.2. Zpracování každého paketu ve firmwaru začíná analýzou a extrakcí zajímavých metadat z hlaviček paketů (blok HFE – Header Field Extractor). Získaná metadata v podobě hlavičky Unified Header (UH) jsou následně použita pro klasifikaci paketu založenou na softwarově definované množině pravidel. Během procesu klasifikace se pracuje pouze s metadaty, paket je mezitím uložen ve vyrovnávací paměti FIFO. [15, s. 3]




Obrázek 4.2: Firmwarová část SDM [15, s. 3]

Pro klasifikaci (v bloku Search) se používá identifikátor toku (pětice zdrojová a sílová IP adresa, zdrojový a cílový TCP/UDP port a číslo protokolu), který se využívá pro hledání v množině pravidel. Pravidla obsahují identifikátor toku a akci, která bude s paketem provedena.

Další blok (Update) slouží pro správu záznamů toků. Ukládá agregovaná data (např. NetFlow) o každém toku, která jsou následně pravidelně exportována do softwaru kontrolním kanálem.

Klasifikační blok pro každý příchozí paket vybere akci. Blok Export sloučí metadata s odpovídajícím paketem z fronty FIFO a následně provede určenou akce. Data, která jsou určena pro software, jsou odeslána prostřednictvím DMA[1] po sběrnici.

Přístupový bod softwaru (SW Access) slouží primárně pro správu pravidel a zahajování exportu toků. Aby bylo možné v tabulce pravidel uchovávat dostatečně velké množství záznamů, nachází se tato tabulka v externí paměti. Řízení přístupu do této paměti zajišťuje jednotka Memory Arbiter, která se stará o správné prokládání přístupů do paměti, směrování čtených dat mezi komponentami a atomičnost všech paměťových operací.
[15, s. 3]

## 4.3 Software

Předzpracování síťového provozu ve firmwaru je řízeno softwarem. Celý systém SDM je navržen tak, aby byl snadno rozšiřitelný vložením nového SDM monitorovacího pluginu, který implementuje konkrétní monitorování (např. monitorování protokolu SMTP). SDM plugin obsahuje tři rozhraní: [15, s. 1]

- vstupní rozhraní pro pakety s metadaty,
- výstupní rozhraní pro data, která plugin analyzoval, detekoval nebo měřil,
- kontrolní rozhraní pro vyjádření zájmu či nezájmu o jednotlivé toky.

Softwarové pluginy jsou tedy jediným koordinačním prvkem firmwaru. Aby mohl software rozhodnout, zda je pro něho určitý síťový tok zajímavý či nikoli, je nutné ho zpočátku vždy exportovat z hardwaru do softwaru. Teprve po průchodu jisté části paketu (případně i několika paketů) je software schopen rozhodnout, zda ho síťový tok nadále zajímá. V opačném případě je do firmwaru vloženo pravidlo pro zahazování tohoto toku. Po nějaké době (zpravidla velmi krátké) se do softwaru předávají pouze pakety zájmového provozu.
[15, s. 4]

[1] Direct Memory Access

# Kapitola 5

# Návrh

Použití výše zmíněného konceptu SDM s sebou nese několik potenciálních problémů. Protože je veškerá logika řízení hardwarového předzpracování implementována v softwaru, zpětná reakce do hardwaru může trvat dlouho. Provedená akce tedy nemá vliv na první pakety toku. Problémem může být přijetí velkého množství krátkých toků, kdy kvůli dlouhé době odezvy mohou být do softwaru exportovány všechny pakety daného toku. [15, s. 4]

Pro každý příchozí tok je nutné v hardwaru rezervovat položku v tabulce pravidel. Pro velké množství toků je nutné mít velkou kapacitu paměti pro tyto tabulky. Řešením může být dostatečně velká externí paměť, která je připojena k FPGA. Na platformě NetModule ZE7000 se však nenachází žádná taková paměť.

Z těchto několika význačných důvodů byla navržena nová architektura, která je velmi podobná konceptu SDM. Proces monitorování je taktéž rozdělen na hardwarovou a softwarovou část. Oproti konceptu SDM je nezbytné, aby v hardwaru probíhalo inteligentnější předzpracování síťového provozu. Do softwaru je následně předána jen část tohoto provozu, o kterém software rozhoduje, zda je pro něj zajímavý či nikoliv.

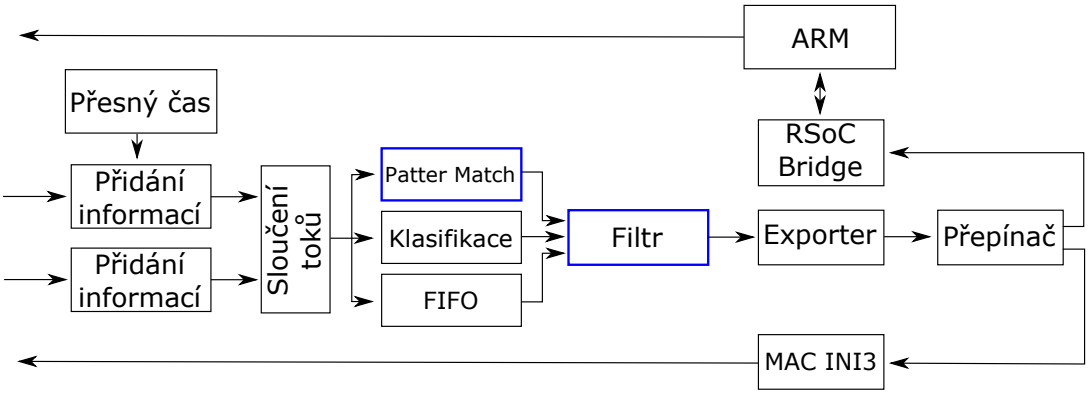
## 5.1 Firmware

Stávající architektura firmwaru Mikrosondy umožňuje filtraci síťového provozu pouze na základě IP adres případně jejich rozsahů. Nová architektura však bude muset umožňovat filtraci jak na základě samostatných IP adres, tak i na základě pětice identifikující síťový tok.

Inteligence předzpracování síťového toku v hardwaru může spočívat v tom, že do softwaru jsou odesílány pouze síťové toky, u kterých je podezření na to, že se jedná o zájmový provoz. Software o těchto tocích následně rozhodne, zda se jedná o požadovanou komunikaci, či nikoli.

K rozhodnutí o tom, zda je daný síťový tok zajímavý pro další zpracování, je výhodné znát typ přenášeného aplikačního protokolu a také informaci o přítomnosti některého specifického identifikátoru (např. konkrétní e-mailovou adresu). Pro tyto účely je do návrhu zahrnuta komponenta Pattern Match (PM), která provádí vyhledávání předem specifikovaných vzorů. Při nalezení požadovaného vzoru je do filtru přidáno pravidlo pro odpovídající síťový tok.

Kompletní návrh architektury je naznačen na obrázku 5.1, modře jsou vyznačeny změny oproti původní. Návrh se oproti stávající architektuře liší v procesní části. Zde se nachází komponenta PM. Každý přijatý paket je paralelně klasifikován v PM a klasifikační kom-



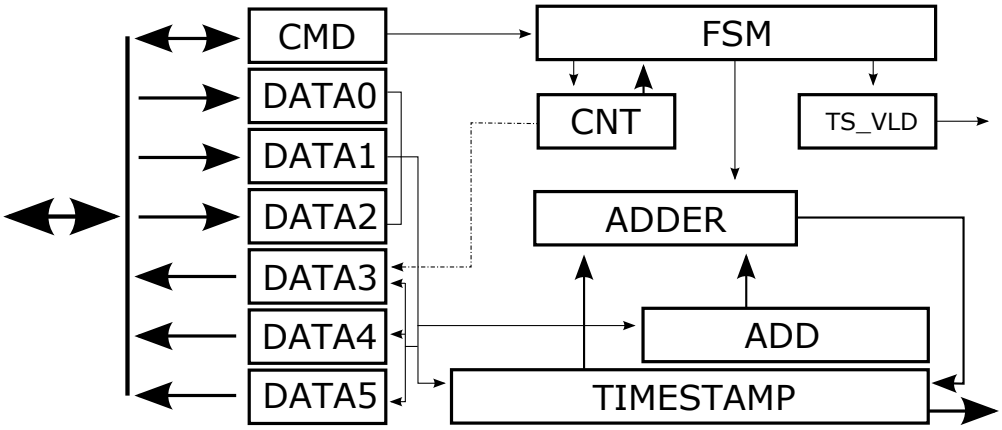

Obrázek 5.1: Návrh firmware Mikrosondy

ponentě. Během klasifikace je každý paket dočasně uložen do vyrovnávací fronty (FIFO). Po vyhodnocení akce, která bude s paketem provedena, je paket z fronty odebrán a propuštěn dále, nebo případně zahozen. Návrh a změny jednotlivých komponent jsou popsány v dalších kapitolách.

### 5.1.1 Jednotka přesného času

Stávající architektura Mikrosondy používá implementaci jednotky přesného času, jejíž licence neumožňuje použití v této práci. Z tohoto důvodu bylo nutné navrhnout novou jednotku přesného času.

Návrh komponenty (zobrazen na obrázku 5.2) obsahuje 96-bitový registr pro uložení časové značky (TIMESTAMP). Šířka registru by měla být dostačující pro rozlišení časových značek s přesností na nanosekundy. K hodnotě nacházející se v tomto registru je každý hodinový takt přičtena hodnota, která se nachází v 48-bitovém registru (ADD). Sčítání provádí sčítačka (ADDER), která musí být schopna sečíst 96-bitovou hodnotu aktuální časové značky a 48-bitovou hodnotu přírůstku.




Obrázek 5.2: Návrh jednotky přesného času

Do návrhu byl dále vložen 32-bitový čítač, který by měl sloužit pro přesné odměření frekvence hardwaru, která se v důsledku zahřívání pouzdra čipu a dalších vlivů může s časem měnit. Způsob využití je popsán u obslužného softwaru.

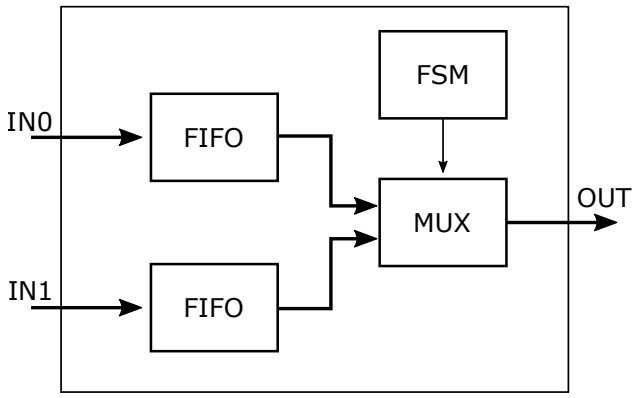
Protože konfigurační sběrnice firmwaru Mikrosondy je 32-bitová, není možné mezi komponentou a softwarem přenášet celý registr časové značky. Z tohoto důvodu je nutné implementovat jednoduchý protokol pro konfiguraci a řízení této komponenty. Pro tento účel je do návrhu vložena sedmice 32-bitových registrů. První z nich (CMD) slouží pro zadávání příkazů komponentě (zápis hodnoty do registru časové značky, přečtení časové značky atd.), ostatní registry pak slouží pro předávání dat. První tři datové registry slouží pro zápis hodnot do registru časové značky a přírůstku. Další trojice datových registrů slouží pro vyčítání stejných hodnot a hodnoty čítače.

Příkazy, které jsou předávány komponentě, jsou vyhodnocovány a řízeny řídicí logikou (FSM). Obvyklými operacemi s jednotkou přesného času jsou především zápis a čtení všech hodnot registrů (časová značka, přírůstek, čítač), dále pak validace a zneplatnění časové značky, spuštění a zastavení čítače.

### 5.1.2 Jednotka pro sloučení datových toků

Ze stejného důvodu, jako nebylo možné použít jednotku přesného času, není možné použít stávající implementaci komponenty pro spojení datových toků a bude nutné ji taktéž navrhnout a implementovat.

Úkolem této komponenty je příjem datových rámců ze dvou vstupů, které následně skládá za sebe do jednoho výstupu. Výběr vstupu, ze kterého se v dalším kroku odebere datový rámec a odešle na výstup, je prováděn algoritmem Round Robin.




Obrázek 5.3: Návrh jednotky pro spojení datových toků

Návrh architektury (viz. obrázek 5.3) obsahuje řídicí stavový automat (FSM), který reflektuje algoritmus Round Robin. Tento plánovací algoritmus cyklicky vybírá vstupy, na jejichž vstupu jsou data. Automat řídí výběr vstupu (MUX), ze kterého budou odebírány rámce. Na každém vstupu je umístěna vyrovnávací fronta. Ta je zde umístěna pro dočasné uložení datového rámce, který je přijímán druhým vstupem. Velikost těchto front je dimenzována přibližně na velikost jednoho paketu.

### 5.1.3 Jednotka Pattern Match

Úkolem této komponenty by mělo být vyhledávání řetězců odpovídajících daným vzorům. Vzory by měli být specifikovány pomocí regulárního výrazu, aby bylo možné snadno popsat konkrétní protokol, identifikátor atd. Pro vyhledávání konkrétního vzoru je možné použít deterministický nebo nedeterministický konečný automat (DKA a NKA).

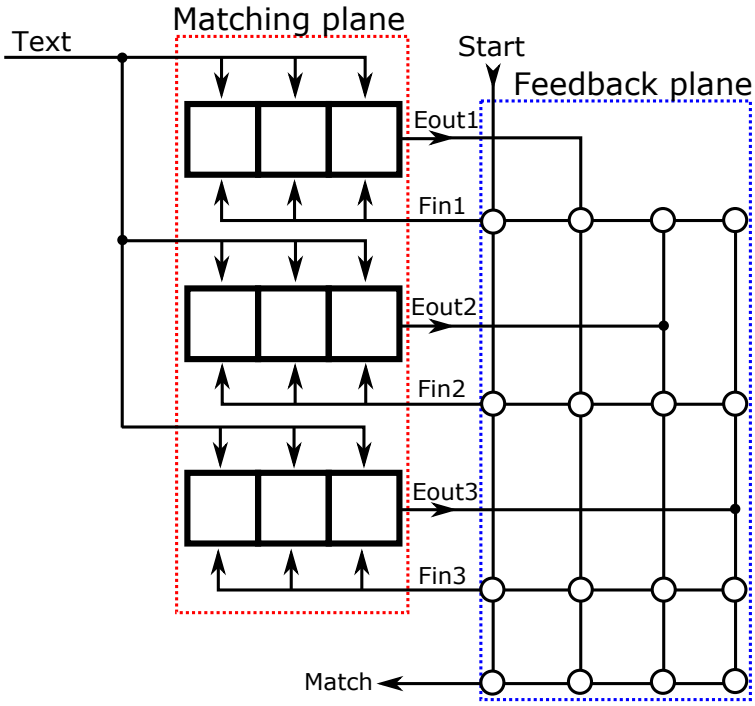
Ve většině FPGA implementací vyhledávání řetězců podle vzoru specifikovaného regulárním výrazem je hledaný výraz uveden již při syntéze obvodu. Výsledný obvod je pak schopen vyhledávat pouze tento vzor. Výhodou tohoto řešení jsou nízké nároky na zdroje a rychlé vyhledávání. Velmi výhodné by však byla možnost tyto vzory zadávat za běhu obvodu. [27, s. 1]

Dynamickou konfiguraci vyhledávaných vzorů umožňuje vyhledávací obvod Takaguchi NKA [27]. Návrh obvodu má dvoudimenzionální strukturu, přičemž jej lze rozdělit na dvě základní části (viz obrázek 5.4), které jsou nazvány Matching Plane (MP) a Feedback Plane (FP).

