VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

ANALYZÁTOR KVALITY HOVORŮ VOIP

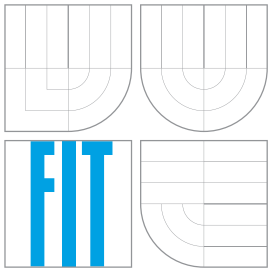
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

MARTIN BASEL

BRNO 2014

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

ANALYZÁTOR KVALITY HOVORŮ VOIP

QUALITY ANALYSIS OF VOIP CALLS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

MARTIN BASEL

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. PETR MATOUŠEK, Ph.D.

BRNO 2014

Abstrakt

Tato bakalářská práce se zabývá metodami pro hodnocení kvality hovorů VoIP. Je zde vysvětlen rozbor protokolu RTP/RTCP a následné počítání statistik potřebných pro určení kvality hovorů. V této práci je také popsána metoda, která byla implementována do pluginů pro sondu FlowMon. Budete zde seznámeni s výsledky a s porovnáním s ostatními programy.

Abstract

This bachelor's thesis focuses on methods for evaluating quality of VoIP calls. You can read about analysis of RTP/RTCP packets and use this knowledge for computing statistics required for assessing quality of VoIP calls. Also, I explain the method, which is implemented in plugins that work in FlowMon probe. Finally, you will see the results compared to the other programs.

Klíčová slova

VoIP, stupnice MOS, R-faktor, kvalita hovorů VoIP, FlowMon, IPFIX

Keywords

VoIP, quality of VoIP calls, MOS scale, R-factor, FlowMon, IPFIX

Citace

Martin Basel: Analyzátor kvality hovorů VoIP, bakalářská práce, Brno, FIT VUT v Brně, 2014

Analyzátor kvality hovorů VoIP

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Petra Matouška, Ph.D. Další informace mi poskytli Ing. Petr Špringl a Mgr. Martin Elich. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Martin Basel
19. května 2014

Poděkování

Rád bych poděkoval Ing. Petru Matouškovi, Ph.D. za vedení mé práce, rady, připomínky a vstřícnost při spolupráci. Dále také děkuji Ing. Petru Špringlovi a Mgr. Martinu Elichovi za rady v implememtační části práce.

Obsah

Kapitola 1

Úvod

Cílem této práce je nastudovat průběh VoIP hovorů, metody hodnocení jejich kvality a za pomoci rozboru síťového provozu najít VoIP hovory a určit jejich kvalitu. Kvalita bude hodnocena pouze na základě síťových parametrů, jelikož jiné faktory (šum, hlasitost, ...) se nachází v jiné oblasti informačních technologií. Jinými slovy, kvalita bude hodnocena objektivně, nezáleží na tom, zdali některý účastník hovoru mluví v příliš hlučném prostředí, příliš potichu nebo nesrozumitelně. Záleží pouze na síťových parametrech, které lze vyčíst při zachytávání paketů.

Výstupem je kromě této práce v textové podobě také sada pluginů do monitorovací sondy FlowMon od firmy INVEA-TECH[1], která dokáže monitorovat síťový provoz. Konkrétně se jedná o sondu obsahující exportér a kolektor IPFIX dat. Pluginy byly vytvořeny v jazyce `C` a jsou určeny pro exportér. Díky těmto pluginům se exportér chová přesně jak potřebujeme, monitoruje VoIP hovory a jejich kvalitu. Naměřené hodnoty předává pomocí protokolu IPFIX na kolektor.

Ve druhé kapitole budete seznámeni s VoIP technologiemi, co je třeba k uskutečnění VoIP hovoru a jak hovor probíhá. Dozvíte se o signalizaci hovorů, nejrozšířenějším signalizačním protokolu SIP a o procesu filtrování VoIP hovorů ze síťové komunikace. Ve třetí kapitole je obsaženo hlavní jádro této práce, čili různé metody hodnocení kvality hovorů, jednotlivé měřené statistiky v této práci a ukazuji také metodu použitou v praktické části. Následující kapitola pojednává o výpočtu jednotlivých statistik z paketů nesoucích audio pro hovor. V páté kapitole je popsána nejen implementace pluginů, ale také architektura celé aplikace. Dozvíte se o spolupráci jednotlivých komponent, komunikaci mezi vlákny uvnitř exportéru a o protokolu IPFIX. V předposlední kapitole je objasněno, jak probíhalo testování naměřených hodnot. Výstup z mého analyzátoru bude srovnán s třemi dalšími programy. V závěrečné kapitole jsou popsány dosažené výsledky a možná vylepšení.

[1] http://www.invea.com/cs

Kapitola 2

VoIP technologie

V této kapitole budete seznámeni s principy technologie VoIP. Nebude zde probrána dopodrobna, jelikož to není hlavním úkolem této práce. Nicméně po přečtení této kapitoly byste měli být seznámeni s tím, jak technologie VoIP pracuje, co je potřeba pro uskutečnění VoIP hovoru, jak se navazuje spojení a na konci kapitoly je vysvětleno, jak jsou znalosti z této kapitoly využity v praktické části implementace.

Technologie VoIP (Voice over Internet Protocol) umožňuje přenášet telefonní hovor po síti (ať už po Internetu nebo po privátní síti). Hovor zpravidla prochází třemi fázemi:

- vytvoření hovoru,
- přenos hlasu,
- ukončení hovoru.

Vytvořením hovoru se rozumí část, kde strany vyjednávají jednotlivé podmínky hovoru, jako jsou čísla portů, kodeky, atd. Dále zahrnuje vyzvánění a potvrzení přijetí hovoru. Vytvoření a ukončení hovoru je zpravidla řízeno stejným protokolem, čili ukončení hovoru opět zahrnuje poslání zprávy oznamující ukončení hovoru a následné potvrzení.

Pro VoIP hovor není třeba hardwarový telefon, lze si vystačit i se softwarovými nástroji (např. Cisco IP Communicator[1] nebo SJphone[2], který byl využit pro vytváření testovacích hovorů).

2.1 Rozdíly mezi VoIP a klasickou telefonní sítí

Pro lepší vysvětlení VoIP technologie se ji pokusím srovnat s klasickou telefonní linkou. Klasická telefonní síť *PSTN* (Public Switched Telephone Network) využívá technologie *přepínání okruhů*, která vyžaduje před zahájením hovoru vytvořit dedikované spojení mezi dvěmi koncovými zařízeními. Hovor využívá plnou kapacitu linky a spojení existuje po celou dobu hovoru. Hlas je přenášen pomocí analogového signálu. Existují tedy zde tři fáze hovoru:

1. vytvoření spojení (zahrnuje test, zdali je volané koncové zařízení dostupné),

[1] http://www.cisco.com/c/en/us/products/collaboration-endpoints/ip-communicator/index.html

[2] http://www.sjphone.org/

2. samotný přenos hovoru,

3. ukončení hovoru (vyvolání příslušné akce jednou stanicí).

Naproti tomu u VoIP se používá technologie *přepínání paketů*, která zajišťuje, že data jsou rozdělena do bloků (paketů), které jsou přenášeny. Hlavní rozdíl oproti PSTN je ten, že VoIP hovory nemají dedikované spojení, čili nemají garantovánu celou šířku pásma. Hlasový signál je převeden do binární podoby a následně rozdělen do paketů. Jinými slovy, VoIP hovory posílají datové toky obsahující data v binární podobě po síti. Každý účastník tedy posílá svůj tok. Pro signalizaci volání se používají signalizační protokoly (nejčastěji SIP nebo H.323), které jsou podrobněji popsány v kapitole 2.4.

2.2 VoIP gateway (brána)

Brána VoIP je termín pro zařízení nebo aplikaci, která spojuje klasickou telefonní síť s VoIP. Pobočková ústředna *PBX* (Private Branch Exchange) je zařízení využívané hlavně ve firemních podnicích, sloužící pro sjednocení telefonů uvnitř sítě a navenek vypadající jako jeden telefon. Spojením PBX a VoIP je cesta pro velké firmy, jak šetřit finanční prostředky. V tomto případě totiž ústředna vidí bránu jako další ústřednu, kde jsou hovory převedeny na VoIP. Při příchozích hovorech jsou na bráně hovory převedeny na analogové hovory a poslány na ústřednu. Nemusí se propojovat pouze PSTN a VoIP, mohou se například propojit VoIP sítě s různými signalizačními protokoly (např. H.323 a SIP). Nejznámější softwarovou bránou je aplikace Asterisk[3].

2.3 Kvalita služeb (Quality of Service)

Aby se VoIP hovory mohly rovnat klasickým telefonním hovorům, je třeba jim zajistit stejnou kvalitu. Jelikož ale PSTN sítě využívají dedikované spojení pro hovor, je tento úkol velice obtížný. *QoS je sada opatření pro zajištění odpovídající kvality toků, v tomto případě hovorů.* QoS poskytuje lepší podmínky pro síťový provoz zajištěním následujících vlastností [3]:

- podpora vyhrazené šířky pásma,
- zamezování ztrátovosti paketů,
- zamezování a řízení zahlcování na síťových zařízeních,
- rozložení provozu,
- nastavování priorit.

Tyto vlastnosti jsou zpravidla nastavovány na směrovačích. V prvé řadě je třeba zajistit dostatečnou šířku pásma. Mějme šířku linky např. 64 kbps, pokud by byl pro hovor použit kodek G.711, který posílá data rychlostí 80 kbps, ztrátovost by byla nejméně 20 %.

Jakmile je zajištěna dostatečná šířka pásma, může být dosažena určitá kvalita. Směrovač na základě jistých mechanismů může určit, zdali se jedná o paket náležící k nějakému VoIP hovoru. Následně tento paket může označit jako důležitý (např. nastavení pole ToS v IP hlavičce). Na každém směrovači může docházet k situacím, kdy příchozích paketů je více,

[3] http://www.asterisk.org/

než je směrovač schopen obsloužit (pomocí směrovacích tabulek musí rozhodovat, kam přepošle daný paket). V důsledku toho vznikají fronty čekajících paketů na obsloužení. QoS mimo jiné zajišťuje, aby pakety náležící VoIP hovorům byly obsluhovány přednostně.

Existuje mnoho typů front, ale společnost Cisco podle článku [3] doporučuje na svých směrovačích použít frontu typu *LLQ (Low Latency Queuing)*. Jedná se o mechanismus, kdy jsou pakety klasifikovány do jistých tříd. Každá třída má svou frontu, neklasifikované pakety mají frontu *default*. Existuje zde také prioritní fronta, ze které se čerpají pakety přednostně. Ovšem je třeba zároveň předcházet jevu, který se nazývá jako *vyhladovění*. Vyhladovění je označení pro situaci, kdy se určitá fronta nedostane na řadu při propouštění svých paketů. Proto jsou pakety v prioritní frontě propouštěny pouze do určitého nastaveného limitu.

Dále může nastat situace, kdy je prioritní fronta pro VoIP hovory prázdná, tudíž se obsluhuje fronta s menší prioritou, kde je nadměrně velký paket (např. 1500 bytů). Pokud je ovšem velikost přenosové linky výrazně nižší (např. 64 kbps), bude trvat posílání tohoto paketu, při daných hodnotách 187,5 ms, což je zpoždění, které je pro VoIP hovor nepřípustné, obzvlášť pokud se jedná pouze o zpoždění na jednom směrovači. Proto se u zajištění QoS používá mechanismus *fragmentace a vkládání paketů*. Větší pakety jsou rozděleny na části, kdy mezi odesíláním jednotlivých částí se kontroluje, zdali nepřišel paket do prioritní fronty.