První z nich provádí porovnávání vstupního textu s řetězci ze zadané množiny $Str(P)$ a druhá reprezentuje provedení přechodu NKA. Nechť $M = (\Sigma, P, \sigma, s, F)$ je automat nad abecedou $\Sigma$, s množinou stavů $P$, přechody $\sigma$, počátečním stave $s$ a množinou koncových stavů, který odpovídá regulárnímu výrazu $P$. Množinu $Str(P)$ pak lze definovat jako:

$$Str(P) = (w \in \Sigma^* \mid \sigma(p, w) \in P, p \in P). \tag{5.1}$$

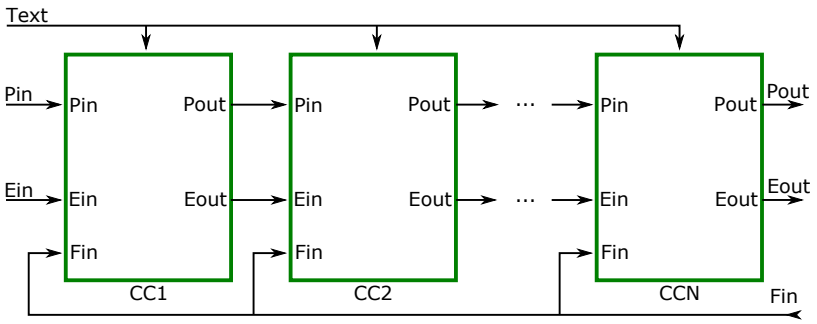
Například pokud pro přechod ze stavu A do stavu B musí NKA přijmout řetězec „abc“, pak „abc“ $\in Str(P)$. [27, s. 3]




Obrázek 5.4: Struktura vyhledávací jednotky [27, s. 3]

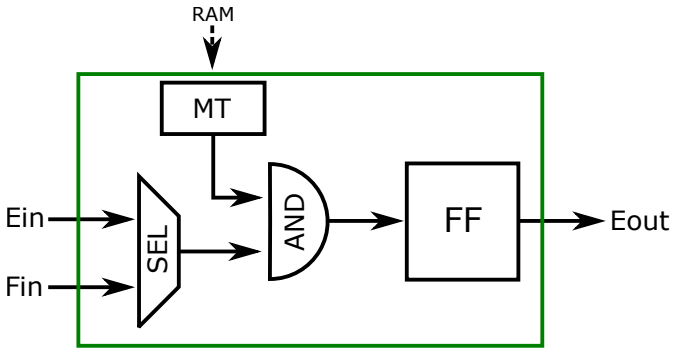
MP se skládá z několika jednorozměrných řad malých procesních jednotek (CC), které provádí porovnávání. Jedna tato řada se nazývá String Matching unit (SMU) a složí pro porovnávání vstupního textu s jedním prvkem množiny $Str(P)$ (naznačeno na obrázku 5.5). Pro každý řetězec $w$ z množiny $Str(P)$ tedy MP obsahuje jednu SMU, která porovnává vstupní text s řetězcem $w$.

CC jsou určeny pro porovnávání znaku textu a možného znaku vzoru. Obrázek 5.6 ukazuje strukturu této jednotky. Ta se skládá z porovnávací tabulky (MT), selektoru (SEL) a Flip-Flop registru (FF). MT je vyhledávací tabulka, která je realizovaná pomocí paměti



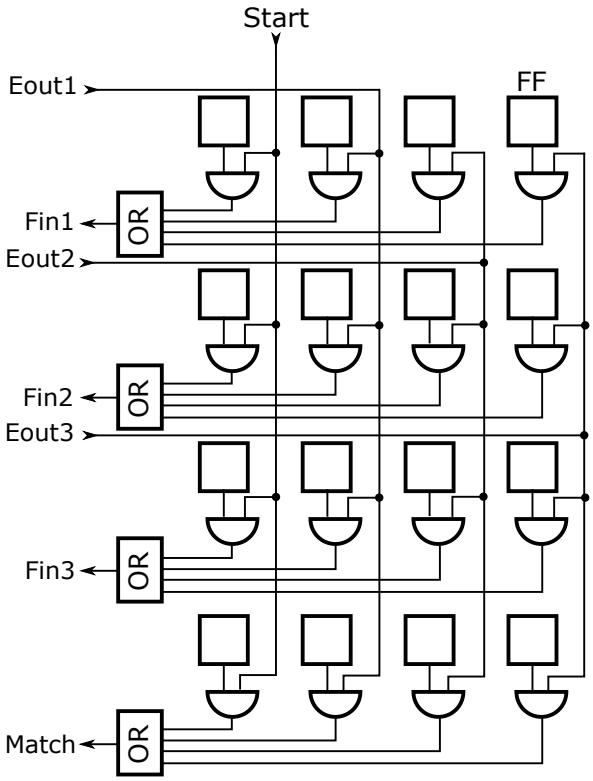

Obrázek 5.5: Jednotka String Matching [27, s. 3]

s náhodným přístupem (RAM) o datové šířce 1 bit. Jako adresa do paměti MT je využit vstupní znak (MT[x]). Hodnota této položky je rovna 1, pokud je vstupní znak shodný s některým ze znaků daných vzorem, v opačném případě je položka rovna 0. MT tedy jednoduše určuje, které znaky se mohou podle vzoru na tomto místě objevovat. Výstup MT je spolu s povolovacím signálem E přiveden na hradlo AND, jehož výstup je uložen do FF. Dle konfigurace je do signálu E procesní jednotky $CC_x$ přiveden povolovací signál $E_{in}$ od procesní jednotky $CC_{x-1}$ nebo $F_{in}$ z FP. [27, s. 3–4]




Obrázek 5.6: Procesní jednotka [27, s. 4]

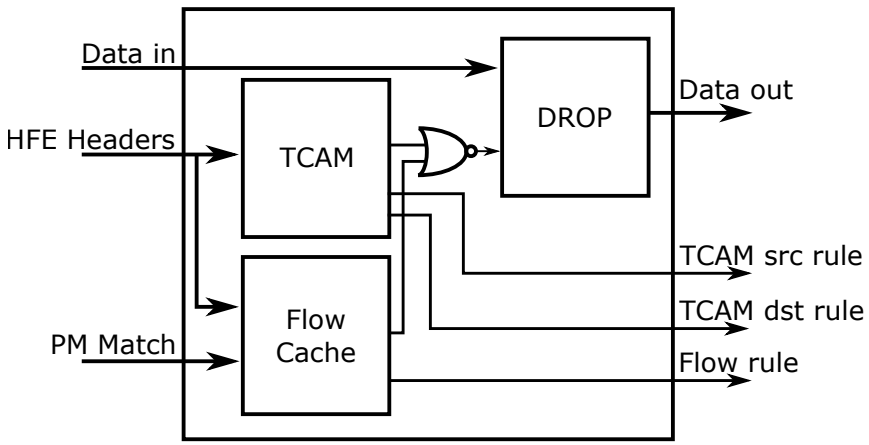
FP realizuje přechody z jednoho stavu NKA do následujícího. Skládá se z $n$ vertikálních vodičů, které jsou připojeny na povolovací signály $E_{out}$ od nejpravější procesní jednotky SMU, a $n$ horizontálních vodičů, které jsou přivedeny na povolovací vstup $F_{in}$ všech procesních jednotek SMU v daném řádku ($n$ je počet řetězců v $Str(P)$ pro vzor P). V FP se dále nachází ještě jeden vertikální vodič, který slouží pro zahájení vyhledávání (Start). V každém místě křížení vertikálního a horizontálního vodiče se nachází programovatelný spínač. Na obrázku 5.7 je naznačena detailní struktura. [27, s. 4]




Obrázek 5.7: Struktura Feedback Plane [27, s. 4]

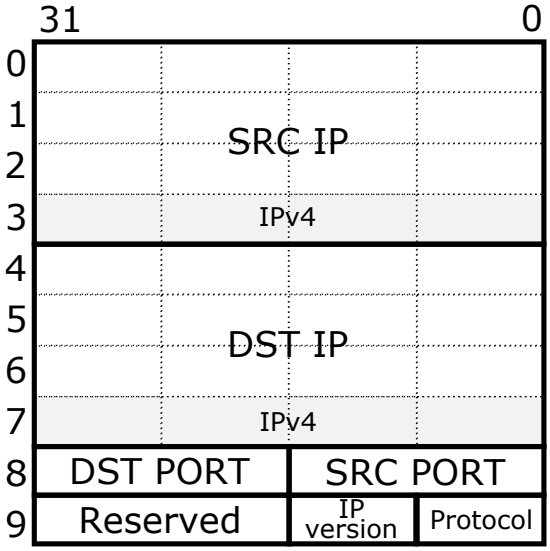
### 5.1.4 Filtrační jednotka

Protože stávající klasifikační jednotka předává do filtru pouze informace o zdrojové a cílové IP adrese přijatého paketu, bude nutné do návrhu firmwaru vložit klasifikační jednotku s novou konfigurací. Pro filtrování podle síťového toku je nutné extrahovat kromě IP adres dále zdrojový a cílový TCP/UDP port a protokol. Výhodné bude exportovat také položku identifikující verzi IP protokolu. Tvar exportovaných dat je naznačen na obrázek 5.9. Pro IP adresy jsou rezervovány 128-bitové položky. IP adresa pro verzi 6 obsadí tuto položku celou, IP adresa verze 4 se pak nachází v posledních 32-bitech této položky.




Obrázek 5.8: Návrh filtrační jednotky

Návrh filtrační jednotky je možné rozdělit na tři základní části, kterými jsou tabulka pravidel pro transportní vrstvu (TCAM), tabulka pravidel pro síťové toky (Flow Cache) a logika zahazování paketu (DROP).

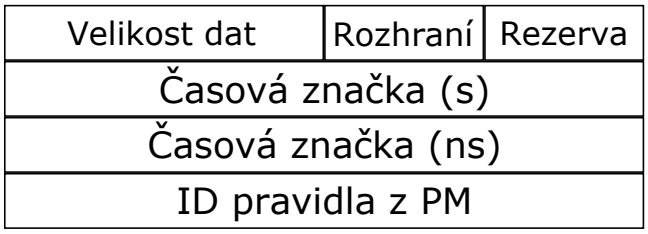



Obrázek 5.9: Tvar informací z klasifikační jednotky

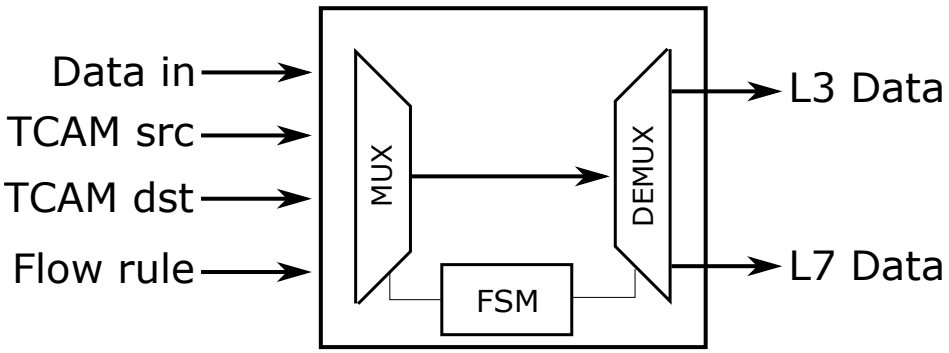
TCAM obsahuje pravidla tvaru zdrojová nebo cílová IP adresa a maska podsítě. Pravidla jsou do této tabulky zapisována pouze ze softwaru. Flow Cache oproti tomu obsahuje pravidla, která se vkládají z pětice zdrojová a cílová IP adresa, zdrojový a cílový UDP/TCP port, číslo protokolu. Do této paměti je možno zapisovat jak ze softwaru, tak i z hardwaru. Hardwarový zápis je nutný v případech, kdy je jednotkou PM nalezen daný vzor. V tomto případě je nutné do tabulky Flow Cache vložit pravidlo pro tento tok. Softwarový zápis je nezbytný při vymazání pravidla, pokud je tok softwarem vyhodnocen jako nezajímavý, nebo při přidání pravidla pro související tok (např. RTP pro SIP, datový kanál pro FTP atd.). Vyhledávání v tabulkách probíhá paralelně. Pokud je v některé z nich nalezeno pravidlo pro daný paket, je tento paket propuštěn dále. V opačném případě je zahozen. Spolu s paketem je nutné dále předat i identifikátor pravidla, podle kterého byl paket propuštěn. Pro TCAM je nutné rozeznat, zda shoda nastala ve zdrojové nebo cílové IP adrese. Schéma návrhu filtrační jednotky je naznačeno na obrázku 5.8.

### 5.1.5 Výstupní jednotka

Úkolem výstupní jednotky je vytvořit INI3 hlavičku před každým rámcem. Nový návrh však bude muset dále rozhodnout, zda se jedná o pakety propuštěné TCAM (L3) nebo o pakety propuštěny Flow Cache (L7), případně oboje. Pro každou variantu bude vytvořen jiný tvar hlavičky a odeslán jiným výstupem. Pro L3 pakety je vytvořena standardní INI3 hlavička (viz. 3.2). Pakety, které byli propuštěny na základě L7 pravidla, obsahují v INI3 hlavičce místo dvou 16-bitových položek identifikující L3 pravidla pouze jednu 32-bitovou hodnotu, která identifikuje L7 pravidlo (viz obrázek 5.10). Po vytvoření odpovídající hlavičky je paket odeslán na odpovídající výstup. Schéma návrhu jednotky je naznačeno na obrázku 5.11.




Obrázek 5.10: Tvar INI3 hlavičky pro L7




Obrázek 5.11: Schéma výstupní jednotky

## 5.2 Software

Stávající architekturu softwaru bude třeba upravit tak, aby bylo možné předávat data z hardwaru odpovídajícím softwarovým modulům, které budou data dále analyzovat.