Dalším mechanismem pro zajištění kvality služeb je *traffic shaping* (rozložení provozu), který zajišťuje, že data jsou posílána v menších shlucích s danou přenosovou rychlostí. Tímto mechanismem se dá předcházet zahlcení, jelikož je regulována šířka přenosového pásma.

2.4 Signalizační protokol SIP

Pro každý VoIP hovor platí, že potřebuje mít ustanovenou relaci, nad kterou se přenáší RTP/RTCP pakety. V současné době se pro vytváření relace používají zejména dva protokoly, SIP a H.323. V této práci jsem se zabýval pouze protokolem SIP, jelikož je novější a stává se čím dál více populárnějším a dnes je již přítomen u většiny VoIP hovorů [17].

SIP (Session Initiation Protocol) je signalizační protokol umožnující sestavení, modifikaci a ukončení relace s jedním nebo více účastníky [17]. SIP pakety se standardně přenáší nad protokolem UDP na portech 5060. Obecně platí, že volající posílá pozvánku k hovoru volanému. Ten může hovor přijmout nebo odmítnout.

Jedná se o textový protokol kódovaný ve znakové sadě UTF-8, který se chová podobně jako například protokol HTTP, kde klient zasílá požadavky na server a server odpovídá klientovi.

2.4.1 Struktura protokolu SIP

Existují dva druhy zpráv se kterými protokol pracuje, *žádosti* (requests) a *odpovědi* (responses). Struktura zprávy je následující:

```
Request-Line / Status-Line
*message-header
CRLF
[message body]
```

Položka `message body` je nepovinná, ovšem první tři položky, včetně `CRLF` (odřádkování), nesmí chybět. První řádek značí, zdali se jedná o paket s *žádostí* nebo *odpovědí*

protokolu SIP. Položka `message-header` obsahuje dodatečné informace k relaci, pro nás je nejvýznamnější informace `Call-ID`, což je jednoznačný identifikátor hovoru. Dále obsahuje například identifikaci volajícího nebo volaného. V nepovinné části `message body` se nachází protokol SDP (více o protokolu SDP v sekci 2.5), pokud je přítomen.

Žádosti (requests)

Žádosti slouží k posílání požadavku. První řádek má následující strukturu:

`Request-Line: Method Request-URI SIP-Version CRLF`

- `Method`: Slouží pro identifikaci typu žádosti, existuje šest typů: `REGISTER, INVITE, ACK, CANCEL, BYE, OPTIONS`. Význam jednotlivých metod naleznete v tabulce 2.1.
- `Request-URI`: Identifikuje volaného uživatele nebo zařízení.
- `SIP-Version`: Obsahuje verzi protokolu.

Method	**význam**
INVITE	pozvánka k hovoru, jakékoliv navazování spojení musí začínat pomocí této metody
REGISTER	registrace účastníka hovoru na SIP server
ACK	potvrzení přijetí hovoru (potvrzuje, že přišla odpověď `OK`)
CANCEL	zrušení žádosti
OPTIONS	dojednává dodatečné informace
BYE	ukončení hovoru

Tabulka 2.1: Typy žádostí protokolu SIP dle RFC 3261 [15].

Odpovědi (responses)

Odpovědi protokolu SIP slouží jako zpětná vazba pro žádosti. První řádek začíná klíčovým slovem `Status-Line` a má následující strukturu:

`Status-Line: SIP-Version Status-Code Reason-Phrase CRLF`

- `Status-Code`: Číselná odpověď, která vyjadřuje výsledek žádosti. Existuje šest možných případů pro odpověď. Vždy záleží na první číslici, další slouží pouze jako doplňující informace. Typy odpovědí jsou uvedeny v tabulce 2.2.
- `Reason-Phrase`: Textový popis položky `Status-Code`. Slouží pouze jako překlad hodnoty `Status-Code` do čitelné podoby.

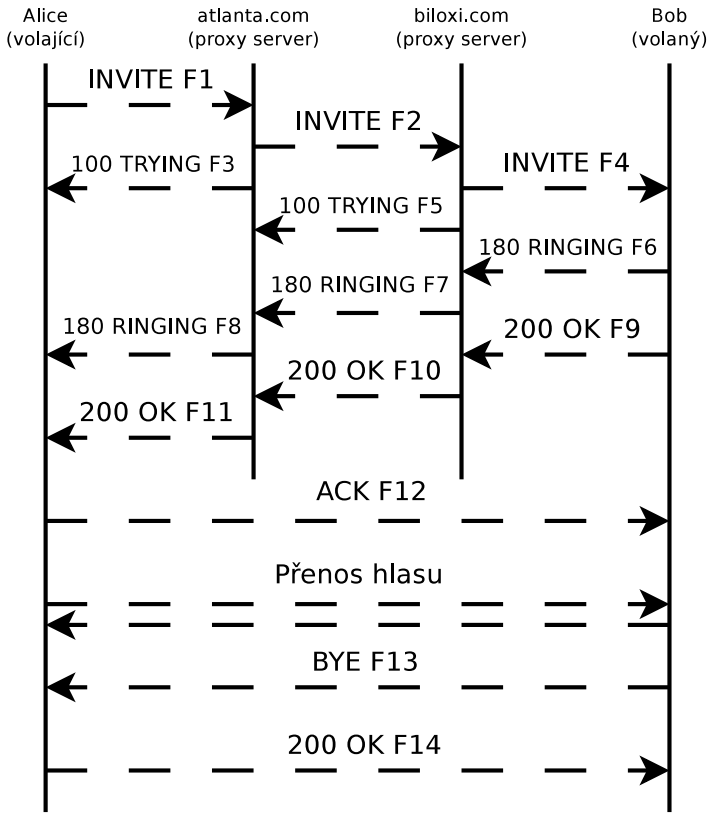
2.4.2 Ukázka navázání spojení přes SIP

Příklad navázání spojení pomocí protokolu SIP je na obrázku 2.1. Alice chce navázat spojení s Bobem, je tedy potřeba mu poslat pozvánku. Alice s Bobem nekomunikují přímo, nýbrž přes *proxy server*[4]. Každá zpráva v obrázku obsahuje identifikátor Fx, kde `x` značí pořadí akce.

[4]Počítač nebo aplikace, která přeposílá klientovy požadavky.

Status-Code	Význam
1xx	žádost byla přijata, bude se dále zpracovávat
2xx	žádost byla v pořádku přijata a potvrzena
3xx	přesměrování žádosti, jsou potřeba další informace
4xx	chyba na straně klienta, špatná žádost
5xx	chyba na straně serveru, žádost v pořádku, ale server není schopen ji vyřídit
6xx	globální chyba, žádný server nemůže vyřídit tuto žádost

Tabulka 2.2: Typy odpovědí protokolu SIP dle RFC 3261 [15].



Obrázek 2.1: Proces navazování spojení mezi dvěma koncovými uzly přes proxy server [15].

2.5 SDP

SDP (Session Description Protocol) je protokol poskytující informace o detailech spojení. Tyto informace poskytuje nezávisle na tom, jak jsou data přenášena. SDP neslouží pro samotné vytváření spojení (to je úloha protokolu SIP). Protokol může být využit pro více aplikací, dále se budu zabývat pouze jeho významem pro VoIP. Pro tuto práci je protokol důležitý z hlediska zjišťování adres a portů, na kterých bude probíhat RTP komunikace.

2.5.1 Struktura protokolu SDP

SDP se přenáší společně s protokolem SIP, přičemž je umístěn v části `message body`. Přítomnost SDP se pozná podle toho, že položka `Content-Type` protokolu SIP (v části

`*message-header`) obsahuje řetězec „`application/sdp`“. SDP je stejně jako SIP textový protokol.

Celý popis relace se skládá z několika řádků. Každý řádek je ve tvaru `<typ>=<hodnota>`. `Typ` musí být jedno z několika písmen charakterizujích, co se má popisovat. Dále se budu věnovat pouze typu *m* a *a*, jenž byly použity v této bakalářské práci.

Popis formátu média

Typ *m* popisuje médium. Jeho struktura je následující:

```
m = médium port/počet_portů transportní_protokol popis_formátu_média
```

`Médium` značí, co se má přenášet. Jelikož v této práci jsem se zabýval VoIP hovorem, tak akceptuji pouze řetězec „`audio`“ (další hodnoty mohou být např. „`video`“). Položka `port` obsahuje cílový port, `počet_portů` značí, kolik párů portů bude využito (pokud je hodnota rovna jedné, není uvedena). U VoIP se používá pro přenos hlasu protokol RTP, se kterým se mohou také posílat pakety RTCP. Oba druhy těchto paketů jsou přijímány na různých portech, standardně na sudém portu (položka `port`) se očekává RTP, na portu s číslem o jedna větším je přijímán protokol RTCP. Pokud se čísla portů mezi sebou neliší pouze o jedna, je nutno tuto skutečnost specifikovat v typu *a* (*attribute lines*). Pro pole `transportní protokol` jsou v této práci přípustné pouze řetězce „`RTP/AVP`“ a „`RTP/SAVP`“ (značí *Secure RTP* [2]).

`Popis_formátu_média` je obsažen ve čtvrté a jakékoliv následující položce. Pokud je transportní protokol RTP (dále budu uvažovat jen tuto možnost, jelikož jinak se data v technologii VoIP nepřenáší), tak je v této položce hodnota `payload type`, což je číslo obsahující kodek. Seznam všech čísel namapovaných na kodeky lze nalézt na stránce *Real-Time Transport Protocol (RTP) Parameters*[5]. Pokud je těchto čísel více, tak každý uvedený kodek může být použit, první z nich se bere jak výchozí [6].

Čísla 96-127 jsou rezervovaná pro dynamické kodeky. Tyto kodeky nemají předem přidělené číslo, a proto musí být na dané číslo namapovány. Mapování zajišťuje popis atributů (typ *a*). Na následujícím příkladu je znázorněno mapování čísla 97 na kodek iLBC [1]. Číslo 8000 vyjadřuje vzorkovací frekvenci kodeku.

```
a=rtpmap:97 iLBC/8000
```

Ukázka jednoho řádku z protokolu SDP popisující médium

Na následující ukázce se pokusím objasnit, jak vypadá jeden řádek z protokolu SDP popisující médium.

```
m=audio 7078 RTP/AVP 8 0
```

Strana posílající tento SDP paket bude přenášet RTP pakety na portu 7078, RTCP pakety na portu 7079. Výchozím kodekem je PCMA (číslo 8), a také může být použit kodek PCMU (číslo 0).

[5] http://www.ietf.org/assignments/rtp-parameters/rtp-parameters.xml#rtp-parameters-1

2.6 Využití znalostí k získání informací o hovoru

Návrh implementace a analýzy paketu bude popsán v dalších kapitolách. Zde je pouze nastíněno zpracování SIP paketů za použití znalostí získaných po prostudování této kapitoly.

Jelikož je SIP textový protokol (kódovaný v UTF-8), při analýze paketu se s ním pracuje velice snadno. Analýza SIP zpráv je nutná pro rozpoznávání jednotlivých hovorů, resp. pokusů o hovor, jelikož ne všechny pokusy o vytvoření musí být úspěšné.

Jelikož každý SIP paket obsahuje položku `Call-ID`, která jednoznačně identifikuje, k jakému hovoru patří, rozlišuji jednotlivé SIP relace na základě této položky.