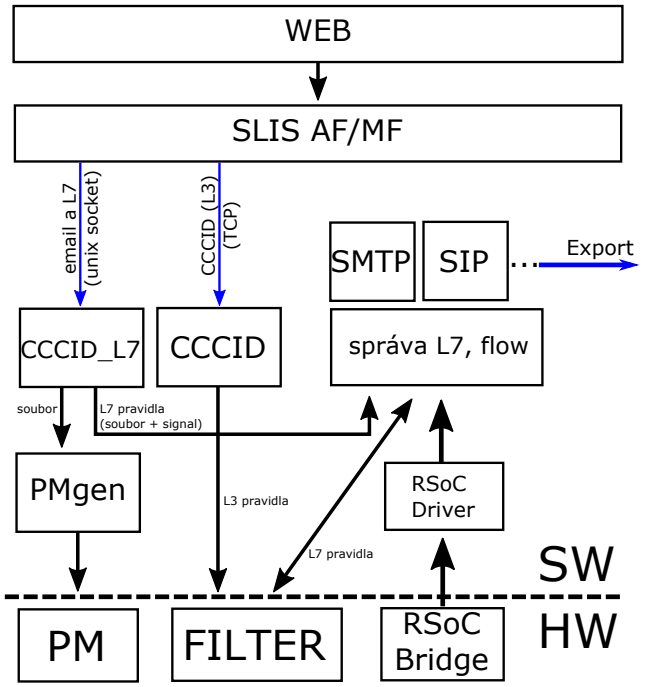
Návrh nové struktury software bude zahrnovat změny aplikace CCCID, která zajišťuje konfiguraci filtrační jednotky. Dále bude nutné vytvořit jednotné rozhraní pro moduly, které budu analyzovat přijatá data, a aplikaci pro konfiguraci PM. Protože návrh firmwaru obsahuje novou jednotku přesného času, bude nutné navrhnout aplikaci, které ji bude obsluhovat.

Celkový návrh software je naznačen na obrázku 5.12. Detaily k jednotlivým součástem jsou uvedeny v následujících kapitolách.

### 5.2.1 Aplikace pro jednotku přesného času

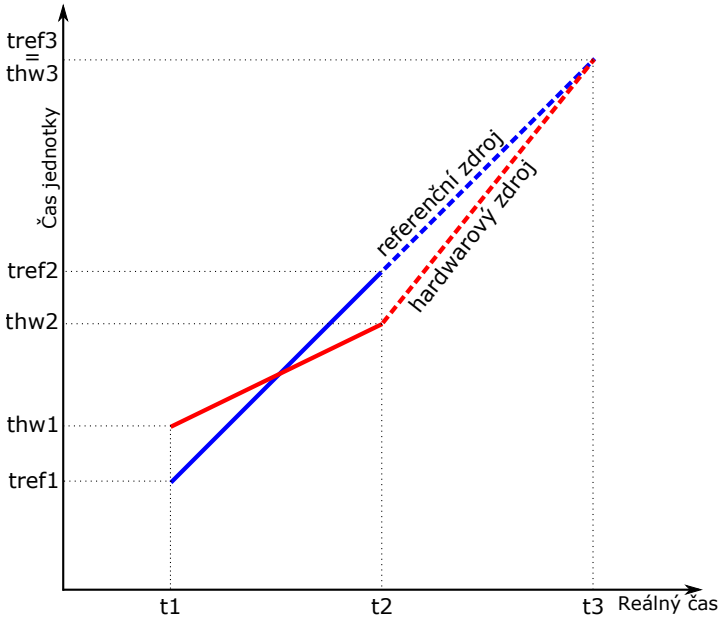
Aplikace pro jednotky přesného času slouží pro její konfiguraci a především pro její korekci. Protože zdroj hodinového signálu, který je přiveden do FPGA, nemusí být dostatečně přesný, s měnící se teplotou se může měnit i frekvence tohoto zdroje, je nutné provádět korekci podle nějakého referenčního zdroje. Referenčním zdrojem může být jiné zařízení v síti (protokol NTP – Network Time Protocol), RTC modul a jiné. Varianta s RTC modulem má tu výhodu, že zařízení nemusí být připojeno k Internetu.

Po nastavení aktuálního času do jednotky přesného času je možné provádět korekci pouze pomocí hodnoty přírůstku, který je přičítán každý hodinový takt. Cílem je nastavit hodnotu přírůstku hardwarového zdroje času tak, aby se generovaný čas co nejvíce přibližoval tomu referenčnímu.




Obrázek 5.12: Návrh softwaru

Pro algoritmus korekce tedy předpokládáme, že referenční zdroj hodin ukazuje čas přesně. Za jednu sekundu reálného času uplyne na referenčním zdroji hodin také sekunda. U hardwarového zdroje může čas plynout buď pomaleji, nebo rychleji. Na obrázku 5.13 je naznačeno, jak plyne čas na referenčním zdroji času (modrá křivka) a na hardwarovém zdroji (červená křivka).




Obrázek 5.13: Graf znázorňující korekci času

Vstupem korekce časové značky komponenty jsou časy referenčního zdroje ($t_{ref1}$ a $t_{ref2}$) a časy hardwarové komponenty ($t_{hw1}$ a $t_{hw2}$) sejmuty v časech $t_1$ a $t_2$. V čase $t_3 = t_2 + (t_2 - t_1)$ bude referenční čas jistě roven $t_{ref3} = t_{ref2} + (t_{ref2} - t_{ref1})$. Pro čas hardwarové komponenty v čase $t_3$ má pak platit $t_{ref3} = t_{hw3}$.

Protože známe hodnotu přírůstku ($ADD$), víme, že za čas $t_2 - t_1$ byla hodnota přičtena $n$-krát, kde $n$ je dáno vtahem:

$$n = \frac{t_{hw2} - t_{hw1}}{ADD}. \tag{5.2}$$

Hodnota $n$ tedy udává frekvenci. Za $n$ taktů chceme, aby se $t_{hw3}$ rovnalo $t_{ref3}$. Novou hodnotu přírůstku ($ADD_{new}$) lze tedy vypočítat jako:

$$ADD_{new} = \frac{t_{hw3} - t_{hw2}}{n}. \tag{5.3}$$

Po dosazení za $t_{hw3}$ a $n$ dostáváme vztah:

$$ADD_{new} = ADD \cdot \frac{2 \cdot t_{ref2} - t_{ref1} - t_{hw2}}{t_{hw2} - t_{hw1}}. \tag{5.4}$$

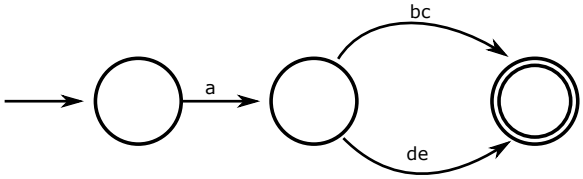
Korekce bude tedy vypadat tak, že se periodicky (jednou za sekundu) bude přepočítávat hodnota přírůstku, podle výše uvedeného vzorce.

Pro prvotní nastavení jednotky je možné využít čítač, který je zahrnut v návrhu jednotky. Ten bude sloužit pro nastavení přírůstku. V čase $t_1$ spustíme čítač, který v čase $t_2$ zastavíme (nesmí dojít k přetečení). Aktuální frekvenci hardwaru vypočítáme jako $f = \frac{N}{t_2 - t_1}$ ($N$ je hodnota čítače). Hodnota přírůstku je potom $ADD = \frac{1}{f}$. Tato hodnota je v sekundách a je nutné ji přizpůsobit, protože registr $ADD$ obsahuje zlomky sekund.

### 5.2.2 Aplikace pro Pattern Match

K navržené jednotce hledající vzory je nutné navrhnou a implementovat aplikaci pro konfiguraci vyhledávaných vzorů (za běhu systému). Konfigurace jednotky spočívá především v nastavení SMU a spínačů ve FP.

Nechť $P$ je hledaný vzor a $M$ je NKA pro tento vzor. Nechť $Str(P)$ je množina řetězců, z nichž každý je použit pro definování nějakého přechodu NKA $M$. Pro každý řetězec w z $Str(P)$ je nastavena jedna SMU tak, aby jednotlivé procesní jednotky obsahovali v paměti znak řetězce $w$ na odpovídajících pozicích (řetězce jsou zarovnány k pravé straně SMU). Výběr povolovacího signálu (SEL) v každé jednotce CC je nastaven tak, aby jednotka CC obsahující první znak řetězce přijímala povolovací signál $F_{in}$ a další jednotky CC v jedné SMU přijímaly povolovací signál $E_{out}$ od sousední jednotky (zleva). Toto nastavení způsobí, že SMU dává vědět, že našlo zadaný řetězec, který je nutný pro uskutečnění přechodu obecně ze stavu $p$ do $q$.

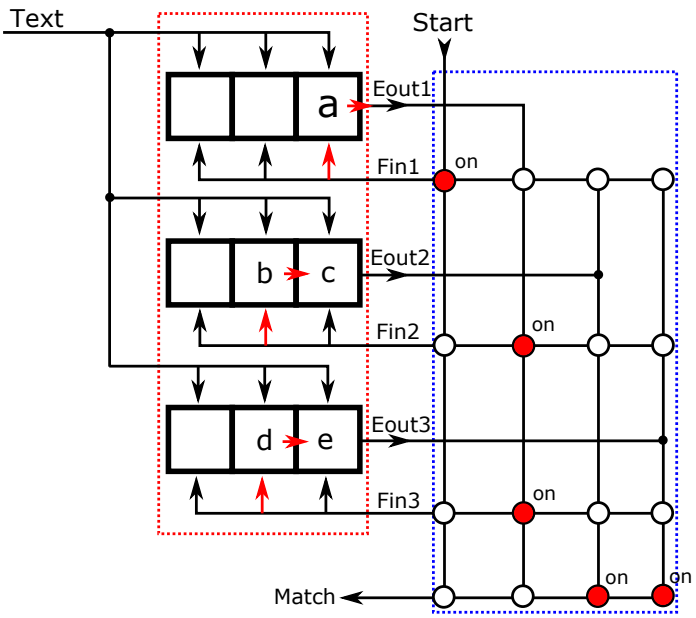



Obrázek 5.14: NKA pro regulární výraz $P = a(bc|de)$

Nastavení FP potom spočívá v nastavování propojek. Každý signál $E_{out}[p]$ jednotky SMU, který je napojen na svislý vodič FP, reprezentuje jeden stav počáteční stav $p$ přechodu NKA $M$ a vodorovné vodiče, které jsou napojeny na vstupní signál $F_{in}$, odpovídá následujícímu stavu $q$ přechodu NKA $M$. Pokud tedy platí, že $\delta(p, w) = q$, pak budou vodiče $E_{out}[p]$ a $F_{in}[q]$ propojeny (zápis jedničky do odpovídající propojky). Počáteční stav NKA $M$ pak reprezentuje vodič Start, koncový stav automatu značí vodič Match.

Pro názornou ukázku použijeme regulární výraz $P = a(bc|de)$. NKA pro tento regulární výraz je zobrazeno na obrázku 5.14.

Množina $Str(P)$ obsahuje tedy řetězce „a“, „bc“, „de“. Každá SMU tedy obsahuje jeden z těchto řetězců. Nastavení propojek pak reflektuje NKA odpovídající tomuto vzoru. Nastavení je znázorněno na obrázku 5.15.

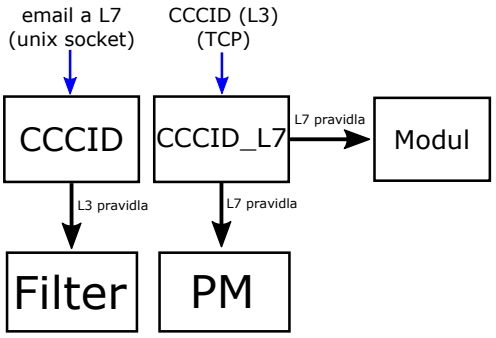



Obrázek 5.15: Nastavení MP a FP pro regulární výraz $P = a(bc|de)$

### 5.2.3 Aplikace CCCID

Stávající implementace aplikace CCCID podporuje pouze konfiguraci pro konkrétní IP adresy. Do návrhu je tedy nutné vložit novou část, která bude schopna na základě podnětů z konfiguračního rozhraní správně nastavit jednotku PM a informovat softwarové moduly o přidání pravidla.

Návrh nové aplikace CCCID rozděluje funkcionalitu na dvě části. První z nich je nezměněná verze CCCID, která naslouchá na TCP spojení, na kterém očekává pravidla. Druhá část (CCCID_L7) slouží pro příjem pravidel pro aplikační protokoly. Předávání pravidel je uskutečněno přes unixový soket. Po obdržení pravidla z konfiguračního rozhraní je vygenerována konfigurace pro PM a pravidlo je předáno odpovídajícímu modulu.




Obrázek 5.16: Návrh aplikace CCCID

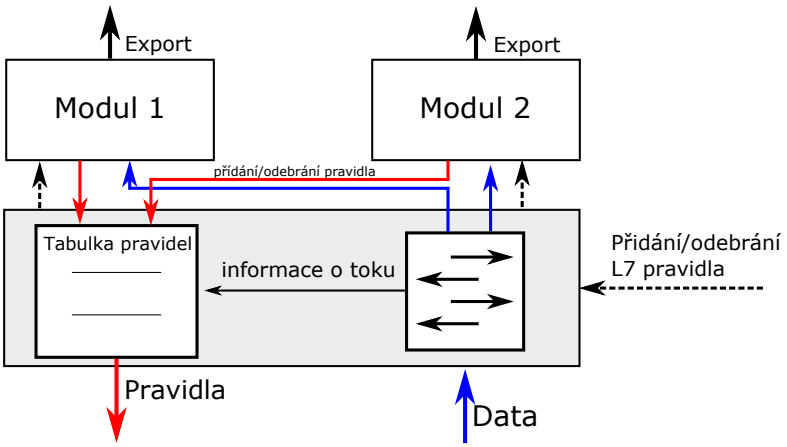
### 5.2.4 Monitorovací moduly

Pro softwarové monitorovací moduly bude nutné navrhnout jednotné rozhraní pro příjem a export dat. Toto rozhraní má na starost především obsluhu tabulky pravidel ve Flow Cache a směrování paketů do odpovídajících modulů.

Aby mohl software efektivně spravovat tabulku pravidel ve Flow Cache a předcházet jejímu zaplnění, musí mít neustálý přehled o položkách, které se v ní nacházejí. Protože neustálé čtení tabulky z HW je neefektivní a pomalé, je nutné implementovat softwarovou tabulku pravidel, která ji zrcadlí. Tabulka je plněna informacemi o síťových tocích podle paketů, které byly do softwaru exportovány. U každého záznamu síťového toku je navíc položka obsahující časovou značku posledního přijatého paketu k tomuto toku. V případě, že se tabulka toků téměř zaplní, je možné z tabulky odstranit toky, které byly nejdéle neaktivní.

Každý modul má čtyři základní rozhraní sloužící pro:

- příjem dat pro analýzu,
- pro export dat,
- mazání nebo přidávání pravidel do filtrační jednotky (Flow Cache),
- přidání pravidla do modulu.




Obrázek 5.17: Rozhraní modulů pro analýzu provozu

Směrování paketů do modulů se provádí na základě čísla pravidla, které se nachází v INI3 hlavičce. Toto pravidlo reprezentuje bitovou mapu, kde každý bit představuje jeden modul. V případě, že data patří danému modulu, nastaví se příslušný bit. S každým přijatým paketem je pak do softwarové tabulky poznačen tok a časová značka posledního výskytu paketu pro tento tok.

Na základě podnětů z webového rozhraní aplikace CCCID_L7 předává pravidla jak generátoru konfigurace pro PM, tak i rozhraní pro moduly. Pravidla jsou předána pouze modulům, pro které jsou určena. Při odstraňování či přidávání pravidla modulem, rozhraní pravidlo vloží jak do hardwarové tabulky, tak i softwarové. Celá struktura tohoto rozhraní je naznačena na obrázku 5.17.

# Kapitola 6

# Implementace

Pro implementaci nové varianty firmwaru Mikrosondy byly převzaty a použity některé komponenty a softwarové aplikace, které byly vytvořeny v rámci projektu *Sondy pro analýzu a filtraci provozu na úrovni aplikačních protokolů.*

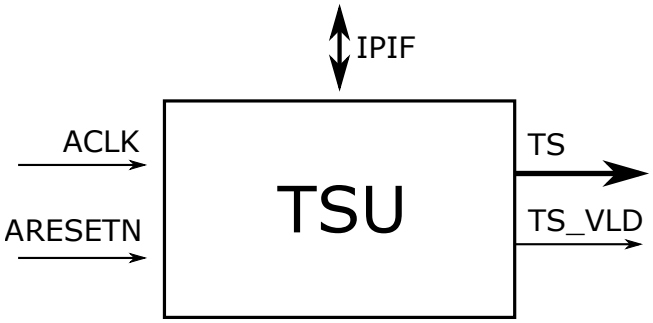
## 6.1 Firmware

Komponenty v nově implementovaném firmware Mikrosondy reflektují výše uvedené návrhy. Pro přenos dat do softwaru se oproti původní implementaci využívají dva datové kanály. Jedním z kanálů jsou do softwaru exportovány pakety, které vyhoví některému z L3 pravidel, druhým kanálem jsou pak exportovány pakety, které vyhoví některému z L7 pravidel. Software tak nebude muset rozpoznávat, o jaký typ paketu se jedná. Protože jsou kanály nezávislé, je možné navíc oba přenosy realizovat paralelně.

Pro implementaci byly převzaty a použity komponenty Pattern Match a komponenta Filter. V dalších kapitolách bude popsána implementace nových komponent a popis rozhraní převzatých komponent.

### 6.1.1 Komponenta přesného času

Rozhraní komponenty pro přesné časové značky (TSU) se skládá z konfigurační sběrnice IPIF [29, s. 15–16], 64-bitového vektoru časové značky (TS) a signálu, který označuje časovou značku jako platnou (TS_VLD). Časová značka je unixového tvaru, kdy 32 horních bitů udává počet sekund a spodních 32 bitů udává počet nanosekund od data 1.1. 1970 .




Obrázek 6.1: Rozhraní komponenty

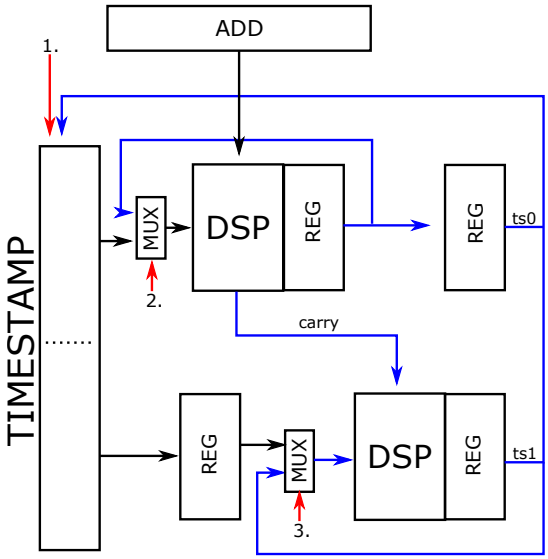
Při implementaci jednotky přesného času bylo nutné nejdříve implementovat komunikační rozhraní. To se skládá z již zmíněné sedmice registrů. Protože se do první trojice registrů pouze zapisovalo a z druhé se pouze četlo, byly tyto trojice umístěny na stejné adresy. Pokud budeme na tyto adresy zapisovat, hodnota se zapíše do první trojice registrů. V případě čtení jsou hodnoty čteny z druhé trojice registrů. Adresový prostor komponenty vypadá následovně:

- 0x00 (CMD) – zadávání příkazů,
- 0x04 (DATA0) – pro zápis do datového registru 0 a čtení datového registru 3,
- 0x08 (DATA1) – pro zápis do datového registru 1 a čtení datového registru 4,
- 0x0C (DATA2) – pro zápis do datového registru 2 a čtení datového registru 5,
- 0x10 (FREQUENCY) – výchozí frekvence (udána při syntéze).

První registr slouží pro zadávání příkazů. Příkazy, které komponenta implementuje jsou:

- 0x0 – žádná operace, předchozí operace byla dokončena,
- 0x1 – označení časové značky jako platnou,
- 0x2 – zneplatnění časové značky,
- 0x3 – nastavení registru časové značky z datových registrů,
- 0x4 – nastavení registru přírůstku z prvních dvou datových registrů,
- 0x5 – zachycení časové značky do datových registrů,
- 0x6 – zachycení registru přírůstku do datových registrů,
- 0x7 – start čítače,
- 0x8 – zastavení čítače a přenesení do prvního datového registru.

Nejpodstatnější součástí komponenty je sčítačka, která sčítá 96-bitový registr časové značky a 48-bitový registr přírůstku. Pro ušetření zdrojů a zlepšení časování je výhodné použít DSP bloky. Na čipu Xilinx Zynq jsou přítomny bloky DSP48E1, které při správné konfiguraci umožňují sčítání 48-bitových hodnot. Protože potřebujeme sečíst 96-bitovou hodnotu, bude třeba použít tyto bloky dva. Aby bylo možné výstup DSP bloků zavést zpět na jejich vstup, je třeba na jejich výstupech aktivovat registry. Tyto registry však zpozdí výstupní data a signál přetečení prvního bloku o jeden takt. Z tohoto důvodu návrh obsahuje další dva registry. Jeden se nachází na výstupu prvního DSP bloku a druhý na vstupu druhého DSP bloku. Ve výsledku vznikne řetězená linka, která je naznačena na obrázku 6.2.




Obrázek 6.2: Architektura sčítačky

Problémem takto řetězené linky je situace, kdy je nutné nastavit novou hodnotu registru TIMESTAMP. Po zapsání hodnoty do registru časové značky obsahují ostatní registry řetězené linky původní (již neplatnou) časovou značku (na obrázku 6.2 jsou modře vyznačeny signály s neplatnou hodnotou). Pro správné zapsání registru TIMESTAMP se využijí k tomuto účelu přidané multiplexory (na obrázku označeno jako MUX). Zápis nové časové značky bude třeba provést ve třech krocích (hodinových cyklech), které jsou v obrázku naznačeny červenými šipkami:

1. Nové hodnoty se zapíší do registru TIMESTAMP.
2. První multiplexor je přepnut tak, aby se do prvního DSP bloku zkopírovala nová hodnota. Během tohoto kroku je zakázán zápis nově vypočítané časové značky do registru TIMESTAMP a je zneplatněn výstup komponenty.
3. Druhý multiplexoru je přepnut tak, aby se do druhého DSP bloku přenesla nová hodnota časové značky. Během toho je stále zakázaný zápis nově vypočítané časové značky do registru TIMESTAMP a je zneplatněn výstup komponenty.

Horních 32 bitů výstupu komponenty značí počet sekund a spodních 32 bitů pak počet nanosekund. Výhodné tedy bude, pokud 32 nejvyšších bytů registru časové značky bude reprezentovat sekundy a dalších 64 bitů pak zlomky sekund. Pro převedení následujících 64 bitů na nanosekundy stačí hodnotu tohoto registru vynásobit číslem $10^9/2^{64}$. To je možné udělat následovně:

1. Z 64 bitů použijeme pouze 32 nejvyšších, což reprezentuje dělení číslem $2^{32}$.
2. Výsledek předchozího kroku vynásobíme číslem 0x3B9ACA00 ($10^9$).
3. Z 64 bitového výsledku použijeme horních 32 bitů (dělení číslem $2^{32}$).

Množství zdrojů spotřebovaných implementací této komponenty je uvedeno v tabulce 6.1. Tyto hodnoty byly zjištěny po syntéze nástrojem Vivado 2015.1.

| LUT | FF | BRAM | DSP |
|---|---|---|---|
| 565 | 738 | 0 | 2 |

Tabulka 6.1: Zdroje komponenty přesného času

### 6.1.2 Komponenta pro sloučení datových toků

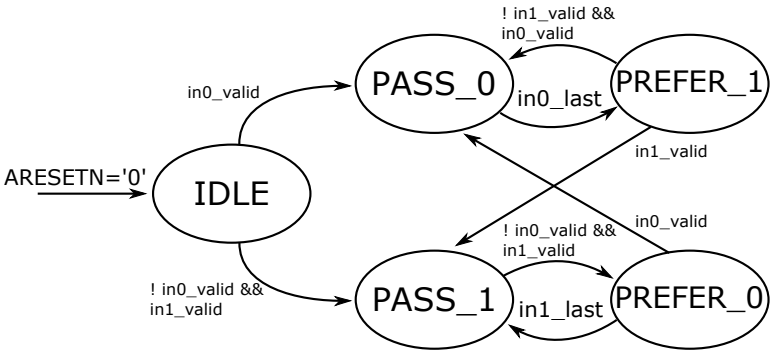
Rozhraní komponenty pro slučování datových toků se skládá za dvou vstupních rozhraní AXI-Stream [5] a jednoho výstupního rozhraní AXI-Stream. Generickými parametry komponenty jsou především:

- C_IN_FIFO_DEPH – udává kapacitu každé vstupní fronty,
- C_OUT_FIFO_DEPH – specifikuje kapacitu výstupní fronty.

Komponenta se skládá především z řídicí logiky a dvojice front FIFO. Každá z front je umístěna na jednom ze dvou datových vstupů. Do těchto front jsou ukládána data, která není možné okamžitě odebírat ze vstupu (dat jsou odebírána z druhého vstupu nebo je výstup z nějakého důvodu dočasně pozastaven). Na výstupu se nachází další vyrovnávací fronta, do které se ukládají data, dokud nebude přijímající komponenta opět připravena k jejich zpracování.

Řídicí logika je implementována pomocí stavového automatu (FSM). Jejím úkolem je pomocí algoritmu Round Robin vybírat vstup, ze kterého se budou odebírat data. Výběr a přepnutí multiplexoru na odpovídající vstup je však možné pouze na rozhraní dvou paketů.

Řídicí stavový automat obsahuje pět stavů (viz obrázek 6.3). Po resetu komponenty se stavový automat dostane do výchozího stavu (IDLE). V tomto stavu se provádí prvotní výběr vstupu. Vybírá se ze vstupů, které ve vstupní frontě obsahují data. Pokud se data nachází v obou frontách, je preferován první vstup. Po vybrání vstupu se automat nachází ve stavu PASS_0 (PASS_1), ve kterém na výstup přepošle jeden paket z prvního (druhého) vstupu. Po přeposlání paketu automat přejde do stavu PREFER_1 (PREFER_0), ve kterém je prováděn výběr vstupu podobně jako ve stavu IDLE. Ve stavu PREFER_0 je preferován první vstup, oproti tomu ve stavu PREFER_1 je preferován druhý vstup.




Obrázek 6.3: Stavový automat komponenty pro sloučení datových toků

Množství zdrojů nezbytných pro implementaci komponenty pro slučování toků je uvedeno v tabulce 6.2. Vybrána byla konfigurace s 32-bitovými frontami o hloubkách 16 položek.

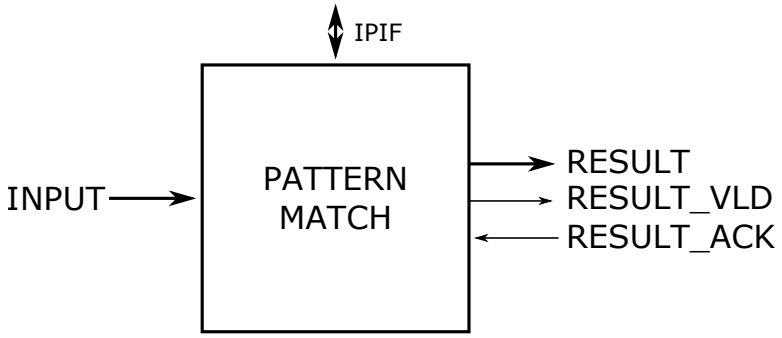
| LUT | FF | BRAM | DSP |
|---|---|---|---|
| 171 | 177 | 0 | 0 |

Tabulka 6.2: Zdroje komponenty pro spojení datových toků

### 6.1.3 Komponenta Pattern Match

Rozhraní komponenty se skládá z konfiguračního rozhraní IPIF, vstupního rozhraní AXI-Stream pro příjem dat, ve kterých jsou vyhledávány vzory. Výsledek vyhledávání je pak předáván pomocí trojice signálů:

- RESULT – bitová mapa identifikující definovanou skupinu vzorů, se kterými byla nalezena shoda (každá skupina má rezervován jeden bit).
- RESULT_VLD – značí, že RESULT je platný.
- RESULT_ACK – slouží pro potvrzení přečtení výsledku.




Obrázek 6.4: Rozhraní komponenty Pattern Match

Entita komponenty obsahuje generické parametry, kterými lze komponentu nastavovat. Vybranými parametry jsou:

- CHAR_WIDTH – datová šířka jednoho znaku (v bitech),
- STRINGS – počet vzorů, které je komponenta schopna vyhledávat,
- STRING_LEN – maximální délka řetězce, které může být vzorem specifikován,
- GROUPS – počet skupin vzorů,
- USE_CONTERS – definuje, zda mají být zahrnuty statistické registry.

Porovnávací tabulka (MT), která byla uvedena v návrhu, je implementována pomocí blokové RAM paměti (BRAM). Do této paměti je nahrávána konfigurace pro procesní jednotky. Flip-Flop registry pro přepínače Feedback Plane (FP) jsou implementovány jako posuvný registr, do kterého je postupně nasouvána konfigurace.