Postup při analýze SIP paketů

1. Na základě `Call-ID` přiřadíme paket ke správnému hovoru. Pokud není hovor s daným `Call-ID` nalezen, je vytvořen nový záznam o hovoru.
2. Pokud paket obsahuje SDP, je třeba jej zpracovat.
3. Na základě typu zprávy SIP je třeba rozhodnout, do jakého stavu přejít (popsáno dále v sekci), popř. zůstat ve stejném stavu.

Všechny důležité informace se dají zjistit rozborem SIP, popř. SDP paketu. Dále je třeba uchovávat informace o stavu hovoru. Záznam o hovoru se může nacházet v jednom ze čtyř následujících stavů:

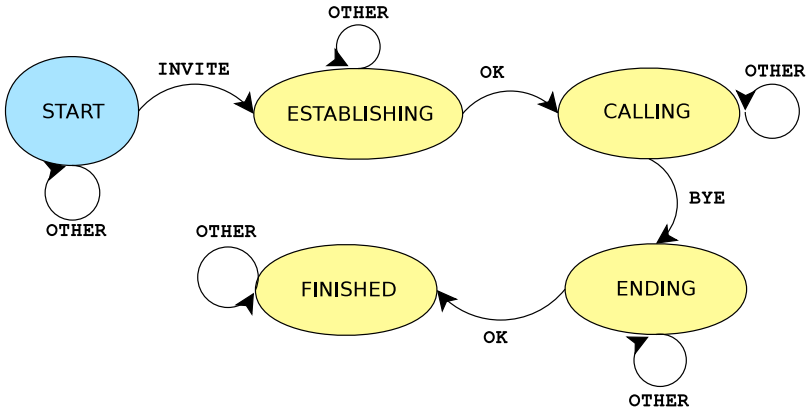
- navazování spojení,
- přenos hlasu,
- ukončování spojení,
- hovor je ukončen.

Jakmile přijde jakýkoliv paket SIP s metodou `INVITE`, je třeba zjistit, zda už hovor s daným `Call-ID` existuje. Pokud neexistuje, vytvoří se nový záznam s výchozím stavem *navazování spojení*. Jelikož čekám na zahájení hovoru, ostatní metody kromě odpovědi `200 OK` ignoruji. Jakmile přijde SIP s odpovědí typu `200 OK`, spojení bylo navázáno a začíná přenos RTP paketů (popsáno v kapitole 4). Přenos RTP probíhá, dokud nebyl odeslán požadavek SIP s metodou `BYE`. Jakmile byl takový paket přijat, přechází se do stavu *ukončování spojení*, ve kterém se bude záznam nacházet, dokud nepřijde další SIP paket s odpovědí `200 OK`. Logika je zobrazena na obrázku 2.2.

Pokud se hovor nachází v nějakém stavu příliš dlouho, je pravděpodobné, že některý ze SIP paketů byl ztracen. Z toho důvodu byly implementovány časovače, které kontrolují, zdali se hovor nenachází ve stejném stavu příliš dlouho. Více o tomto problému naleznete v kapitole 6.1.

2.7 Shrnutí

Technologie VoIP slouží pro přenášení telefonního hovoru po síti. Analogový signál je převeden do binární podoby a rozdělen do paketů. Narozdíl od klasických telefonních linek nemá VoIP garantovánu celou šířku pásma, tím pádem může docházet ke zhoršení kvality



Obrázek 2.2: Přechody mezi jednotlivými stavy pomocí zpráv protkolu SIP

hovoru. Sada opatření pro zlepšení podmínek přenosu paketů nesoucích telefonní hovor se nazývá *Quality of Service*.

Pro navazování spojení slouží signalizační protokoly. V této práci uvažuju pouze signalizační protokol SIP, který je v dnešní době nejrozšířenější. V některých paketech je také přítomen protokol SDP, který obsahuje mimo jiné adresy a porty, na kterých bude přenášen hovor (RTP tok). Díky protokolu SIP jsem tedy schopen vyfiltrovat ze síťového provozu jednotlivé hovory, pomocí SDP určit na jakých adresách a portech bude hovor přenášen, což mi umožňuje rozeznávat pakety náležící jednotlivým VoIP hovorům a měřit z nich statistiky.

Kapitola 3

Metody hodnocení kvality hovorů VoIP

Ačkoliv je měření kvality VoIP hovorů v této práci založeno pouze na monitorování síťového provozu, budete v této kapitole seznámeni s více druhy měření. V počátcích tohoto odvětví bylo využíváno subjektivní měření, dnes jsou využívány převážně objektivní metody. Všechny schválené metody měření kvality VoIP hovorů organizací ITU-T jsou uvedeny na stránce *Terminals and subjective and objective assessment methods*[1]. V této kapitole budete seznámeni pouze s několika nejdůležitějšími. Dále jsou zde představeny jednotlivé faktory, které beru v potaz při měření kvality. V sekci 3.4 je popsána metoda implementovaná v exportéru.

3.1 Stupnice MOS

Pro hodnocení kvality hovoru se v praxi používá stupnice *MOS* (Mean Opinion Score). Ve standardu P.800.1 [8] je tato stupnice definována jako průměr známek, kde známka značí hodnoty na předem dané stupnici, které použily subjekty pro ohodnocení kvality hovorů v poslechových nebo konverzačních testech.

Stupnice obsahuje pět hodnot:

- **Vynikající kvalita**: neznatelné rušení, kvalita podobná hovoru tváří v tvář.
- **Dobrá kvalita**: občas lze registrovat rušení, ale zvuk je stále čistý a rušení neobtěžuje účastníky hovoru.
- **Průměrná kvalita**: rušení obtěžuje účastníky hovoru.
- **Nízká kvalita**: rušení velmi obtěžuje, je třeba vyvinout značného úsilí, aby uživatel porozuměl.
- **Špatná kvalita**: nesrozumitelná řeč, komunikace je téměř nemožná.

Tato stupnice byla zavedena již při počátečních pokusech o subjektivní hodnocení kvality hovorů. V dnešní době objektivní metody vracejí jako výsledek hodnocení odhad na stupnici MOS.

[1] http://www.itu.int/rec/T-REC-P/en

3.2 Vysvětlení některých metod

Existují tři způsoby získání hodnoty pro stupnici MOS:

- **Subjektivní metoda**: metoda založená na výběru skupiny lidí a jejich subjektivním názoru na poslechovou kvalitu. Pro tuto metodu je potřeba velký pocet účastníků. V praxi se moc často nepoužívá.
- **Intrusivní metoda**: objektivní metoda, jejíž princip je založen na porovnávání původního a přeneseného signálu.
- **Neintrusivní metoda**: objektivní metoda, která se při posuzování kvality u VoIP hovorů používá nejčastěji, jelikož se dá implementovat do monitorovacích zařízení. Je založená na monitorování spojení. Výsledný MOS je pouze odhadem, jelikož není k dispozici originální signál.

3.2.1 Subjektivní hodnocení kvality hovorů

Standard ITU-T P.800 [7] definuje několik metod pro subjektivní hodnocení kvality hovorů. Zde se pokusím rozebrat pouze dvě základní, jelikož subjektivní hodnocení již není dnes kvůli finančním a časovým možnostem používané [17].

Conversation-opinion tests (konverzační testy)

Metoda probíhá v laboratořích, kde jsou co nejpřesněji nasimulovány určité podmínky (např. různé druhy šumů). Dokonce jsou doporučeny i rozměry místností, kde by měli hodnotící být umístěni. Hodnotící jsou umístěni každý v jiné místnosti a výsledkem každého z nich je hodnota na MOS stupnici.

Listening-opinion tests (poslechové testy)

Jednou z metod spadající do této kategorie je *ACR* (Absolute Category Rating). Na rozdíl od předchozí kategorie, zde jsou předem připraveny testovací vzorky a v místnosti by se neměl vyskytovat žádný šum. Posluchačům jsou tyto vzorky pouštěny v náhodném pořadí, vždy s krátkou prodlevou, během které posluchač ohodnotí vzorek na MOS stupnici.

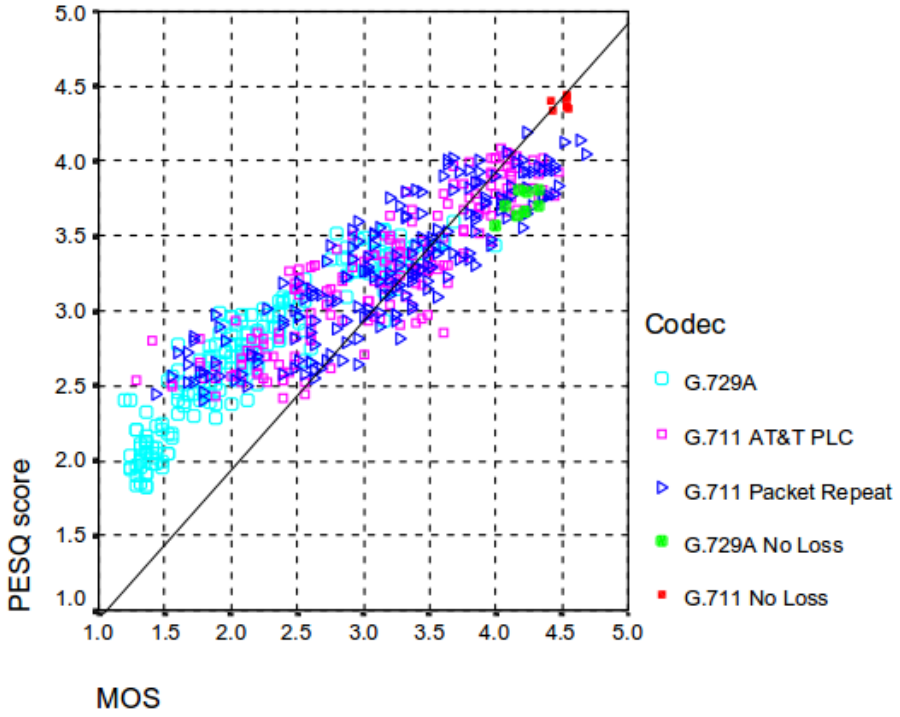
3.2.2 Metoda PESQ

Perceptual Evaluation of Speech Quality (PESQ) je metoda pro objektivní hodnocení kvality hovorů. Je popsána ve standardu ITU-T P.862 [11]. Metoda porovnává originální signál a signál, který prošel komunikačním systémem. Standard udává, že tato metoda není vhodná pro rozsáhlejší měření přenosové kvality. Je měřen pouze dopad hlukového zkreslení a šumu, čili může nastat situace, kdy se naměřená kvalita jeví velice dobře, ovšem např. kvůli ozvěně, velkému zpoždění apod. může být kvalita přenosu velice nízká.

Nejdříve je třeba vstupní a výstupní signál rozdělit na několik menších intervalů, aby se mezi sebou porovnávaly korespondující intervaly. Vlivem zpoždění by totiž mohlo docházet k situacím, kdy by se nemusely porovnávat stejné části hovoru. Poté jsou oba signály převedeny do interní reprezentace, v níž jsou porovnávány.

Výsledkem je hodnota, která se blíží hodnotě ze subjektivního poslechového testu na MOS stupnici (jinými slovy, tato metoda se snaží o zautomatizování poslechových testů z předchozí sekce). Odchylky naměřených hodnot od hodnot, kterými byly testované vzorky

předem ohodnoceny, můžete vidět na obrázku 3.1. Na obrázku je vidět, že výsledné skóre je spíše o něco vyšší, než hodnoty na MOS stupnici, největší rozdíly jsou pro kodek G.729.