Komponenta umožňuje nahrání dvou konfigurací, mezi kterými je možno přepínat. Každý spínač FP tedy obsahuje dva Flip-Flop a paměť BRAM je rozdělena na dvě části. Jedna z konfigurací je aktivní a druhá se může postupně nahrávat. Přepnutí komponenty do druhé konfigurace je provedeno v jednom hodinovém taktu.

Adresový prostor komponenty je následující:

- 0x0 (RW):
  - bit 0 – při zápisu značí příkaz pro přepnutí konfigurace, při čtení indikuje přepínání konfigurace,
  - bit 1 – reset konfigurací (je nutné aby byl nastaven i bit 0),
  - bit 2–31 – identifikace verze komponenty a rezerva,
- 0x04 (W) – bity 0–31 paměti BRAM,
- 0x08 (W) – bity 32–63 paměti BRAM,
- 0x0C (W) – po zápisu do tohoto registru se provede zápis 72-bitové hodnoty do paměti BRAM:
  - bity 0–7 – bity 64–71 paměti BRAM,
  - bity 8–15 – adresa v paměti,
  - bity 16–23 – adresa BRAM v rámci jednoho řádku BRAM,
  - bity 24–31 – adresa řádku BRAM,
- 0x10 (W) – vstupní data, která jsou nasouvána do posuvného registru.

Množství zdrojů nezbytných pro implementaci komponenty Pattern Match je uvedeno v tabulce 6.3. Vybrána byla konfiguraci s datovou šířkou znaku (CHAR_WIDTH) 8 bitů, dvanácti vyhledávanými řetězci (STRINGS) o délce 36 znaků (STRING_LEN) a pouze jednou výstupní skupinou.

| LUT | FF | BRAM | DSP |
|---|---|---|---|
| 990 | 1657 | 24 | 0 |

Tabulka 6.3: Zdroje komponenty Pattern Match

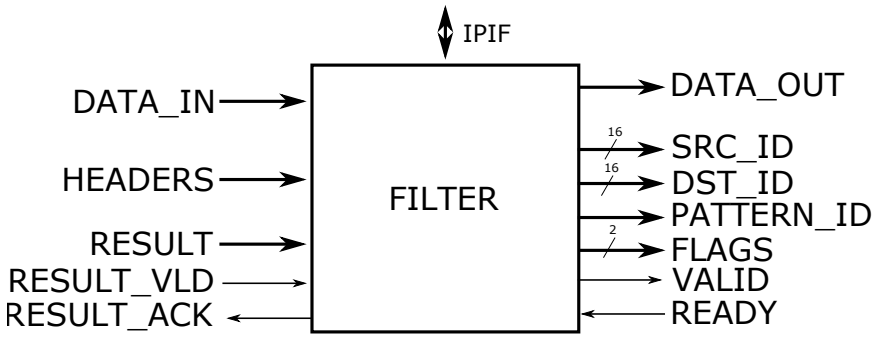
### 6.1.4 Komponenta Filter

Rozhraní komponenty se skládá z konfigurační sběrnice IPIF. Vstupem komponenty jsou signály z PM a dvě sběrnice AXI-Stream. První z nich slouží pro předání dat z klasifikační jednotky, druhá slouží jako vstup paketů. Nejdůležitějšími generickými parametry komponenty jsou:

- CAM_ITEMS – kapacita tabulky TCAM (pro L3 pravidla),
- CUCKOO_TABLES – počet tabulek ve Flow Cache (pro L7 pravidla),
- CUCKOO_TABLE_ITEMS – kapacita jednotlivých tabulky ve Flow Cache.

Výstup komponenty se skládá ze signálů:

- DATA_OUT – AXI-Stream sběrnice pro pakety, které odpovídají některému z pravidel, které se nachází v tabulkách,




Obrázek 6.5: Rozhraní komponenty filtrace

- SRC_ID – identifikace použitého L3 pravidla pro zdrojovou IP adresu,
- DST_ID – identifikace použitého L3 pravidla pro cílovou IP adresu,
- PATTERN_ID – obsahuje bitovou masku, kterou filtrační jednotce předal PM (identifikace L7 pravidel),
- FLAGS – obsahuje vektor příznaků:
  - bit 0 – značí, že byl v paketu nalezen některý ze vzorů (komponentou PM) a podařilo se do filtru úspěšně vložit pravidlo pro odpovídající síťový tok,
  - bit 1 – značí, že se pravidlo pro odpovídající síťový tok nepodařilo přidat v důsledku zaplnění tabulek,
- VALID – značí platnost výstupních hodnot,
- READY – následující komponenta tímto signálem značí připravenost pro příjem hodnot.

Množství zdrojů nutných pro implementaci komponenty Filter je uvedeno v tabulce 6.4. Vybrána byla konfigurace komponenty s 64 záznamy v tabulce TCAM a čtyřmi tabulky ve Flow Cache o kapacitě 1024 záznamů.

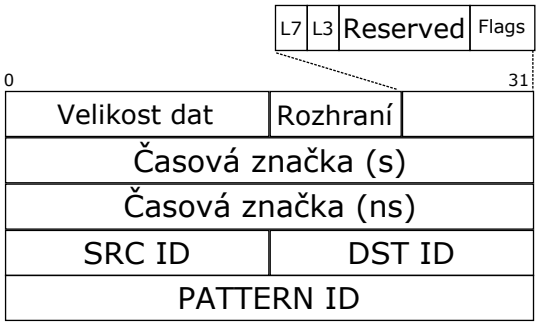
| LUT | FF | BRAM | DSP |
|---|---|---|---|
| 2170 | 2973 | 12 | 0 |

Tabulka 6.4: Zdroje komponenty filtrace

### 6.1.5 Výstupní komponenty

Výstupní jednotka ze skládá ze dvou komponent. První z nich je komponenta BLOB_HDR_COMPLETER. Tato komponenta přijímá data a informace od filtrační komponenty a vytvoří před každý paket hlavičku, která je téměř totožná s INI3 hlavičkou (viz. 3.2). Jedná se o modifikaci, která obsahuje navíc 32-bitové slovo obsahující identifikaci L7 pravidla, příznaky FLAGS od filtrační jednotky a informaci i tom, kam má být paket na čipu směrován (viz. obrázek 6.6).

Komponenta je implementována jako multiplexor, který vybírá mezi výstupními informacemi (SRC_ID, DST_ID, PATTERN_ID, FLAGS) a daty (DATA_OUT) od filtrační

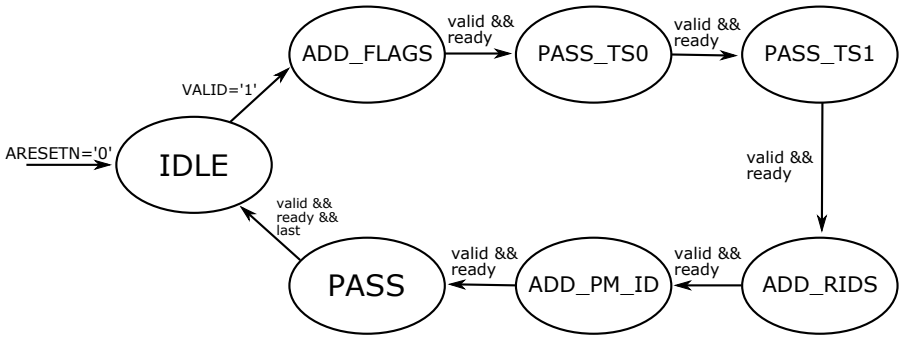



Obrázek 6.6: Tvar hlavičky na výstupu komponenty BLOB_HDR_COMPLETER

jednotky. Tímto způsobem se sestavuje uvedená INI3 hlavička (tvar viz obrázek 6.6). Multiplexor je řízen stavovým automatem, který je naznačen na obrázku 6.7.

Řídicí stavový automat obsahuje následující stavy, mezi kterými se postupně přechází v uvedením pořadí:

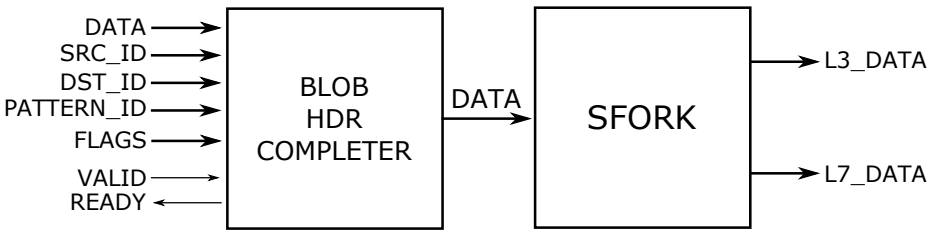
- IDLE – v tomto stavu komponenta čeká na platné hodnoty od filtrační jednotky (nastavení signálu VALID).
- ADD_FLAGS – do částečné INI3 hlavičky jsou vloženy příznaky FLAGS a směrovací informace, které říkají, zda byl paket exportován na základě L3 nebo L7 pravidla.
- PASS_TS0 – ve stavu je propuštěno první slovo časové značky.
- PASS_TS1 – ve stavu je propuštěno druhé slovo časové značky.
- ADD_RIDS – jsou vloženy identifikace L3 pravidel (SRC_ID a DST_ID).
- ADD_PM_ID – do hlavičky je vložena identifikace L7 pravidla (bitmapa od PM).
- PASS – v tomto stavu je propuštěn zbytek paketu.




Obrázek 6.7: Řídicí automat komponenty BLOB_HDR_COMPLETER

Druhou komponentou na výstupu je SFORK. Tato komponenta má za úkol přeposlat paket na jeden ze dvou vstupů. Výběr výstupů je prováděn na základě směrovacích informací z hlavičky. Dále tato komponenta na každém z výstupů odstraní jedno ze slov obsahující identifikaci pravidel:

- L3 výstup – výstupů je určen pro pakety propuštěny na základě L3 pravila a z hlavičky je odstraněno slovo obsahující identifikaci L7 pravidla.
- L7 výstup – určen pro pakety, které byly propuštěny na základě L7 pravidla, u těchto paketů je z hlavičky odstraněno slovo obsahující identifikaci L3 pravidla.




Obrázek 6.8: Rozhraní a zapojení výstupních komponent

Množství zdrojů nezbytných pro implementaci výstupních komponent je uvedeno v tabulce 6.5.

| | LUT | FF | BRAM | DSP |
|---|---|---|---|---|
| BLOB_HDR_COMPLETER | 54 | 108 | 0 | 0 |
| SFORK | 7 | 10 | 0 | 0 |

Tabulka 6.5: Zdroje výstupních komponent

## 6.2 Software

Softwarová část implementace byla až na aplikaci obsluhující jednotku přesného času celá převzata a reflektuje výše uvedený návrh.

### 6.2.1 Aplikace pro jednotku přesného času

Implementace aplikace pro řízení jednotky přesného času reflektuje její návrh. Pro přístup k hardwaru je využita knihovna *hwio*. Třemi základními úkoly aplikace jsou výpis informací o hardwarové jednotce, nastavení hardwarové hodnoty časové značky a její korekce.

Aplikace implementuje následující přepínače (pokud není specifikován žádný přepínač, spustí se korekce časové značky):

- C – spustí korekci hardwarové časové značky.
- P – zobrazí informace o hardwarové jednotce (především aktuální hodnotu časové značky a aktuální frekvenci hardwaru).
- D – spustí aplikaci v ladícím režimu (aplikace není spuštěna na pozadí).

- w – zapíše systémový čas do hardwarové jednotky.
- o – korekce je spuštěna pouze jednou (nikoli periodicky).

Jako referenční čas je použit systémový čas (získaný např. pomocí funkce *gettimeofday()*), který je synchronizován pomocí protokolu NTP (Network Time Protocol) nebo pomocí externího RTC modulu.

Pro korekci časové značky je nutné zjistit aktuální hodnotu přírůstku (vyčtením z hardwaru) a časovou značku z hardwaru a aktuální čas softwaru. Následně je opakovaně prováděna korekce a uspání programu na jednu sekundu. Po probuzení programu je tedy nutné zachytit čas ze softwaru a hardwaru. Aktuální softwarový čas je získán funkcí *gettimeofday()*. Bezprostředně po přečtení softwarového času se přečte čas hardwarové komponenty. Pro jeho vyčtení se do řídicího registru hardwarové jednotky (CMD) zapíše funkcí *hwio_comp_write()* příkaz pro zachycení časové značky. Následně je možné funkcí *hwio_comp_read()* vyčítat data z datových registrů. První 32-bitový datový registr obsahuje jednotky sekunda. Po vyčtení následujících dvou datových registrů získáme 64-bitovou hodnotu $X$, která udává zlomky sekund ($X \cdot 2^{-64}$ s). Pro získání nanosekundové hodnoty je nutné $X$ násobit konstantou $10^9/2^{64}$. Po získání těchto časů je možné provést korekci přírůstku dle vzorce 5.4. Nově vypočítaná hodnota je zapsána do hardwaru a program je opět uspán.

Pokud je rozdíl časů tak velký, že ho nelze v krátké době srovnat pouze pomoci změny hodnoty přírůstku, je časová značka zápisem odpovídajícího příkazu zneplatněna. Pomocí čítače nacházejícího se v komponentě (CNT) je změřena aktuální frekvence FPGA a vypočítána hodnota přírůstku. Do komponenty přesného času je zapsána hodnota přírůstku a aktuální časová značka získaná softwarově. Následně je časová značka opět zápisem odpovídajícího příkazu označena jako platná. K velkému rozdílu časových značek dochází při spuštění Mikrosondy.

### 6.2.2 Aplikace pro Pattern Match

Pro konfiguraci jednotky Pattern Match jsou vytvořeny dvě aplikace. První z nich (*pmgen*) slouží k vygenerování konfigurace ze vstupního souboru, který obsahuje L7 identifikátory zadané regulárním výrazem (např. e-maily) a jsou odděleny lomítkem. Aplikace vzory načte, převede je na NKA, který je zredukován. Následně je vygenerována konfigurace obdobně, jak bylo uvedeno v návrhu. Výstupem je binární soubor obsahující novou konfiguraci.

Druhá aplikace (*pmctl*) slouží pro nahrání vygenerované konfigurace a řízení jednotky. Aplikace implementuje následující přepínače:

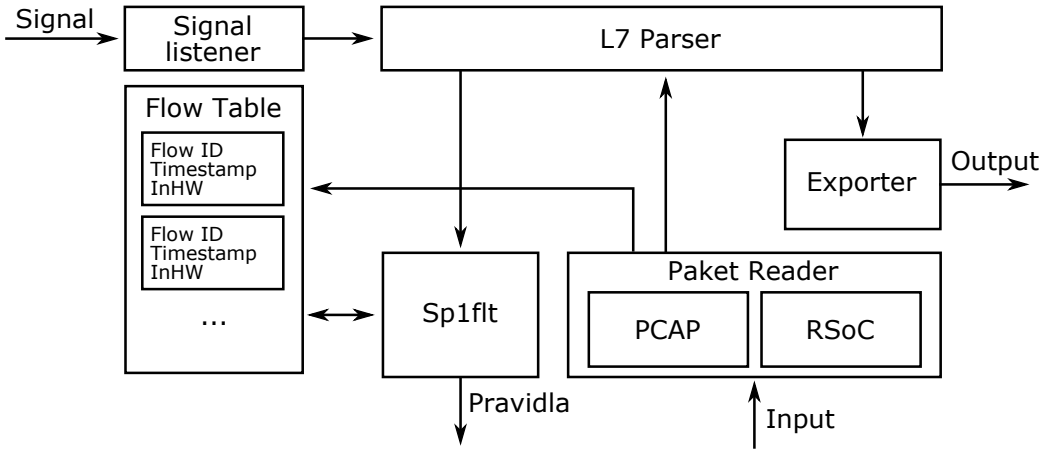
- C <file> – slouží pro nahrání konfigurace ze souboru *file* a přepnutí na tuto konfiguraci.
- D – povolí ladící výpisy.
- P – zobrazí statistiky a informace o jednotce.
- R – resetuje jednotku.
- S – přepne konfiguraci jednotky.
- Z – vynuluje statistické čítače.

### 6.2.3 Aplikace CCCID

Implementace nového návrhu aplikace CCCID rozšiřuje stávající o možnost zadávání L7 identifikátorů. Tyto identifikátory jsou předány z webového uživatelského rozhraní prostřednictvím unixového soketu. Aplikace CCCID tyto identifikátory přijme a uloží do souboru. Jednotlivé identifikátory a jejich vzory jsou v souboru od sebe odděleny lomítkem (`vzor1/vzor2/vzor3`). Soubor je po vytvoření předán rozhraní pro moduly, kterému je současně zaslán signál (upozornění na novou konfiguraci). Dále je spuštěna aplikace *pmgen*, kterou je ze souboru obsahující identifikátory vygenerována konfigurace pro PM. Následně je tato konfigurace nahrána do jednotky aplikací *pmctl*.

### 6.2.4 Rozhraní pro moduly

Úkolem rozhraní pro moduly je spravovat načítání a předávání paketů z hardwaru, starat se o hardwarovou tabulku toků, exportovat zájmový provoz a spravovat konfigurace.




Obrázek 6.9: Struktura rozhraní pro moduly

Pakety jsou z hardwaru do systému předávány prostřednictvím rozhraní PCAP nebo RSoC Framework. Pro předávání paketů do systému je implementovaná třída Packet Reader, která odstiňuje od konkrétního rozhraní. Z předaného paketu jsou extrahovány informace nutné pro identifikaci síťového toku (zdrojová a cílová IP adresa, zdrojový a cílový TCP/UDP port, protokol). Pomocí těchto informací jsou aktualizovány informace o odpovídajícím síťovém toku v softwarové tabulce toků (Flow Table). Načtený paket je následně předán modulu (L7 Parser) ke zpracování.

Velmi důležitou součástí rozhraní pro moduly je Flow Table. Tato tabulka je implementována jako seznam struktur (Flow Data). Struktura Flow Data obsahuje:

- identifikaci toku,
- poslední časovou značku příchodu paketu k tomuto toku,
- příznak toho, zda se záznam nachází v hardwarové tabulce.

Při každém načtení paketu je v seznamu vyhledána položka, která odpovídá tomuto toku, a je aktualizována časová značka pro tento tok. Pokud se v seznamu záznam nenachází, je nově vytvořen. Z INI3 hlavičky paketu je možné podle příznaků FLAGS zjistit, zda je

pravidlo pro tento tok uloženo i v hardwarové tabulce, nebo při jeho vkládání došlo k chybě a nebylo vloženo. Pokud je záznam označen, že se nenachází v hardwarové tabulce, je nutné ho do tabulky vložit softwarově (po uvolnění místa).

Data z Flow Table využívá správce hardwarové tabulky toků (Sp1flt). Pokud je softwarová tabulka zaplněna nad předem daný limit, jsou postupně z obou tabulek (jak hardwarové tak softwarové) odstraňovány záznamy s nejstarší časovou značkou. Po uvolnění tabulek toků jsou do hardwarové tabulky vloženy záznamy ze softwarové tabulky, které obsahují příznak, že se nenachází v hardwarové tabulce. Síťový tok je z tabulek také odstraněn, pokud k danému toku již delší dobu nepřišel žádný paket (vypršel timeout a tok byl označen za ukončený). Správce dále umožňuje modulům přidávat a odstraňovat pravidla pro toky. To modulům umožňuje odstranit síťový tok, který byl ukončen (například ukončením TCP spojení).

Exporter přijímá exportovaná data od modulů (např. zájmový provoz) a odesílá je na požadované místo. Implementovaný Exporter odesílá data do sběrného místa pomocí TCP spojení. Pokud není možné z nějakého důvodu data odesílat (např. výpadek spojení) jsou data dočasně ukládána do vyrovnávací paměti.

Pro identifikaci skutečnosti, že došlo ke změně konfigurace, očekává Signal listener, který je součástí rozhraní pro moduly, příjem signálu. Po jeho přijetí je volána metoda modulu, které zajistí, že si modul načte novou konfiguraci (např. ze souboru).

### 6.2.5 Modul pro SMTP

Jedním z implementovaných modulů je modul pro analýzu SMTP provozu. Tento modul má za úkol rozhodnout, zda dané pakety přenáší SMTP komunikaci konkrétního uživatele identifikovaného e-mailovou adresou.

Seznam e-mailových adres je modulu předán pomocí textového souboru. Modul tyto e-mailové adresy načte a uloží je do řetězce představující regulární výraz:

$$\text{e-mails} = (\text{e-mail}_1 \mid \text{e-mail}_2 \mid \text{e-mail}_3). \tag{6.1}$$

V prvním exportovaném paketu toku se nachází L7 identifikátor (e-mailová adresa), který byl nalezen jednotkou Pattern Match. Po přijetí tohoto paketu je nutné ověřit, že se jedná o SMTP komunikaci. Zadaný identifikátor se mohl tedy nacházet pouze v SMTP příkazu *MAIL FROM* nebo *RCPT TO*. Pokud by se identifikátor nacházel například v těle e-mailu nebo na webové stránce, není tento výskyt akceptován.

V implementaci je tedy sestaven regulární výraz, kterým je ověřováno, zda tento paket začíná jedním z uvedených příkazů a je následován jednou z e-mailových adres:

$$regex = (\text{MAIL FROM} \mid \text{RCPT TO}) + \text{emails} \tag{6.2}$$

Pokud první paket vyhoví sestavenému regulárnímu výrazu, je tento tok exportován. V následujících paketech je vyhledáván pouze příznak ukončení TCP spojení (SYN paket). Po jeho obdržení je tok odstraněn z tabulek pravidel. Pokud není v prvním paketu regulární výraz nalezen, je odpovídající tok odstraněn z tabulek toků.

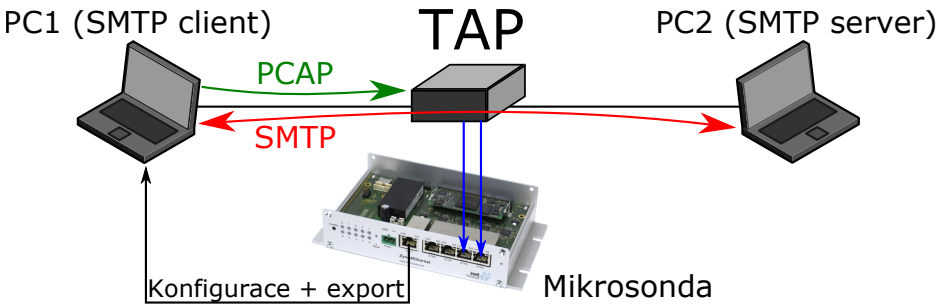
# Kapitola 7

# Testování

Mikrosonda s popsanými úpravami byla testována ve dvou etapách. V první byla ozkoušena funkcionalita na testovací síti, kde se testovala správná funkcionalita nové implementace Mikrosondy. Ve druhé pak byly otestovány limity této implementace.

## 7.1 Testovací síť

Pro otestování správné funkčnosti nové implementace Mikrosondy byla vytvořena testovací síť. Ta se skládá z dvojice počítačů ($PC_1$ a $PC_2$) síťového zařízení TAP a Mikrosondy. $PC_1$ obsahuje dva Ethernetové konektory. Pomocí jednoho konektoru je $PC_1$ připojen k $PC_2$ přes síťový TAP. Druhý konektor slouží pro administraci a export dat z Mikrosondy. Síťový TAP kopíruje komunikaci mezi $PC_1$ a $PC_2$ a odesílá ji prostřednictvím monitorovacích portů do Mikrosondy ke zpracování. Topologie testovací sítě je naznačena na obrázku 7.1.




Obrázek 7.1: Topologie testovací sítě

Na této testovací síti probíhaly dva testy. Při prvním testu bylo na $PC_1$ uloženo několik vzorků reálného provozu v souborech formátu PCAP (Packet Capture), které obsahovaly SMTP komunikaci. Mikrosonda byla nakonfigurována prostřednictvím GUI na odposlech e-mailové adresy (např. *alice@foo.com*). Na $PC_1$ nástroj Wireshark přijímal a zobrazoval data exportovaná Mikrosondou. Po konfiguraci byl nástrojem `tcpreplay` vysílán PCAP soubor do sítě:

```
sudo tcpreplay -i eth3 -t smtp.pcap
```

Mikrosonda tato data zachytila, klasifikovala a exportovala tok, který obsahoval SMTP komunikaci zvoleného uživatele (např. mezi uživateli *alice@foo.com* a *bob@foo.com*). Kont-

rola exportovaných dat v nástroji Wireshark ukázala, že Mikrosonda exportovala všechnu SMTP komunikaci.

Pro druhý test byl na $PC_2$ nainstalován SMTP server a na $PC_1$ nakonfigurován e-mailový klient. Mikrosonda opět odposlouchávala konkrétní e-mailovou adresu (např. *test@myserver.net*). Samotný test pak spočíval v tom, že se z e-mailového klienta na $PC_1$ odesílaly e-mailové zprávy na $PC_2$, kde běžel SMTP server.

Tento test ukázal, že Mikrosonda zachytí SMTP komunikace, kterou odeslal klient z odposlouchávané e-mailové adresy nebo na odposlouchávanou e-mailovou adresu. Testy dále ověřily, že e-maily, které nebyly odeslány ani určeny pro odposlouchávanou e-mailovou adresu nebo obsahovaly odposlouchávanou e-mailovou adresu pouze v těle, Mikrosonda neexportovala.

## 7.2 Test limitů

Tato kapitola popisuje testy rychlosti přenosu mezi hardwarem a softwarem, které jsou zprostředkovaný RSoC Frameworkem, rychlosti zpracovávání jednoho toku a počet zpracovaných toků modulu SMTP.

Pro test rychlosti přenosu dat z hardwaru do softwaru se využil paketový generátor Spirent. Tento generátor generoval na oba odposlechové porty Mikrosondy pakety maximální rychlostí linky. Mikrosonda se nacházela v režimu *Packet capture*, ve kterém do softwaru propouštěla všechny přijaté pakety. Pro měření rychlosti byla využita aplikace *rsocdrv-read*, který se nachází v RSoC Frameworku. Tato aplikace vyčítá data z hardwaru a měří průměrnou rychlost tohoto přenosu. Pro měření byl využit příkaz:

```
rsocdrv-read /dev/rsoc-acc.0 -l 16384 -m > /dev/null
```

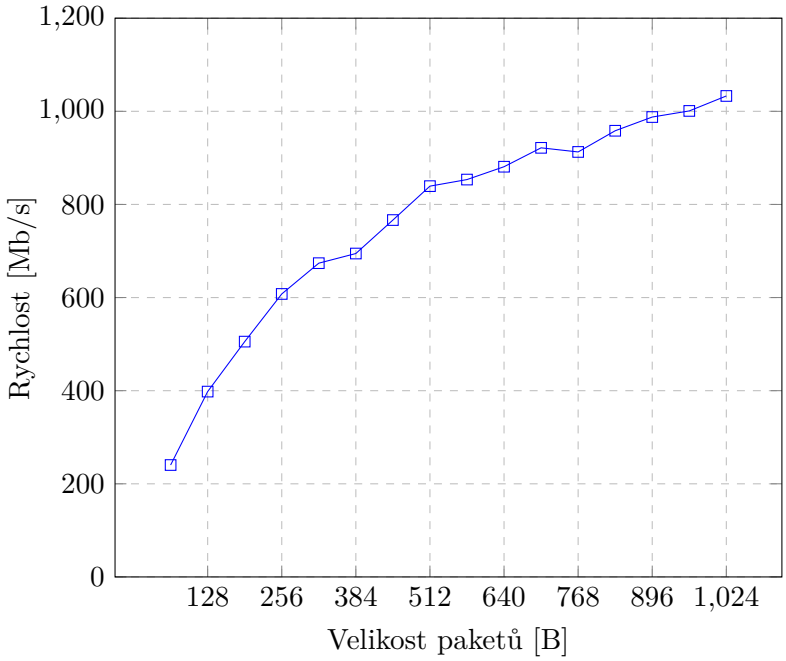
V tomto testu paketový generátor vysílal postupně pakety o velikostech 64–1024 B (s krokem 64). Výsledky měření tohoto testu uvádí grafy 7.2 a 7.3. Z grafů je patrné, že rychlost přenosu dat se se zvětšující velikostí paketů zvětšuje, počet přenesených paketů však klesá. Nejvyšší rychlosti 1 033 Mb/s dosáhl přenos paketů o velikosti 1 024 B. Pro stejnou velikost paketů pak RSoC Framework přenesl nejmenší počet paketů (126 960).

Další test měřil maximální počet zpracovaných paketů modulem pro SMTP. U této varianty měření paketovým generátorem vytvářel pouze jeden síťový tok. Regulárního výrazu se tedy vyhledával pouze v prvním paket. U ostatních se pouze hledal příznak ukončení TCP spojení.