Obrázek 3.1: Odchylky mezi naměřenými hodnotami metodou PESQ a danými hodnotami pro hovor [14].

3.2.3 Standard P.563

Standard ITU-T P.563 [9] popisuje další metodu pro objektivní měření kvality hovorů. Na rozdíl od metody PESQ, tato metoda nepoužívá k ohodnocení kvality originální signál a systémem deformovaný signál, nýbrž pouze deformovaný signál. Patří tedy do skupiny neintrusivních metod.

Každý přijatý signál musí být předem připraven pro samotné hodnocení kvality hlasu. Detektor hlasu *VAD* (Voice Activity Detector) identifikuje intervaly, ve kterých je v signálu přítomen hlas. Následně je tento signál připraven na několik analýz určující klíčové parametry hovoru. Na základě těchto parametrů je určena kvalita. Jednotlivé analyzující bloky jsou následující:

- analyzátor hlasu (mužský, ženský) a nepřirozenosti (robotický hlas),
- analyzátor šumu,
- analyzátor přerušování a tlumení hlasu.

Stejně jako u metody PESQ je výsledkem hodnota MOS-LQO (MOS-Listening Quality Objective).

3.2.4 E-Model

E-Model je definován ve standardu ITU-T G.107 [10]. Nabízí mírně odlišný přístup pro hodnocení kvality hovorů oproti předchozím metodám. Ty hodnotily hlavně poslechovou kvalitu (přenášený signál), zatímco E-Model se také zaměřuje na ostatní faktory ovlivňující kvalitu hovoru. Do výpočtů zahrnuje také přenosové charakteristiky, čili se dostáváme od poslechové kvality ke konverzační, kde je celková kvalita závislá např. na zpoždění. Jinými slovy, pokud bude přenesen signál ve vysoké kvalitě, ale za nepřiměřeně dlouhou dobu, nebude brán hovor jako kvalitní.

Výstupem je skalární hodnota *R-faktor*, která reflektuje kvalitu hovoru na stupnici 0-100. R-faktor se vypočítá následovně:

$$R = R_o - I_s - I_d - I_{e-eff} + A \tag{3.1}$$

K výpočtu každé hodnoty z předchozí rovnice je třeba mnoho vzorců, ty jsou uvedeny přímo ve standardu ITU-T G.107 [10]. Zde je uveden pouze význam jednotlivých hodnot pro představu, na čem R-faktor záleží:

R_o : Poměr užitečného signálu a šumu (SNR - signal to noise ratio).

I_s : Součet všech rušících vlivů během hovorů, které jsou k němu vázány a nelze je oddělit (nízká hlasitost, kvantizační šum, neoptimální nastavení zařízení).

I_d : Faktor reflektující všechny druhy zpoždění (echo, echo posluchače, zhoršení způsobené příliš velkým absolutním zpožděním od úst k uchu).

I_{e-eff} : Skutečný faktor zhoršení zařízení. Vychází z I_e (zhoršení způsobené zařízením), který obsahuje pro jednotlivé kodeky hodnoty změřené subjektivně. Skutečný faktor zhoršení zařízení navíc reflektuje odolnost kodeku proti ztrátám.

A : Faktor zvýhodnění, bere v potaz použité zařízení (mobilní telefon, pevný telefon, terminály v těžce dostupných oblastech, ...).

Pro všechny vstupní hodnoty jsou zavedeny počáteční hodnoty ve standardu ITU-T G.107 [10].

Výstupní hodnota R se poté převádí na stupnici MOS podle následující rovnice [10]:

$$MOS = \begin{cases} 1 & R \leq 0 \\ 1 + 0,035 \cdot R + R \cdot (R - 60) \cdot (100 - R) \cdot 7 \cdot 10^{-6} & R < 0 < 100 \\ 4,5 & R \geq 100 \end{cases} \tag{3.2}$$

V tabulce 3.1 můžete pro představu vidět, jak odpovídají některé hodnoty R-faktoru hodnotám na stupnici MOS.

3.3 Sledované statistiky

V této práci jsou pro posuzování kvality brány v potaz následující faktory:

- klouzavý rozptyl (jitter),
- ztrátovost paketů,

- zpoždění

Tyto faktory byly vybrány, jelikož jsme schopni je změřit ze síťového provozu. Význam těchto hodnot naleznete dále. S jejich výpočtem budete seznámeni v sekcích 4.3 a 4.4 společně s protokoly RTP a RTCP.

R-faktor	Hodnota na stupnici MOS
90	4.34
80	4.04
70	3.60
60	3.10
50	2.58

Tabulka 3.1: Převedení R-faktoru na stupnici MOS [10].

3.3.1 Rozptyl (jitter)

Rozptyl je definován jako doba mezi očekávaným a skutečným příchodem paketu. Důvodem tohoto jevu je, že ne všechny RTP pakety mají stejné podmínky přenosu, popř. nemají stejnou cestu k cíli. V tom případě se může stát, že později odeslané pakety dorazí dříve než pakety, které byly odeslány před nimi. Pro výpočet kvality je použit klouzavý rozptyl, který reflektuje celkový průběh. Pro jeho výpočet se používá průměr šestnácti předchozích rozptylů.

K řešení tohoto problému slouží buffer, kam se příchozí pakety ukládají. Buffer má ovšem omezenou velikost, pokud paket přijde příliš brzy nebo příliš pozdě, tak je zahozen. V bufferu se RTP pakety řadí podle sekvenčního čísla v RTP hlavičce.

3.3.2 Ztrátovost paketů

Tato hodnota vyjadřuje poměr *počet odeslaných paketů ku počtu úspěšně doručených paketů*. Výpočet se provádí na základě sekvenčních čísel v RTP hlavičce. Existují dvě příčiny ztrátovosti: paket může být ztracen na síti (hovory běží nad protokolem UDP), nebo může být paket zahozen na zařízení, jelikož jitter buffer není dostatečně velký pro uložení paketu s tímto sekvenčním číslem. Každý kodek má svou vlastní přijatelnou míru ztrátovosti paketů.

3.3.3 Zpoždění

Zpoždění koncových bodů (end-to-end delay) je definováno jako doba mezi promluvením a vyslechnutím daného signálu. Nezpůsobuje na rozdíl od předchozích faktorů nesrozumitelnost. Způsobuje situace, kdy se účastníci hovoru těžko dorozumívají kvůli tomu, že RTP pakety dorazily příliš pozdě. Je žádoucí, aby RTP pakety dorazily do cíle co nejdříve a tím pádem uživatel na druhé straně mohl co nejdříve odpovědět. Zpoždění do 150 ms ani neregistrujeme, do 400 ms již poznáme, ovšem stále se lze dorozumívat. Zpoždění více jak 400 ms už uživatele omezuje v tom, že se nevědomky navzájem přerušují [17].

3.4 Metoda hodnocení kvality hovorů použitá v této práci

Existuje tzv. zjednodušený E-Model, který vypočítá R-faktor pouze na základě zpoždění koncových bodů a ztrátovosti paketů. Vzorec vznikl dosazením implicitních hodnot ze standardu ITU-T G.107 [10]. Základní vzorec ze kterého tento model vychází je následující [18]:

$$R = 94,7688 - 1,4136 - I_d - I_{e-eff} \tag{3.3}$$

Hodnoty I_d a I_{e-eff} mají stejný význam jako v sekci 3.2.4, ovšem výpočet se liší. I_d je vyjádřeno jako funkce zpoždění koncových bodů podle následujícího vzorce [18]:

$$I_d = \begin{cases} 0,0267 \cdot T & T < 175\ ms \\ 0,1194 \cdot T - 15,876 & 170\ ms \leq T \leq 400\ ms \end{cases} \tag{3.4}$$

V předchozím vzorci je reflektováno, že při zpoždění větším než 170 ms klesá R-faktor rychleji. Skutečný faktor zhoršení zařízení se počítá dle vzorce [10]:

$$I_{e-eff} = I_e + (95 - I_e) \cdot \frac{P_{pl}}{\frac{P_{pl}}{BurstR} + B_{pl}} \tag{3.5}$$

I_e je faktor zhoršení zařízení a je to tabulková hodnota pro jednotlivé kodeky. P_{pl} značí ztrátovost paketů v procentech a $BurstR$ vyjadřuje, zdali je ztrátovost náhodná ($BurstR = 1$) nebo ve shlucích ($BurstR > 1$). Pro vyšší kvalitu je žádoucí, aby ztrátovost byla co nejmenší, a pokud už nějaká existuje, aby byla náhodně rozptýlená. B_{pl} (Packet-loss Robustness Factor) vyjadřuje odolnost kodeku proti ztrátám a je to tabulková hodnota. Hodnoty pro I_e a B_{pl} jsou uvedeny v tabulce 3.2.

Kodek	**Kódovací rychlost**	I_e	B_{pl}
G.711 (bez PLC)	64 kb/s	0	10
G.711 (s PLC)	64 kb/s	0	34
G.723.1	5.3 kb/s	19	24
G.723.1	6.3 kb/s	15	20
G.726-16	16 kb/s	40	69
G.726-24	24 kb/s	25	38
G.726-32	32 kb/s	12	24
G.726-40	40 kb/s	7	24
G.728	16 kb/s	16	27
G.729	8 kb/s	10	18
G.729A	8 kb/s	11	17
GSM FR 6.10	13.2 kb/s	26	43

Tabulka 3.2: Odhady hodnot pro I_e a B_{pl} [12].

Zjednodušený E-Model bere tedy v potaz ztrátovost paketů, zpoždění a kvalitu kodeku. Dle článku [13] lze do tohoto výpočtu zakomponovat také rozptyl. Tuto metodu se pokusím nastínit v následujících odstavcích.

Oproti zjednodušenému E-Modelu je změněn vzorec pro výpočet skutečného faktoru zhoršení zařízení I_{e-eff}. Místo ztrátovosti, ke které dochází při síťovém provozu, je použita efektivní ztrátovost P_{plef}. Tato hodnota reflektuje také ztrátovost, ke které dochází vlivem omezené velikosti jitter bufferu. Efektivní ztrátovost je součtem ztrátovosti na síti, na jitter

bufferu a součinem těchto hodnot (pravděpodobnost, že by obě ztrátovosti mohly nastat naráz). P_{plef} se počítá dle následujícího vzorce [13]:

$$P_{plef} = P_{pl} + P_{dejitter} + P_{pl} \cdot P_{dejitter} \tag{3.6}$$

Pravděpodobnost času příchodu paketů lze spolehlivě nasimulovat pomocí distribuční funkce „*Pareto*“[2] [13]. Pomocí ní lze odhadnout, zdali vzhledem k velikosti rozptylu a jitter bufferu bude příchozí paket zahozen (ověřené hodnoty pro „*Pareto*“ distribuční funkci byly přejaty z [13]). Vztah je ukázán na následující rovnici, x značí velikost jitter bufferu a σ je velikost rozptylu [13].

$$P_{dejitter} = \frac{(1 + \frac{-0.1x}{\sigma})^{20}}{2} \tag{3.7}$$

Jak již bylo řečeno, faktor I_{e-eff} nahrazuje ztrátovost na síti efektivní ztrátovostí. Článek [13] uvádí následující vzorec pro výpočet skutečného faktoru zhoršení zařízením následovně:

$$I_{e-eff} = I_e + (95 - I_e) \cdot \frac{P_{plef}}{P_{plef} + B_{pl}} \tag{3.8}$$

Výsledný vzorec pro výpočet R-faktoru je tedy následující:

$$R = 94.7688 - 1.4136 - I_d - \left(I_e + (95 - I_e) \cdot \frac{P_{plef}}{P_{plef} + B_{pl}} \right) \tag{3.9}$$

3.5 Kodek

Kodek slouží pro převod analogového signálu do digitální podoby a zpět. Hlas (analogový signál) je zakódován do přenositelné podoby a na druhém zařízení je pomocí kodeku dekódován. Existuje metoda *PLC* (Packet Loss Concealment), pomocí které lze zahlazovat ztracené pakety. Tím pádem lze částečně maskovat ztrátovost. Ovšem tuto techniku neberu v této práci v potaz.

V předchozí sekci byla hodnota I_{e-eff} (skutečný faktor zhoršení zařízením) závislá mj. na hodnotách I_e a B_{pl}. Právě tyto 2 hodnoty reflektují kvalitu kodeku ve výsledném vzorci.

V tabulce 3.3 lze vidět závislost R-faktoru na kvalitě kodeku. Pro všechny výpočty byly použity následující hodnoty: T=100 ms, jitter=40 ms, jitter buffer=20 ms, ztrátovost 0 %.

Kodek	**Ie**	**Bpl**	**R-faktor**
G.711 (s PLC)	0	34	90.19
G.711 (bez PLC)	0	10	89.01
G.729	10	18	79.85
G.723.1	19	24	71.12
GSM	26	43	64.40

Tabulka 3.3: Závislost R-faktoru na kvalitě kodeku.

[2] http://mathworld.wolfram.com/ParetoDistribution.html

3.6 Shrnutí

Existuje celá řada standardů, které dokážou ohodnotit kvalitu hovorů. Pro účely této práce samozřejmě nepřipadá v úvahu žádná ze subjektivních metod. Metoda PESQ dokáže hodnotit hovor pouze na základě originálního a degradovaného signálu, což při monitorování provozu sítě není možné. Standard P.563 používá pouze systémem degradovaný signál, ovšem také není pro tuto práci vhodný, jelikož pracuje pouze se samotným přenášeným signálem, nikoliv se síťovými parametry. E-Model sice bere v potaz ztrátovost a zpoždění, ovšem stále je zaměřen mj. na rozbor signálu a použitých zařízení (faktor zvýhodnění). Jedna možnost by byla použít zjednodušený model a R-faktor tedy spočítat pouze na základě ztrátovosti a zpoždění.

Dle článku [13] lze do výpočtu výpočtu zjednodušeného E-Modelu zakomponovat i rozptyl (potažmo velikost jitter bufferu), díky kterému může také docházet ke ztrátovosti paketů. Podle testů ve článku [13] byla většina hodnot vypočítaných dle této metody přesnější než odhad podle originálního E-Modelu. *Z předchozích dvou důvodů byla tedy pro výpočet výsledné kvality použita metoda uvedená v sekci* 3.4.

Kapitola 4

Výpočet kvality přenosu RTP a RTCP

V předchozích kapitolách jste se dozvěděli o technologii VoIP a průběhu navazování hovorů. V této kapitole budete seznámeni s protokolem RTP, jehož funkcí je přenášet samotná data, hlas překonvertovaný do digitální podoby. Spolu s RTP tokem se také může přenášet RTCP tok, který obsahuje informace o RTP toku. Z těchto dvou protokolů lze měřit jednotlivé faktory ovlivňující kvalitu hovorů, což je uvedeno v sekcích 4.3 a 4.4.

RTP (Real-time Transport Protocol) poskytuje přenos audio a video dat nad protokolem IP. Standardně používá nespojovanou službu UDP, jelikož pro hovory je důležitější zajistit jejich plynulost (i za cenu ztráty některých paketů), než nulovou ztrátovost paketů. RTP nehlídá:

- doručování dat ve správném pořadí,
- doručování dat v určitém čase,
- spolehlivé doručení dat.

Pro každý VoIP hovor jsou potřeba dva RTP toky, jeden směrem *volající* → *volaný* a druhý směrem *volaný* → *volající*. Každý z nich má svou vlastní posloupnost sekvenčních čísel a časových značek, nejsou tedy na sobě nijak závislé.

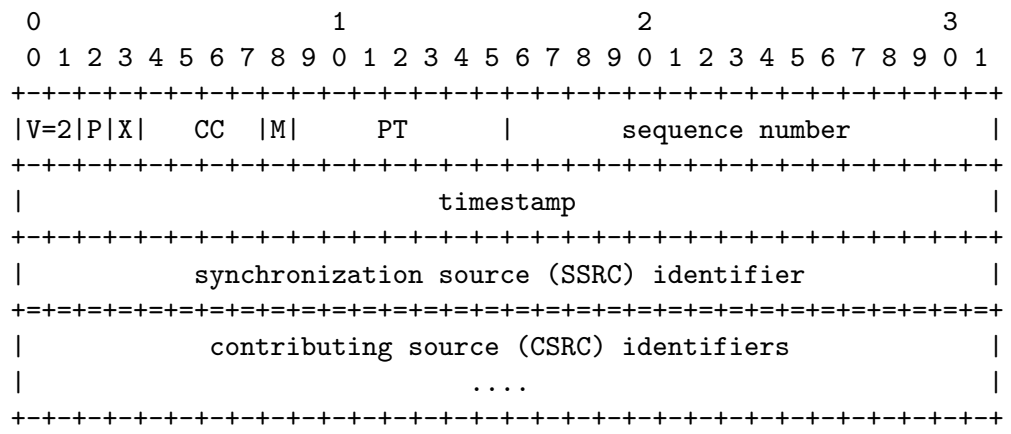
4.1 RTP hlavička

Jelikož se v práci zabývám pouze hodnocením kvality VoIP hovoru na základě síťového provozu, potřebuji se zabývat RTP hlavičkou, nikoliv hlasem převedeným do digitální podoby (*payload*). Podoba RTP hlavičky je uvedena na obrázku 4.1.

Prvních 12 bajtů musí být obsaženo v každé RTP hlavičce, pole `CSRC` je nepovinné. V této práci budou využita následující pole:

- **PT (payload type)**: Obsahuje formát přenášených dat. U VoIP hovoru obsahuje kodek podle tabulky uvedené na stránce *RTP Payload types (PT) for standard audio and video encodings*[1].

[1] http://www.iana.org/assignments/rtp-parameters/rtp-parameters.xhtml#rtp-parameters-1



Obrázek 4.1: RTP hlavička, přejato z RFC 3550 [16].

- **Sequence number**: Jednoznačný identifikátor pořadí, v jakém byl RTP paket odeslán. Automaticky se zvyšuje po odeslání RTP paketu. Počáteční hodnota je náhodná.
- **Timestamp**: časová značka RTP paketu. Pravidelně se inkrementuje o určitou hodnotu na základě hodin (nemůže se zvětšovat v libovolných časových intervalech nebo o měnící se hodnotu). Počáteční hodnota je náhodná.

  Mějme kodek s frekvencí 8 KHz, pakety přenáší audio o délce 20 ms. V tom případě se za sekundu přenese 50 paketů. Počet vzorků (8000) ku počtu paketů za sekundu (50) udává výslednou hodnotu pro `timestamp` (160).

  – 1. RTP paket, `timestamp = 8160`
  – 2. RTP paket, `timestamp = 8320`
  – 3. RTP paket, `timestamp = 8480`
  – ...

Pole `SSRC` (synchronization source identifier) je jednoznačný identifikátor zdroje RTP paketu. Jeho hodnota je náhodná a konstantní po celou dobu RTP relace.

Pro jeden VoIP hovor bude v obou RTP tocích stejná položka `payload type`, jelikož kodek se za dobu hovoru měnit nemůže. V každém RTP toku ale `sequence number` začíná od jiné počáteční hodnoty, stejně tak logicky položka `SSRC` musí obsahovat pro různé toky různé hodnoty, jelikož každý RTP tok má jiný zdroj.

4.1.1 Validace RTP hlavičky

Validace RTP hlavičky je sada několika kroků, aby se ověřilo, zdali je možné z tohoto paketu počítat statistiky. Tato kontrola je sice slabá, ovšem pokud přijde vadný RTP paket, nebude to mít vliv na chod exportéru, pouze nebudou počítány statistiky z tohoto paketu. Kroky, které vedou k úspěšné validaci RTP hlavičky jsou následující [16]:

1. položka `Version` musí být rovna 2.
2. položka `Payload Type`, obsahující informace o kodeku, musí být exportéru známá.

Jakmile jsou splněny předchozí podmínky, je třeba zkontrolovat sekvenční čísla. RTP tok je považován za validní pouze po zachycení několika paketů (definováno jako makro `MIN_SEQUENTIAL`) s po sobě jdoucími sekvenčními čísly. Tento algoritmus je uveden v RFC 3550 [16] v příloze A.1 a byl odtamtud přejat. Je zde také ošetřen případ, kdy je zachycen paket se špatným sekvenčním číslem (liší se o více, než je povoleno). V tomto případě se může jednat o případ, kdy zdroj začal vysílat pakety s jiným bázovým sekvenčním číslem. Dokud není zdroj s novými sekvenčními čísly opět validován, nepočítají se z přijatých RTP paketů statistiky.

4.2 RTCP

RTCP (RTP Control Protocol) slouží pro přenos statistik a informací o kvalitě příjmu RTP paketů. RTCP tok nemusí být přítomen u všech RTP proudů. Pokud je ovšem přítomen, lze jej využít pro sběr statistik. Jestliže není, statistiky se budou počítat přímo z hodnot v RTP hlavičce. RTCP pakety jsou posílány v určitých intervalech. Výpočet této periody je nad rámec této práce a pro analyzátor nepodstatný. Pokaždé, když sonda zachytí RTCP paket, snaží se z něj získat dané informace.

Existuje pět typů RTCP zpráv:

- **SR** (Sender Report): Generují je aktivní účastníci hovoru.
- **RR** (Receiver Report): Generují je pasivní účastníci hovoru.
- **SDES** (Source Description): obsahuje popis některých položek (hlavně *CNAME*, což je jednoznačný identifikátor relace).
- **BYE**: Indikuje konec účasti.
- **APP**: Aplikačně specifické funkce.

Jednotlivé zprávy se mohou spojovat, čili jeden paket může obsahovat více zpráv, přičemž s každou zprávou lze pracovat individuálně. Každý RTCP paket musí obsahovat minimálně jednu zprávu typu `Sender Report` nebo `Receiver Report`, které jsou pro nás z hlediska posuzování kvality nejdůležitější, jelikož obsahují informace o RTP tocích.

4.2.1 Struktura RTCP zpráv SR a RR

`SR` se posílá tehdy, pokud byly od předchozího odeslání této zprávy poslány nějaké RTP pakety, jinak se posílá `RR`. Oba typy zpráv mají podobnou strukturu, která je uvedená v RFC 3550 [16].

Sender report

Typicky má tento druh zprávy minimálně tři části - hlavičku, informace pro aktivní účastníky hovoru a samotnou zprávu obsahující informace o RTP toku. Vyznačuje se tím, ze položka `Packet type` v hlavičce obsahuje identifikátor `200`.

Receiver report

Struktura je podobná jako u `SR`, jediný rozdíl je v tom, že v `RR` se neposílá informace pro aktivní účastníky hovoru.