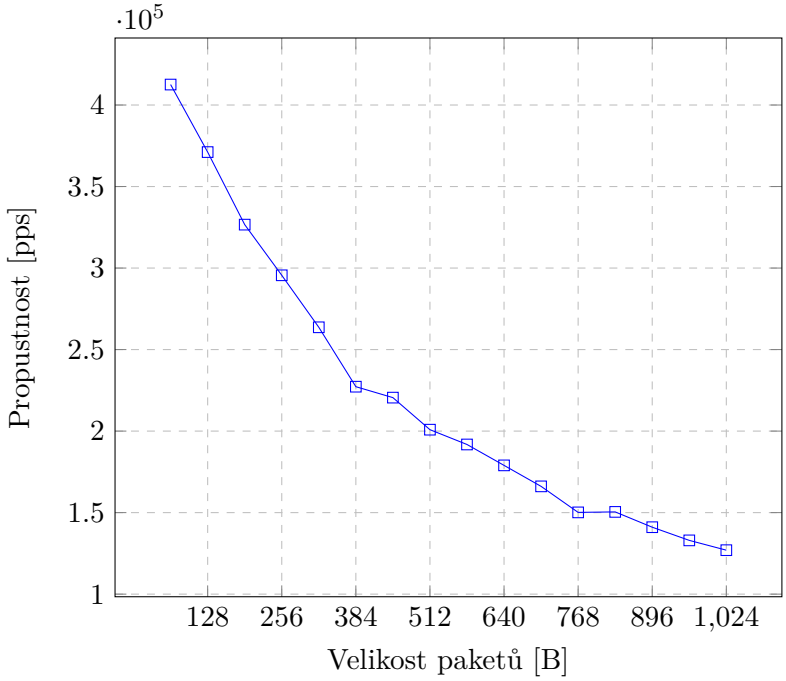
U tohoto testu Mikrosonda do softwaru odesílala všechny přijaté pakety, přestože se v nich nenachází hledaný vzor. Pro měření počtu zpracovaných paketů se každou sekundu vyčetly statistiky filtru. Takto naměřené hodnoty byly zaneseny do grafu 7.4. Tento graf ukazuje, že počet zpracovaných paketů mírně klesá s rostoucí velikostí paketů.

Poslední z testů se snažil zjistit, kolik síťových toků je schopen SMTP modul zpracovat. Tento test se velmi podobá předchozímu testu. Oproti němu však bylo nutné generovat pakety tak, aby každý z nich patřil k jinému síťovému toku. Aby každý paket patřil k jinému síťovému toku, paketový generátor pro každý paket náhodně generuje zdrojovou a cílovou IP adresu a zdrojový a cílový TCP port. Vyhledávání regulárního výrazu nyní probíhalo již v každém paketu. V tomto testu generátor opět vytvářel pakety o různých velikostech. Aby mohly pakety obsahovat i TCP hlavičku (kvůli TCP portům), byla velikost paketu generována až od velikosti 128 B. Graf 7.5 zobrazuje naměřené hodnoty. Z tohoto grafu je patrné, že s velikostí paketů značně klesá počet zpracovaných paketů.

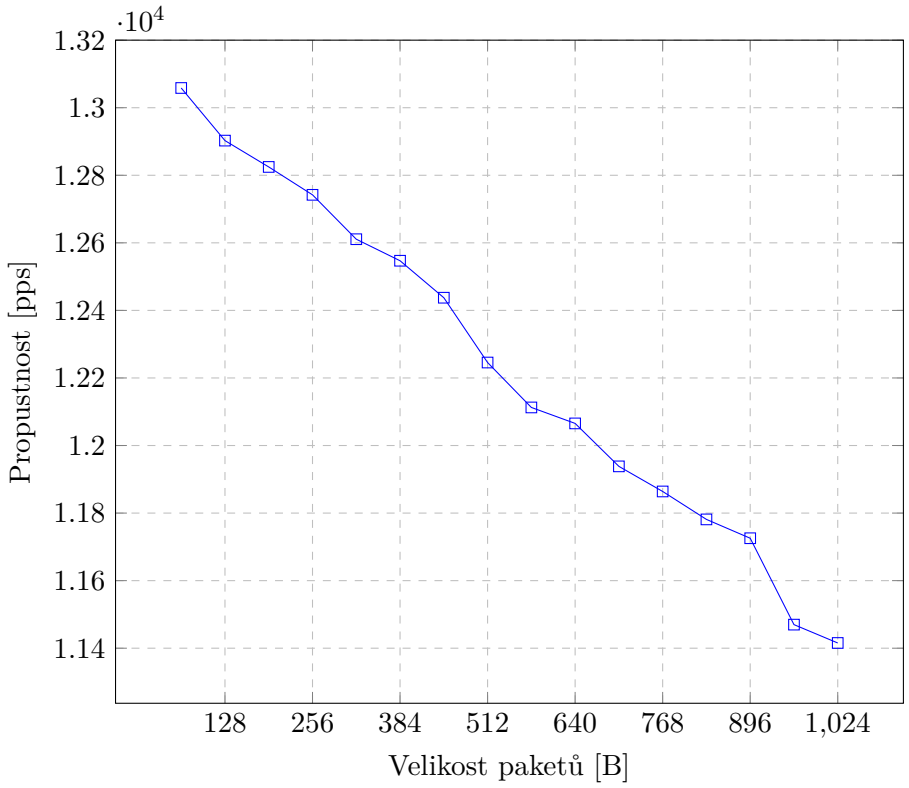
Grafy taktéž ukazují, že rychlost analýzy paketů nelimitoval přenos dat z hardwaru do softwaru ale softwarový modul pro SMTP. Počet přenesených paketů pro danou velikost paketů byl značně vyšší než počet zpracovaných paketů. Modul nejvíce zpomalil příjem paketu nového toku, což způsobuje vyhledávání vzoru.



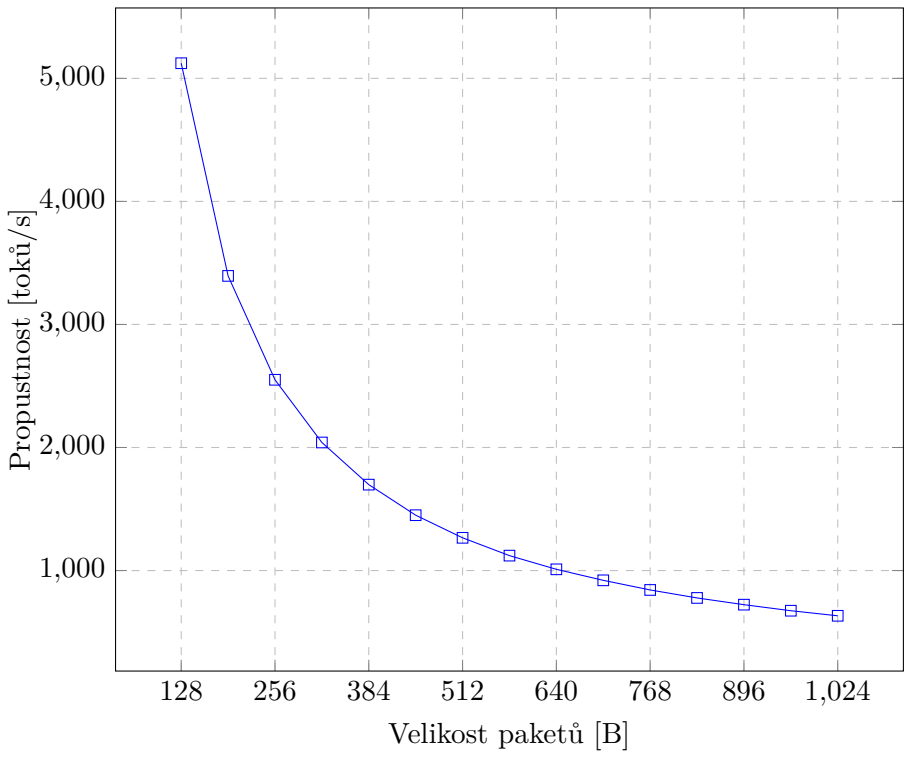

Obrázek 7.2: Rychlost přenosu dat z HW do SW RSoC Frameworkem




Obrázek 7.3: Počet přenesených paketů z HW do SW RSoC Frameworkem




Obrázek 7.4: Rychlost zpracování jednoho toku modulem SMTP




Obrázek 7.5: Rychlost zpracování toků modulem SMTP

# Kapitola 8

# Návrh rozšíření

Nová implementace Mikrosondy obsahuje několik nedostatků. Prvním z nich je skutečnost, že se síťový tok začne exportovat až v momentě, kdy se nalezne v některém z paketů odpovídající vzor (např. konkrétní e-mailová adresa). Pokud se vzor nenachází v prvním paketu, přijdeme o několik paketů, které by mohly být užitečné. Příkladem může být protokol SMTP. Pokud je hledaný e-mail nalezen ve specifikaci příjemce zprávy (v příkazu RCPT TO), dochází ke ztrátě paketu, který obsahuje identifikaci odesílatele, případně identifikaci dalších příjemců. Tyto informace se sice nachází i v hlavičce těla zprávy, ale nemusí být shodné.

Dalším nedostatkem je fakt, že při zahlcení softwarového modulu může docházet k zahazování paketů. V těchto situacích však Mikrosonda nenabízí informaci o počtu zahozených paketů. Tato informace je důležitou metrikou pro posouzení kvality odposlechu.

V rámci této práce byl tedy vytvořen návrh rozšíření hardwaru a softwaru, který řeší oba tyto problémy.

## 8.1 Rozšíření firmwaru

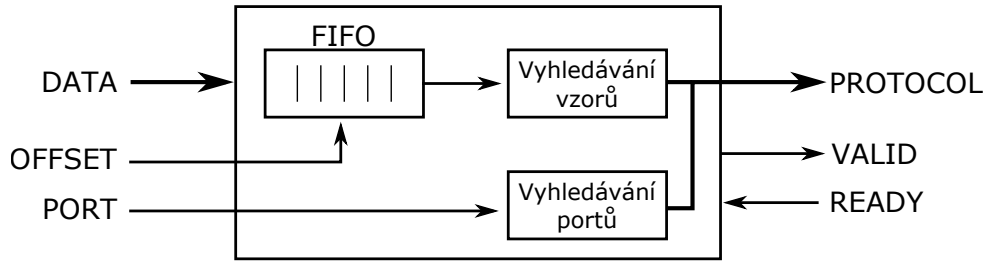
Návrh rozšíření firmwaru se zaměřuje na uvedené nedostatky, které se snaží minimalizovat. Řešení těchto problémů vyžaduje provést úpravy v návrhu, které popisují následující kapitoly.

### 8.1.1 Zachycení celého toku

Pro zachycení prvních paketů síťového toku je nutné do návrhu firmwaru vložit novou komponentu Protocol Identifier (PI). Návrh této komponenty se téměř shoduje s komponentou Pattern Match (PM), oproti které není požadována konfigurovatelná za běhu. Tato komponenta rozpoznává protokol podle jeho vzoru či podle standardního TCP/UDP portu. Aby byla detekce protokolu co nejpřesnější, komponenta hledá vzory pouze v aplikačních datech nikoli v hlavičkách, jak je tomu u PM. Pro tento účel je nutné z klasifikační jednotky do PI předat informaci o tom, kde začínají aplikační data (offset) a čísla TCP/UDP protokolů.

Na vstupu PI se nachází vyrovnávací fronta, která umožňuje přeskočit libovolný počet položek v jednom taktu. Po obdržení offsetu od klasifikační jednotky fronta přeskočí na aplikační data (dáno offsetem). V aplikačních datech se vyhledávají vzory protokolů.

V komponentě PI se následně nachází tabulka portů, které jsou standardně používány pro požadované protokoly. Pokud je v datech nalezen vzor některého z protokolů nebo komunikace probíhala na některém ze standardních portů služeb, je o této skutečnosti informována filtrační jednotka. Do filtru se předává identifikace nalezeného protokolu. Návrh architektury této komponenty ne naznačen na obrázku 8.1.



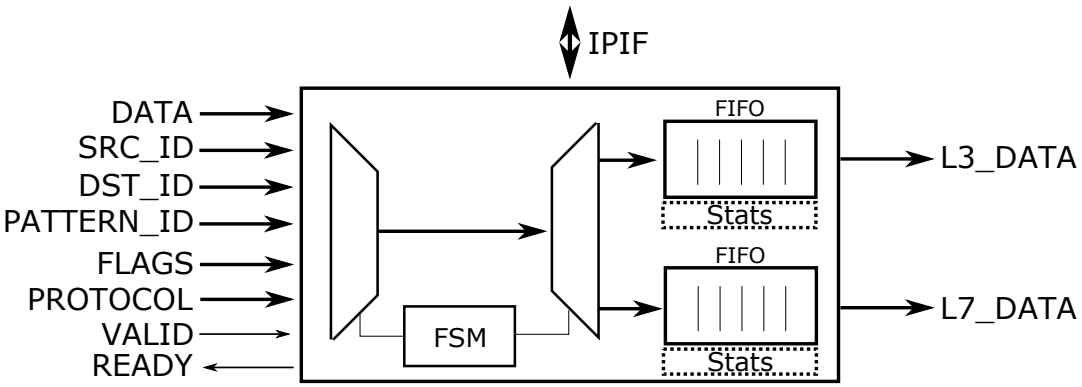

Obrázek 8.1: Návrh komponenty Protocol Identifier

Protože síťový tok, který přenáší požadovaný protokol není s velkou pravděpodobností zájmový, mohlo by se do softwaru exportovat velké množství zbytečných dat. Z tohoto důvodu je nutné do softwaru exportovat jen prvních několik paketů. Pro tuto funkcionalitu bude nutné dále přizpůsobit Flow Cache.

Flow Cache kromě přidání a odebrání pravidla pro konkrétní tok musí umožňovat přidání pravidla, které je platné jen pro prvních $N$ paketů. V tabulce se u každého pravidla nachází čítač, který značí počet paketů, které je možno propustit dále. Filtrační jednotka na svém výstupu pak musí obsahovat další signál, kterým je identifikován nalezený protokol.

### 8.1.2 Kontrola zahlcení

Při zahlcení softwarových modulů musí firmware zahazovat příchozí pakety. Pro kontrolu zahazování paketů návrh upravuje výstupní část. Ta kromě komponent, které zajistí vytvoření hlavičky správného tvaru před každý paket, dále obsahuje speciální vyrovnávací fronty (FIFO), které při zaplnění dokáží zahazovat celé pakety a informovat o této skutečnosti. U každé fronty se pak nachází množina čítačů, které informují o počtu zahozených a propuštěných paketů.