Důležité položky pro výpočet kvality hovorů VoIP

Ze zpráv `SR` a `RR` jsou pro výpočet kvality hovorů důležité následující položky:

- **NTP timestamp**: Časová značka vytvoření této zprávy ve formátu *NTP* (Network Time Protocol)[2]. Udává počet sekund vzhledem k datu 1.1.1900.
- **Last SR timestamp**: Hodnota udávající časovou značku (prostředních 32 bitů z pole `NTP timestamp`) poslední přijaté zprávy.
- **Delay since last SR Timestamp**: Interval mezi přijetím poslední zprávy a odesláním této zprávy. Vyjádřeno v jednotkách 1/65536 sekund.
- **Interarrival jitter**: Hodnota klouzavého rozptylu počítaná na koncovém zařízení.
- **Cumulative number of packet lost**: Celkový počet ztracených RTP paketů.

4.3 Získávání měřených hodnot z RTP paketů

Pokud RTP tok neposílá souběžně RTCP, je třeba požadované hodnoty získat přímo z RTP paketů. Popis jednotlivých položek a jejich význam najdete v sekci 3.3. Zde se zaměřuji pouze na získávání hodnot z RTP paketů.

4.3.1 Klouzavý rozptyl

Výpočet této hodnoty probíhá stejně, jak jej definuje RFC 3550 [16] pro výpočet hodnoty `Interarrival jitter` v RTCP. Nejdříve je potřeba spočítat okamžitý rozptyl. K tomu jsou potřeba následující hodnoty:

- **S**: Položka `timestamp` v RTP hlavičce (hodnotu je potřeba převést do časových jednotek vydělením vzorkovací frekvence kodeku).
- **R**: Čas zachycení paketu.

Okamžitý rozptyl pro paket zachycený v čase i se spočítá následovně [17]:

$$D_i = (R_i - R_{i-1}) - (S_i - S_{i-1}) \tag{4.1}$$

Pro výpočet klouzavého rozptylu J v čase i je třeba průměr šestnácti předchozích rozptylů [17]:

$$J_i = J_{i-1} + \frac{|D_i| - J_{i-1}}{16} \tag{4.2}$$

Příklad výpočtu s reálnými hodnotami

Mějme dva po sobě následující příchozí pakety ve stejném RTP toku s následujícími hodnotami:

čas zachycení	kodek	sequence number	timestamp
25.051616	ITU-T G.711 PCMA	0	400
25.051641	ITU-T G.711 PCMA	1	560

[2] http://www.ntp.org/

Jitter u prvního paketu bude nulový. Okamžitý rozptyl mezi pakety 0 a 1 se spočítá následovně:

$$D(1) = (25.051641 - 25.051616) - (\frac{560}{8000} - \frac{400}{8000}) = -0,019975 \tag{4.3}$$

Kodek G.711 má vzorkovací frekvenci 8000 Hz. Vzorkovací frekvence pro jednotlivé kodeky jsou uvedeny na stránce *Real-Time Transport Protocol (RTP) Parameters*[3].

Okamžitý rozptyl převedeme na klouzavý (ten je také obsažen v RTCP paketech a je označen jako Interarrival Jiter):

$$J(1) = 0 + \frac{|-0,019975| - 0}{16} = 1,248\ ms \tag{4.4}$$

Tímto způsobem se tedy počítá rozptyl pro všechny příchozí RTP pakety. Pro srovnání, program Wireshark ukazuje pro paket se sekvenčním číslem 1 hodnotu pro rozptyl 1,25 ms. Drobný rozdíl je způsoben zaokrouhlovacími chybami. Výslednou kvalitu takové rozdíly neovlivní.

4.3.2 Ztrátovost paketů

Podle RFC 3550 [16] je třeba pro výpočet ztrátovosti paketů využít hodnotu `extended_max`, jelikož položka `Sequence number` je pouze šestnácti bitová. Je třeba ji rozšířit na 32 bitů.

Číslo `cycles` má nulovou počáteční hodnotu a zvyšuje se pokaždé, když sekvenční číslo dosáhne svého maxima ($2^{16} - 1$) a další sekvenční čísla budou přicházet od nejnižší hodnoty. Hodnota je zvyšována o číslo 65 536 [16]. `Extended-max` je tedy 32-bitová hodnota, kde vyšších 16 bitů značí počet přetečení sekvenčních čísel a nižších 16 bitů obsahuje nejvyšší sekvenční číslo.

$$extended\_max = cycles + max\_seq\_number \tag{4.5}$$

Položku `max_seq_number` lze vyčíst z RTP hlavičky (viz 4.1) a je to nejvyšší hodnota položky `sequence number` ze všech obdržených validních RTP hlaviček. Hodnota `expected` udává počet očekávaných RTP paketů.

$$expected = extended\_max - first\_received\_seq\_number + 1 \tag{4.6}$$

Položka `lost` značí celkový počet ztracených RTP paketů. K výpočtu je potřeba hodnota `received`, což je počítadlo přijatých RTP paketů (toto počítadlo také obsahuje duplikované RTP pakety nebo pakety, které přišly ve špatném pořadí).

$$lost = expected - received \tag{4.7}$$

Problém nastává u duplicitních paketů. V této práci není kontrolována duplicita paketů, tudíž může nastat situace, kdy tento výpočet nebude zcela přesný. Mějme např. dva ztracené pakety a dva duplicitní pakety, v tomto případě tedy bude hodnota `lost` rovna nule, ovšem správně by měla být rovna dvěma. Pokud se tedy v toku vyskytují duplicitní pakety, výpočet nemusí být přesný.

Jako směrodatný údaj považuji ztrátovost paketů, což je poměr ztracených paketů ku celkovému počtu paketů, které by měli být přijaty.

$$packet\ loss = \frac{lost}{expected} \tag{4.8}$$

[3] http://www.ietf.org/assignments/rtp-parameters/rtp-parameters.xml#rtp-parameters-1

4.3.3 Zpoždění koncových bodů

Pro výpočet zpoždění koncových bodů z RTP toku neexistuje žádný standard. Je to z toho důvodu, že sonda nemá se zdrojem toku žádné sesynchronizované hodiny. Na sondu přicházejí pakety, které sice obsahují položku `Timestamp`, ovšem tato hodnota je pro výpočet zpoždění nepoužitelná, jelikož začíná na náhodné hodnotě.

Jako odhad zpoždění byl proto přejat algoritmus z programu VoIPmonitor[4]. Zpoždění bude záviset na aktuálním rozptylu a na předchozí hodnotě zpoždění.

$$avgdelay = \frac{(avgdelay\_prior \cdot received - 1) + jitter}{received} \tag{4.9}$$

Toto je pouze odhad zpoždění, přesnější výsledky vykazuje výpočet zpoždění z RTCP paketů.

4.4 Získávání měřených hodnot z RTCP paketů

Pokud se společně s RTP tokem posílá také RTCP tok, lze jej taktéž využít pro sledování kvality. Ovšem je potřeba si uvědomit, že rozptyl a ztrátovost jsou hodnoty, které jsou počítány na straně přijímajícího a následně odesílány jako zpětná vazba. Pokud například putuje RTCP paket z adresy A na adresu B, nese v sobě informace o rozptylu a ztrátovosti pro tok z adresy B na adresu A. Pro zpoždění, které je nutno vypočítat z několika hodnot, toto neplatí. Hodnoty je třeba agregovat. Řešení tohoto problému naleznete v kapitole 5.1.2.

4.4.1 Klouzavý rozptyl

Protokol RTCP počítá tuto hodnotu dle RFC 3550 [16], algoritmus je uveden v sekci 4.3.1. Hodnota je v RTCP paketu uložena jako položka `Interarrival Jitter`. Průměrný rozptyl na sondě je tedy počítán následovně (N značí počet přijatých RTCP paketů):

$$avg\_jitter = \frac{\sum_{i=1}^{N}(interarrival\ jitter_i)}{N} \tag{4.10}$$

4.4.2 Ztrátovost paketů

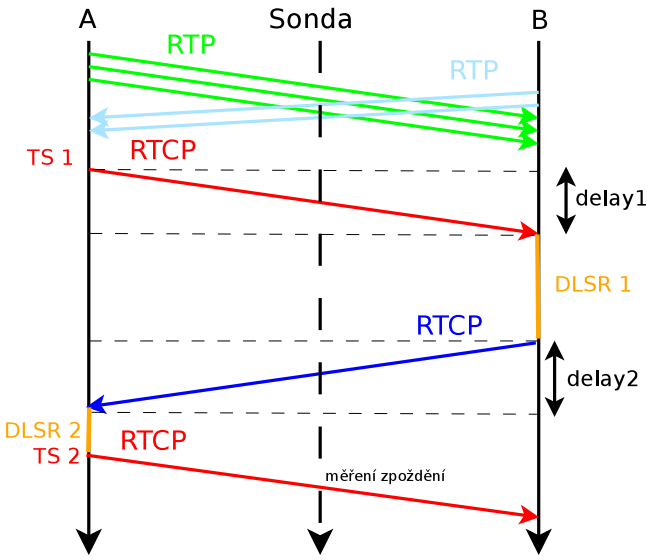
Vyjádřit ztrátovost pomocí RTCP paketů je opět velice jednoduché. K výpočtu jsou potřeba dvě hodnoty, celkový počet ztracených paketů a počet předpokládaných přijetých RTP paketů (součet přijetých a ztracených paketů).

$$packet\ loss = \frac{cumulative\ number\ of packet\ lost}{cumulative\ number\ of packet\ lost + sender\ packet\ count} \tag{4.11}$$

4.4.3 Zpoždění koncových bodů

Výpočet zpoždění je ze všech tří měřených faktorů nejobtížnější, protože vyžaduje speciální struktury a také složitější výpočty. Výpočet lze na rozdíl od RTP toků provést přesněji, jelikož RTCP protokol obsahuje synchronizované hodiny v podobě hodnoty `NTP timestamp`. Měří se zpoždění RTCP paketů, které jsou vysílány v určitých intervalech (řádově sekundy).

[4] http://www.voipmonitor.org



Obrázek 4.2: Znázornění výpočtu *RTD* (Round trip delay) z RTCP paketů.

K výpočtu jsou zapotřebí dva toky navzájem opačného směru. Výpočty v této práci se mírně liší od výpočtů v RFC 3550 [16], jelikož tam tvůrci předpokládají, že k měření dochází přímo na zdroji RTCP paketů. Ovšem v této práci je sonda umístěna v síti mezi dvěma zdroji RTCP paketů. V RFC 3550 lze počítat přímo s časem zachycení paketu na koncovém zařízení. Zde je tato hodnota nahrazena mechanismem, který určuje zachycení RTCP paketu na zdroji tak, že od časové hodnoty značící vznik dané RTCP zprávy odečte hodnotu značící prodlevu mezi naposledy přijatou zprávou a vznikem této zprávy (DLSR). Názorná ukázka měření zpoždění je na obrázku 4.2.

Budeme měřit tzv. *RTD* (Round trip delay), který vyjadřuje čas mezi posláním paketu a přijetím odpovědi (na obrázku je to součet intervalů *delay1* a *delay2*). Nás ale zajímá jednosměrné zpoždění *OWD* (One way delay), proto RTD podělíme dvěma a dostaneme výsledek.

Od časového okamžiku *TS 2*, kdy byl vygenerována zachycená zpráva, odečteme dobu, kdy se nic nedělo (`DLSR 2`). Od této hodnoty odečteme *DLSR 1* ze stejného důvodu jako v předchozím případě. Na závěr ještě zbývá odečíst *TS 1*, což je doba od kdy se měří RTD (tuto hodnotu nese paket náležící toku $B \rightarrow A$ v poli `LSR`). K výpočtu je tedy třeba příchozí RTCP paket a naposledy zachycený paket opačného RTCP toku.

Výpočty zpoždění pro jednotlivé toky se musí střídat, čili spočítá se zpoždění pro tok $A \rightarrow B$, poté se musí počítat pro tok $B \rightarrow A$. K výpočtům nedochází u prvních paketů jednotlivých toků.

Předchozí hodnoty jsou z RTCP získány následovně:

- **TS 2**: Prostředních 32 bitů z pole *NTP timestamp* zachyceného paketu.
- **DLSR 2**: Hodnota *DLSR* ze zachyceného paketu.
- **DLSR 1**: Hodnota *DLSR* z posledního zachyceného RTCP paketu opačného toku.
- **TS 1**: Hodnota *LSR* z posledního zachyceného RTCP paketu opačného toku.

Pokud neexistuje opačný RTCP tok k danému toku, není možné spočítat zpoždění, jelikož by chyběly hodnoty *DLSR 1* a *TS 1*.

4.5 Interpretace výsledků

Při interpretaci výsledků je nutné brát v úvahu to, že sonda je při reálném provozu umístěna mezi dvěma uzly. Tím pádem mohou být naměřené hodnoty z RTP toků zkresleny. Sonda totiž zachytává pakety, ale pak pracuje pouze s jejich kopiemi a originální paket prochází dál. Situaci se pokusím vysvětlit na následujícím příkladu. Mějme dva mezi sebou komunikující uzly A a B. Pokud je sonda například hned za uzlem A a naměříme nulovou ztrátovost, ve skutečnosti RTP tok může mít na koncovém uzlu B ztrátovost vyšší. Pro sondu neexistuje žádný mechanismus kterým by se dala tato hodnota zjistit přesně. Obdobná situace může nastat i pro rozptyl nebo zpoždění.

Jelikož hodnoty pro RTCP pakety jsou počítány přímo na koncových uzlech, je výsledná kvalita přesnější. *Proto je vždy lepší při interpretaci výsledků odečítat výslednou kvalitu z daného RTCP toku, pokud je přítomen.* Algoritmy pro počítání rozptylu a ztrátovosti z RTP toků v praktické části této práce jsou stejné jako algoritmy, se kterými počítá protokol RTCP. Ovšem z výše popsaných důvodů se mohou výsledky lišit. Jinými slovy, statistiky pro RTP tok jsou počítány mezi koncovým uzlem a sondou, proto se liší od statistik z RTCP paketů, které jsou počítány mezi 2 koncovými uzly.

Na obrázku 4.3 je zobrazen výstup z kolektoru při analýze hovoru. Lze na něm pozorovat některé jevy popsané výše.

1. Průměrný rozptyl ve směru 192.168.2.106 → 213.168.165.12 je na koncovém uzlu přibližně 10x větší než je naměřený na sondě. Mohlo by to být způsobeno tím, že sonda byla blízko uzlu 192.168.2.106, tím pádem by nebyla schopna počítat rozptyl který by nastal po průchodu paketu sondou.

2. Hodnoty pro zpoždění se mezi RTP a RTCP toky liší, je to dáno nepřesnou metodou pro počítání zpoždění z RTP toků.

Src IPv4:sPort	->	Dst IPv4:dPort	Flow_Type	Avg_Jitter[ms]	Delay[ms]	Packet_Loss[%]	R_Factor	MOS	Quality
192.168.2.106:7078	->	213.168.165.12:13446	RTP	7.910	7	0.000	92.999	4.405	Excellent
192.168.2.106:7079	->	213.168.165.12:13447	RTCP	76.062	15	0.000	90.091	4.341	Excellent
213.168.165.12:13447	->	192.168.2.106:7079	RTCP	0.344	11	0.000	92.906	4.404	Excellent
213.168.165.12:13446	->	192.168.2.106:7078	RTP	0.604	0	0.000	93.200	4.409	Excellent

Obrázek 4.3: Ukázka výstupu z kolektoru.

4.6 Shrnutí

Pro přenos hovoru se používá protokol RTP, v této práci nás ovšem nezajímají samotná data která přenáší, ale pouze hlavička, ze které lze měřit jednotlivé faktory ovlivňující kvalitu. Rozptyl a ztrátovost jsou u RTP toků počítány dle RFC 3550 [16]. Jelikož zdroje RTP toků mezi dvěma zařízeními nejsou nijak synchronizovány, nelze přesně určit zpoždění. Výpočet zpoždění z RTP toku je méně přesný než z RTCP toku. Algoritmus pro výpočet zpoždění z RTP paketů byl přejat z programu VoIPmonitor.

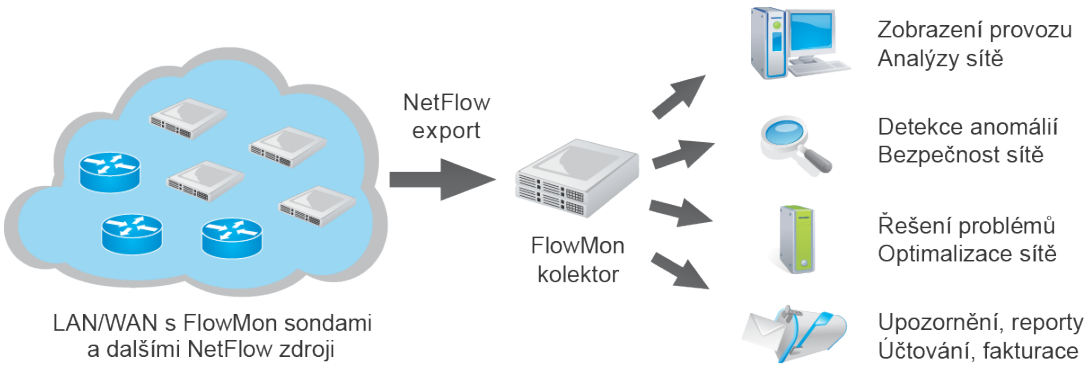
Pokud je přítomen protokol RTCP, který tyto hodnoty obsahuje, je lepší hodnoty odečítat z tohoto toku, protože jsou počítány přímo na koncových zařízeních. Ve výstupu budou tedy vždy minimálně dva RTP toky charakterizující hovor a u každého z nich může být přítomen RTCP tok.

Kapitola 5

Implementace analyzátoru v sondě

Pro určení kvality hovorů je potřeba monitorování jednotlivých RTP/RTCP toků. K tomuto účelu mi posloužila sonda FlowMon od firmy Invea-Tech[1]. Dále v této práci bude pod pojmem sonda myšlena právě sonda od firmy Invea-Tech. V této kapitole budete seznámeni s principem exportéru, kolektoru a s protokolem IPFIX, pomocí něhož tyto dvě komponenty komunikují. Další podstatnou částí této kapitoly je popis implementace jednotlivých pluginů pro exportér.

Sonda je zařízení umístěné v síti, jehož hlavním úkolem je monitorovat síťový provoz. Ze zachycených paketů může počítat určité statistiky, které poté posílá na kolektor. Na kolektoru probíhá vyhodnocování informací, ať už v textové podobě (použito v této práci) nebo např. v podobě grafů. Tento proces je ukázán na obrázku 5.1.



Obrázek 5.1: Spolupráce exportéru a kolektoru, přejato ze stránek produktu FlowMon[2].

5.1 Exportér

Exportér je program běžící na sondě, která je umístěna v síti. Jeho stěžejní úkoly v této práci jsou:

- zachytávání paketů ze síťového provozu,
- rozpoznávání hovorů na základě signalizačního protokolu SIP,

[1] http://www.invea.com/cs
[2] http://www.invea.com/cs/products/flowmon

- třídění paketů do příslušných toků,
- počítání statistik z RTP/RTCP toků,
- export ukončených toků na kolektor.

V praxi je FlowMon sonda hardwarový nástroj, pro účely této práce jsem pracoval s virtuální verzí FlowMon. Jedná se o počítač s 64-bitovou verzí operačního systému CentOS.

Funkci exportéru plní aplikace `flowmonexp`, která již byla nainstalována na sondě. Jedná se o multivláknovou aplikaci, kde každé vlákno má specifickou úlohu. Důležitá vlákna pro tento projekt jsou:

- **Vstupní vlákno**: Zde probíhá zachytávání paketů, třídění do jednotlivých toků a také extrakce informací z jednotlivých paketů.
- **Flow cache vlákno**: Toto vlákno již pracuje s jednotlivými toky. Jedná se o vytváření a aktualizaci toků. Zde také probíhají veškeré výpočty týkající se kvality hovorů VoIP. Jednotlivé toky jsou uloženy v paměti, která se nazývá *flow cache*.
- **Expirační vlákno**: Periodicky prochází *flow cache* a zjišťuje, které toky jsou připraveny pro exportování na kolektor.
- **Exportní vlákno**: Stará se o export expirovaných toků.

Program `flowmonexp` je již hotovým programem. Pro přizpůsobení exportéru našim potřebám *musíme dodat jednotlivé pluginy, které upraví funkčnost exportéru a tím pádem exportér dokáže ze síťového provozu identifikovat jednotlivé VoIP hovory a analyzovat jejich kvalitu*. V našem případě exportér využívá následující pluginy: vstupní plugin, procesní plugin a exportní plugin. Pluginy byly implementovány v jazyce `C`.

Exportní plugin byl dodán společně s exportérem, implementace vstupního a procesního pluginu bude popsána v následujících sekcích. Každý plugin musí povinně obsahovat následující funkce (`pluginType` značí typ pluginu: vstupní nebo procesní).

void *plugin_pluginType_init(char *params, flow_record_getter_t **getter_list, int full_packet, int data_offset)

Tato funkce se volá vždy jen jednou a to při startu exportéru. Je vhodná pro alokování zdrojů a zpracování parametrů, se kterými byl plugin spuštěn, nebo pro inicializaci tzv. getterů (viz 5.1.4). Vstupní plugin např. otevírá zdroj paketů nebo inicializuje tabulku hovorů. Funkce vrací ukazatal na strukturu, která obsahuje privátní data a je k dispozici za běhu pluginu.

plugin_desc_t *plugin_pluginType_desc()

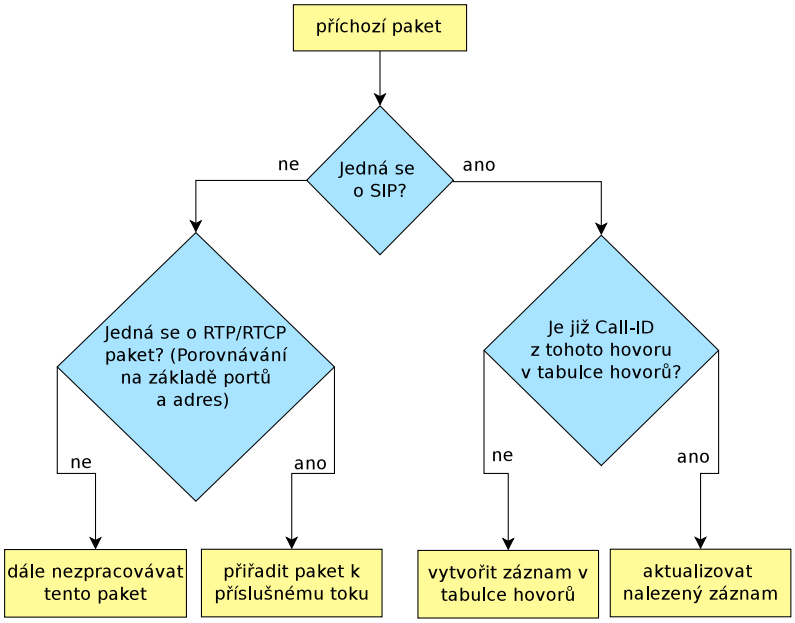
Tato funkce je velice jednoduchá a jejím cílem je pouze vrátit ukazatel na strukturu `plugin_desc_t`, která obsahuje popis pluginu. Struktura jednak obsahuje jméno pluginu (jméno je potřeba k identifikaci pluginu při zadávání parametrů) nebo např. nápovědu k pluginu.

int plugin_pluginType_shutdown()

Jak již název této funkce napovídá, je funkce volána opět pouze jednou a to před ukončením běhu exportéru. Slouží pro uvolňování zdrojů.