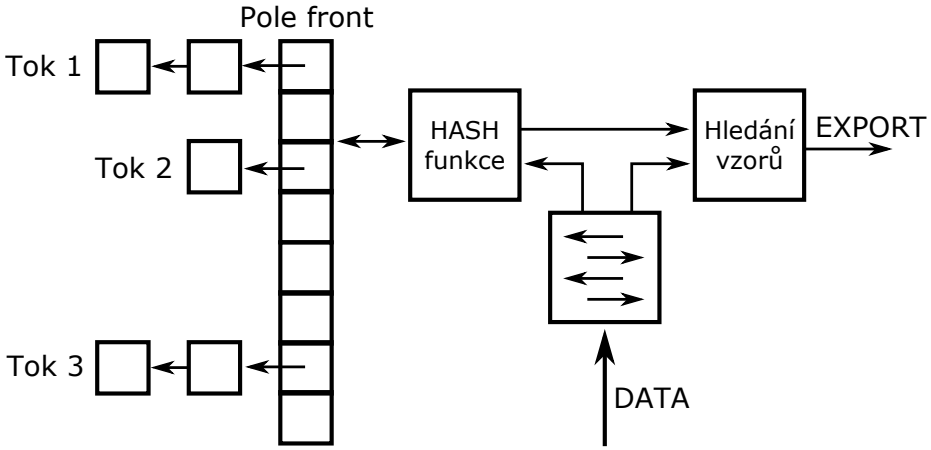

Obrázek 8.2: Návrh výstupní komponenty

## 8.2 Rozšíření softwaru

Tento návrh upravuje software tak, aby bylo možné rozpoznat pakety, které představují pouze prvních několik paketů toku potenciálně zajímavého provozu. Hardwarová komponenta pro export pro tento účel do INI3 hlavičky vkládá další příznak. Takto označené pakety nejsou určeny pro okamžité zpracování ale pro dočasné uložení.

Pro uložení těchto paketů vytvořený návrh obsahuje navíc pole front. Toto pole obsahuje fixní počet front, výběr fronty pro tok je tedy prováděn hašovací funkcí. V případě konfliktu se tok uloží do následující prázdné fronty. Každý síťový tok je ukládán do jiné fronty. Toky jsou ve frontě drženy pouze po omezenou dobu. Pro tento účel obsahují jednotlivé fronty časové značky udávající čas posledního vložení paketu. Po vypršení časového limitu jsou všechny pakety z fronty vymazány.

Pokud je do softwarového modulu exportován síťový tok na základě L7 pravidla a modul potvrdí, že jde o zájmový provoz, je nutné nejdříve zkontrolovat, zda se ve frontách nenachází pakety související s tímto tokem. Pokud tomu tak je, jsou nejdříve exportovány pakety z odpovídající fronty. Struktura softwaru je naznačena na obrázku 8.3, kde jsou fronty realizovány pomocí vázaného seznamu.




Obrázek 8.3: Návrh rozšíření softwaru

Počet paketů, které jsou exportovány do softwaru na základě komponenty PI, je třeba vhodně zvolit. Pokud se L7 identifikátor, který vyhledává jednotka Pattern Match, nachází příliš daleko od začátku toku, nemusí být exportovány všechny pakety tohoto toku. Pakety, které se nachází za prvními $N$ pakety toku (exportované na základě komponenty PI) a současně před paketem, který obsahuje hledaný vzor, jsou zahozeny. Počet nesmí být příliš velký, aby nedošlo k zahlcení softwarového modulu nebo zaplnění softwarových front. Pokud by byl počet příliš malý, často bude nastávat výše zmíněná situace.

# Kapitola 9

# Závěr

Úkolem této práce bylo seznámit se s aplikací Mikrosonda vyvíjenou v projektu *Moderní prostředky pro boj s kybernetickou kriminalitou na Internetu nové generace*, s platformou NetModule ZE7000 a vybranou množinou aplikačních protokolů a dále navrhnout a implementovat vhodné úpravy Mikrosondy pro monitorování aplikačních protokolů.

První část práce se zabývá studiem aplikačních protokolů. Pro studium jsou vybrány protokoly, které zastupují vždy jednu z nejpoužívanějších služeb síťové komunikace. Teoretická část se zaměřuje na protokoly SMTP (protokol pro zasílání elektronické pošty), POP3 (protokol pro vyzvedávání elektronické pošty z poštovního serveru), FTP (protokol pro přenos souborů) a SIP (protokol pro signalizaci užívaný v internetové telefonii). U každého protokolu jsou prozkoumány možnosti jeho detekce.

Druhá část se zaměřuje na studium hardwarových a softwarových součástí aplikace Mikrosonda. Jelikož byla aplikace představena již v mé bakalářské práci, popis se zaměřuje především na změny provedené oproti popsanému výstupu.

Práce dále představuje již existující řešení koncept Software Defined Monitoring (SDM). Tento koncept rozděluje funkcionalitu monitorování síťového provozu mezi hardware a software. Z tohoto konceptu jsou použity základní myšlenky pro návrh nové architektury.

Řešení se inspiruje konceptem SDM, avšak z důvodu nedostačujícího výkonu použitého procesoru je rozšířena FPGA část systému. FPGA firmware vyhledává konkrétní aplikační identifikátory nebo vzory specifické pro daný aplikační protokol. Do softwaru jsou exportována pouze potenciálně zájmová data. Díky tomu se z FPGA firmwaru exportuje jen zlomek síťového provozu. Softwarové aplikace následně ověřují, zda se opravdu jedná o zájmová data.

Pro takto navržené a implementované řešení byla vytvořena testovací síť, na které se testovala správná funkcionalita nové implementace Mikrosondy. Testovací síť sestává ze SMTP klienta a serveru. Vzájemnou komunikaci Mikrosonda monitoruje a exportuje jen požadovaná data. Testy prokázaly, že Mikrosonda exportuje právě zájmovou komunikaci.

Další testy se zaměřily na měření limitů nové implementace. Výsledky měření ukázaly, že je nová implementace schopna exportovat z FPGA firmwaru do softwarové části více než 400 000 paketů za sekundu a SMTP modul zpracuje více než 13 000 paketů a 5 000 síťových toků za sekundu.

V závěru práce jsou analyzovány nedostatky nové implementace. Implementace není schopna především zachytit celý síťový tok. Pro tento případ lze minimalizovat navrženým rozšířením, které spočívá v detekci konkrétních aplikačních protokolů v FPGA a v dočasném uložení odpovídajících síťových toků. Uložené toky mohou být později vyhodnoceny jako zájmové a předány pro další zpracování.

# Literatura


[1] *RFC 791 Internet Protocol* [online]. Září 1981 [cit. 2015-12-30].
URL http://tools.ietf.org/html/rfc791

[2] *RFC 793 Transmission Control Protocol* [online]. Září 1981 [cit. 2015-12-31].
URL http://tools.ietf.org/html/rfc793

[3] *TCP/IP* [online]. Listopad 2015 [cit. 2015-12-30].
URL https://cs.wikipedia.org/wiki/TCP/IP

[4] *Moderní prostředky pro boj s kybernetickou kriminalitou na Internetu nové generace* [online]. 2015 [cit. 2016-01-04].
URL http://www.fit.vutbr.cz/research/grants/index.php.cs?id=517

[5] ARM: *AMBA 4 AXI4-Stream Protocol Specification* [online]. 2010 [cit. 2016-05-20], ID030510.
URL https://silver.arm.com/download/download.tm?pv=1074010

[6] Byrne, J.: *ARM Outmuscles Atom on Benchmark* [online]. Duben 2011 [cit. 2016-05-01].
URL http://parisbocek.typepad.com/blog/2011/04/arm-outmuscles-atom-on-benchmark-1.html

[7] Cesnet: *SIP [online]*. 2012 [cit. 2016-01-04].
URL https://sip.cesnet.cz/cs/protokoly/sip

[8] Deering, S.; Hinden, R.: *RFC 2460 InternetProtocol, Version 6 (IPv6) Specification* [online]. Prosinec 1998 [cit. 2015-12-31].
URL http://tools.ietf.org/html/rfc2460

[9] Dražil, J.: *Zynq testy propustnosti* [online]. Září 2014 [cit. 2016-05-19].
URL https://merlin.fit.vutbr.cz/wiki-sec6net/index.php/Zynq_testy_propustnosti

[10] Enclustera: *Xilinx-based Mars Module Products and Roadmap* [online]. Únor 2015 [cit. 2016-05-01].
URL http://www.enclustra.com/assets/files/products/fpga_modules/XilinxBasedMarsModules_SelectionGuideAndRoadmap.pdf

[11] Fukač, T.: *Mikrosonda na platformě Xilinx Zynq*. Bakalářská práce, Vysoké učení technické v Brně, 2014.

[12] Handley, M.; Jacobson, V.; Perkins, C.: *RFC 3261 SIP: Session Initiation Protocol* [online]. Červenec 2006 [cit. 2016-01-10].
URL https://tools.ietf.org/html/rfc4566

[13] ITU: *Information Technology – Open System Interconnection – Basic Reference Model: The Basic Model* [online]. 1994 [cit. 2015-12-26].
URL http://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-X.200-199407-I!!PDF-E&type=items

[14] Kabelová, A.; Dostálek, L.: *Velký průvodce protokoly TCP/IP a systémem DNS*. 1. dotisk 5. aktualizované vydání, Computer Press Brno, 2012 [cit. 2016-01-09], ISBN 978-80-251-2236-5, 488 s.

[15] Kekely, L.; Puš, V.; Kořenek, J.: Software Defined Monitoring of Application Protocols. In *IEEE INFOCOM 2014 - IEEE Conference on Computer Communications*, 2014 [cit. 2016-05-01].
URL http://www.fit.vutbr.cz/research/view_pub.php?file=%2Fpub%2F10657%2F1569805973.pdf&id=10657

[16] Myers, J.; Rose, M.: *RFC 1939 Post Office Protocol - Version 3* [online]. Květen 1996 [cit. 2016-01-02].
URL https://www.ietf.org/rfc/rfc1939.txt

[17] Pavol, K.; Tomáš, F.; Jan, V.; aj.: *Popis Prototypu mikrosondy (uSondy) pro monitorování IPv6 provozu* [online]. 2015 [cit. 2016-01-04].
URL http://www.fit.vutbr.cz/research/view_product.php?file=%2Fproduct%2F428%2FPopis+Prototypu+mikrosondy.pdf&id=428

[18] Pavol, K.; Tomáš, F.; Jan, V.; aj.: *Uživatelská dokumentace k Prototypu mikrosondy (uSondy) pro monitorování IPv6 provozu* [online]. 2015 [cit. 2016-01-04].
URL http://www.fit.vutbr.cz/research/view_product.php?file=%2Fproduct%2F428%2FUzivatelska+dokumentace+k+Prototypu+mikrosondy.pdf&id=428

[19] Postel, J.: *RFC 768 User Datagram Protocol* [online]. Srpen 1980 [cit. 2015-12-31].
URL http://tools.ietf.org/html/rfc2460

[20] Postel, J.; Reynoldse, J.: *RFC 959 File Transfer Protocol (FTP)* [online]. Říjen 1985 [cit. 2016-01-09].
URL https://www.ietf.org/rfc/rfc959.txt

[21] Postel, J. B.: *RFC 821 Simple Mail Transfer Protocol* [online]. Srpen 1982 [cit. 2016-01-01].
URL https://tools.ietf.org/html/rfc821

[22] Ramakrishnan, K.; Floyd, S.; Black, D.: *RFC 3168 The Addition of Explicit Congestion Notification (ECN) to IP* [online]. Září 2001 [cit. 2015-12-31].
URL http://tools.ietf.org/html/rfc3168

[23] RehiveTech: *RSoC Framework* [online]. 2016 [cit. 2016-05-01].
URL http://rsoc-framework.com/description/

[24] Riabov, V. V.; College, R.: *SMTP (Simple Mail Transfer Protocol)* [online]. Květen 2005 [cit. 2016-01-01].
URL https://www.rivier.edu/faculty/vriabov/Information-Security-SMTP_c60_p01-23.pdf

[25] Rosenberg, J.; Schulzrinne, H.; Camarillo, G.; aj.: *RFC 3261 SIP: Session Initiation Protocol* [online]. Červen 2002 [cit. 2016-01-10].
URL https://www.ietf.org/rfc/rfc3261.txt

[26] Soumar, M.: *Signalizační protokol pro přenos hlasu přes datové sítě - SIP [online].* 2003 [cit. 2016-01-04].
URL http://www.elektrorevue.cz/clanky/03003/index.html

[27] Takaguchi, H.; Wakaba, Y.; Wakabayashi, S.; aj.: AN NFA-BASED PROGRAMMABLE REGULAR EXPRESSION MATCHING ENGINE HIGHLY SUITABLE FOR FPGA IMPLEMENTATION. In *SASIMI 2013 Proceedings*, 2013 [cit. 2016-05-03].
URL http://sasimi.jp/new/sasimi2013/files/pdf/p231_R4-5.pdf

[28] Vozňák, M.: Signalizace SIP. In *Teorie a praxe IP telefonie - 2. dvoudenní odborný seminář*, Listopad 2006 [cit. 2016-01-10], str. 41.
URL http://www.phonet.cz/archiv/dok_osta/ipt-2006_Signalizace_SIP.pdf

[29] Xilinx: *LogiCORE IP AXI4-Lite IPIF v2.0* [online]. Prosinec 2013 [cit. 2016-05-20].
URL http://www.xilinx.com/support/documentation/ip_documentation/axi_lite_ipif/v2_0/pg155-axi-lite-ipif.pdf

[30] Xilinx: *Zynq-7000 All Programmable SoC Overview* [online]. Leden 2016 [cit. 2016-05-01].
URL http://www.xilinx.com/support/documentation/data_sheets/ds190-Zynq-7000-Overview.pdf

# Přílohy

# Příloha A

# Obsah CD

CD obsahuje následující adresáře:

- **vivado** – projekt pro vývojový nástroj Vivado
- **buildroot-1g** – balíčky a rozšíření pro Buildroot
- **sprobe** – zdrojové soubory pro *usonda_core*
- **binary** – binární soubory
- **DP** – zdrojový text pro tuto práci

Pro vytvoření firmwaru je nutné provést následující kroky:

1. stáhnout a nainstalovat systém `fpgabuild` (více informací u vedoucího práce)
2. přejít do adresáře `sprobe/uprobe/src/core/`
3. příkazem `fpgabuild synth` provést syntézu *usonda_core*
4. spustit nástroj Vivado, otevřít projekt a nastavit cesty ke zdrojovým souborů IP komponent
5. vygenerovat blok design a spustit syntézu
6. zkopírovat soubor `vivado/phase_7.runs/impl_1/design_1_wrapper.bit` do `buildroot-1g/board/netmodule/ze7000/ze7000_fpga.bit`

Překlad softwaru a příprava obsahu SD karty je možné provést následujícími příkazy:

```
$ git clone git://git.buildroot.net/buildroot --branch 2015.11.1
$ mkdir build
$ cd build
$ make BR2_EXTERNAL="$PWD/../buildroot-1g" O="$PWD" \
       -C "$PWD/../buildroot" ze7000_defconfig
$ make
```

Těmito příkazy je provedeno:

1. stažení nástroje Buildroot
2. vytvoření složky, kde bude probíhat překlad
3. přechod do adresáře pro překlad
4. konfigurace nástroje Buildroot
5. překlad

Nyní stačí zkopírovat soubory z adresáře `build/images/sd/` na SD kartu Mikrosondy.