5.1.1 Vstupní plugin

Ve vstupním pluginu dochází k zachytávání jednotlivých paketů. Tím pádem zde také probíhá rozpoznávání jednotlivých hovorů a třídění jednotlivých paketů do toků. Princip fungování vstupního pluginu je popsán na obrázku 5.2. Jednotlivé záznamy o hovorech jsou ukládány v tabulce hovorů. Filtrace hovoru ze síťové komunikace je popsána v kapitole 2.6. Tabulka hovoru je implementována jako hashovací tabulka z knihovny `uthash`[3].



Obrázek 5.2: Princip práce vstupního pluginu

uint64_t plugin_input_get_flow(void* plugin_private, flow_record_t* record)

Toto je stěžejní funkce vstupního pluginu. Z ní se postupně volají funkce pro analýzu paketů. Zajišťuje zachytávání paketů a kontroluje, zdali se paket obsahuje protokol SIP. Pokud obsahuje, aktualizuje tabulku hovorů. Jestliže paket neobsahuje protokol SIP, snaží se pomocí tabulky hovorů zjistit, zdali se jedná o paket RTP/RTCP.

Jedním z úkolů této funkce je získávání informací z RTP/RTCP paketů a jejich předávání procesnímu pluginu pro počítání statistik. Jedná se o číselné hodnoty, jejichž získávání ovšem není tak snadné jako v případě protokolu SIP nebo SDP, protože zde jsou data uložena v binární podobě. Pro tyto účely byla vytvořena makra `GETUINTx(source, offset)`, kde `x` značí velikost čtené hodnoty v bitech (8, 16, 24, 32). Parametr `offset` je číselná hodnota značící začátek číselné hodnoty v řetězci `source`. Offsety jednotlivých hodnot jsou obsaženy v dokumetu RFC 3550 [16] v kapitolách 5.1 (pro RTP) a 6.4 (pro RTCP).

Jakmile jsou hodnoty z paketu získány, jsou uloženy do struktury, která je k danému záznamu uložena s určitým posunem. Tento posun je nastaven ve funkci `plugin_input_init` a jelikož je to globální proměnná, dokáže s ním pracovat i procesní plugin a tím pádem dokáže procesní plugin pracovat i s hodnotami získanými z RTP/RTCP paketů. Takto funguje předávání získaných hodnot mezi vstupním a procesním pluginem.

[3] http://troydhanson.github.io/uthash/

Pokud máme informace o hovorech, a zjistíme, že příchozí RTP/RTCP paket náleží nějakému hovoru, je potřeba jej přiřadit ke správnému toku. Toto se děje pomocí funkce `record_process_packet`, která vrací číselný hash, na základě kterého se paket přiřadí ke správnému toku. Hash se získá na základě informací v řetězci `packet`, který jsme zachytili. Informace jsou zároveň uloženy v parametru `record`.

Pokud se nejedná o RTP/RTCP paket, vrací funkce hodnotu `0`. Jinak vrací hodnotu, kterou získala z funkce `record_process_packet` a na základě které se budou jednotlivé pakety třídit do toků.

Tabulka hovorů

Tabulka hovorů má dva hlavní účely: uchovávání informací o jednotlivých hovorech z pohledu signalizačního protokolu SIP a uložení informací o adresách a portech jednotlivých účastníků hovoru. Klíčem, podle kterého se v ní vyhledává, je jednoznačný identifikátor hovoru `Call-ID` z protokolu SIP.

V tabulce jsou umístěny porty pouze pro toky RTP, protože tato čísla byla získána z protokolu SDP. Jelikož potřebujeme zpracovávat i RTCP toky, je třeba do porovnávání zakomponovat i porty o jedna vyšší. Pro vyhledávání v tabulce na základě portů a adres slouží funkce `findByFlowParams`, která jako argumenty dostává zdrojové/cílové adresy/porty. Vrací buď nalezený záznam v tabulce hovorů nebo `NULL` a pokud je záznam nalezen, do argumentu `type` uloží hodnotu značící, zdali se jedná o RTP nebo RTCP tok.

5.1.2 Procesní plugin

Hlavním úkolem procesního pluginu je počítat jednotlivé statistiky pro určení kvality hovoru. Jakmile procesní plugin obdrží paket, je již jasné že se jedná o RTP nebo RTCP paket příslušný k probíhajícímu VoIP hovoru. Procesní plugin má k dipozici tři funkce, které budou dále rozebrány.

void plugin_process_create(void *plugin_private, flow_record_t *record)

Tato funkce se volá vždy, když má být vytvořen nový tok. V praxi to znamená tehdy, když přijde paket s hash hodnotou, která ještě není obsažena v paměti flow cache. Funkce je u každého toku volána právě jednou a slouží převážně pro inicializaci počátečních hodnot jednotlivých měřených statistik.

void plugin_process_update(void *plugin_private, flow_record_t *record, flow_record_t *update)

Z daných tří funkcí je tato volána nejčastěji, a to při aktualizaci toku, čili pokud přijde paket a jeho hash hodnota je nalezena v paměti flow cache. Na základě hodnot získaných z RTP/RTCP paketů se v této funkci počítají jednotlivé statistiky. Samotné vzorce pro počítání statistik je uvedeno v kapitolách 4.3 a 4.4. Vzorce z těchto kapitol zde byly implementovány.

Problém nastává v případě, kdy se počítají statistiky pro RTCP toky.

- Pro výpočet zpoždění je potřeba práce se dvěma toky.
- RTCP posílá rozptyl a ztrátovost jako zpětnou vazbu, přenášené informace se vztahují k opačnému RTP toku. Proto je potřeba opět pracovat se dvěma toky, aby se kvalita počítala z hodnot pro stejný RTP tok.

Ovšem exportér umožňuje pracovat pouze s tokem ke kterému patří právě zpracovávaný paket. Z toho důvodu je i v procesním pluginu použita hashovací tabulka z knihovny `uthash`, která uchovává informace z více toků a data z jednotlivých toků jsou v ní agregována. Jelikož je definovaná jako globální proměnná, je přístupná odkudkoliv.

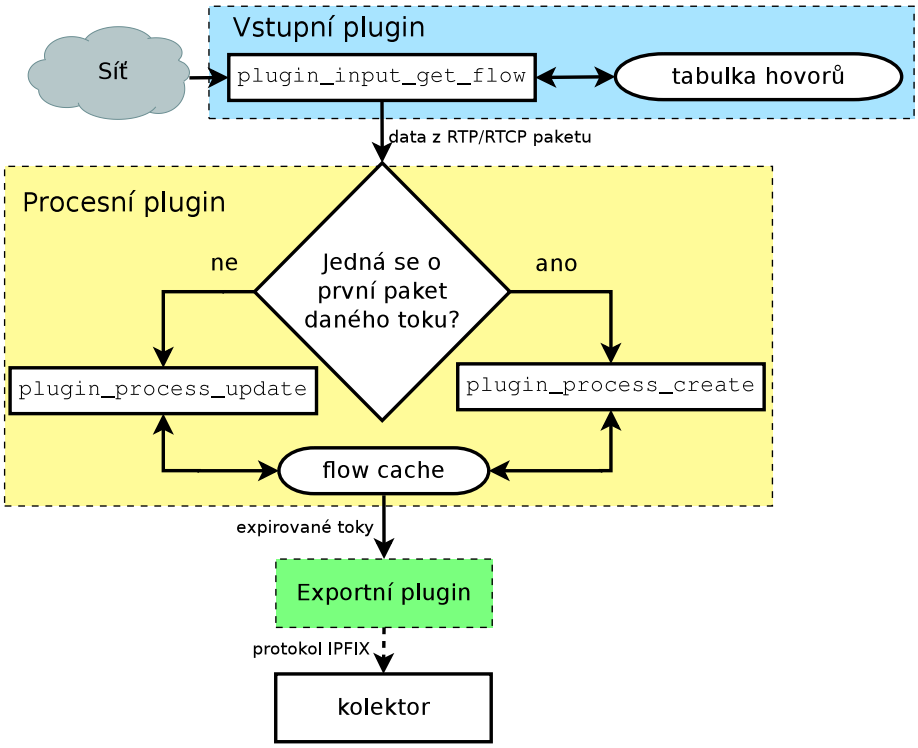
void plugin_process_release(void *plugin_private, flow_record_t *record, int reason)

Tato funkce je stejně jako funkce při vytváření toku volána právě jednou, a to těsně před exportováním toku. V této práci byla využita pro nastavení platnosti hodnot u tzv. *getterů*.

5.1.3 Spolupráce pluginů

Na obrázku 5.3 je zobrazen princip analýzy jednotlivých toků (shrnutí sekcí 5.1.2 a 5.1.1). Obousměrné šipky na obrázku znamenají práci s pamětí (čtení/zápis). Exportní plugin byl dodán společně s exportérem a nebyl v této práci nijak upraven.

Pokud chceme předávat vlastní data mezi pluginy, je třeba je ukládat do struktury, která je vzhledem k záznamu uložena s určitým offsetem. Vstupní plugin předává procesnímu pravě odkaz na tento záznam. Jelikož je offset globální proměnná, jeho hodnota je známa i procesnímu pluginu, a proto pro něj není problém přistoupit k těmto datům.



Obrázek 5.3: Spolupráce jednotlivých pluginů.

5.1.4 Gettery - funkce pro naplnění toků vlastními hodnotami

Tyto funkce umožňující pracovat s položkami v tocích, které se budou exportovat. V praxi to znamená, že jsou používány pro naplnění vlastních položek, které bude protokol IPFIX

přenášet (více o IPFIX a definici vlastních položek v sekci 5.2). V této práci byly tedy použity gettery pro naplnění položek: *rozptyl*, *zpoždění*, *ztrátovost*, *výsledná kvalita (MOS, R-faktor, slovní popis)* a *typ toku (RTP nebo RTCP)*.

Getter je potřeba nejdříve inicializovat. Jako nejvhodnější místo pro volání funkce pro inicializaci getteru se jeví `init` funkce. Všechny gettery jsou sdruženy do seznamu,který se nazývá `getter_list`. Každý getter musí obsahovat tři stěžejní funkce:

- **Validity funkce**: funkce říká, kdy jsou data připravená k zápisu do flow recordu. V této práci je toto řešeno pomocí bitové mapy. Každý getter má svůj vlastní bit. Hodnota 1 říká, že pro tento getter jsou připravena data k zapsání do flow recordu, hodnota 0 znamená, že data připravena nejsou.
- **Length funkce**: funkce vrací délku, kterou budeme zapisovat do flow recordu.
- **Fill funkce**: funkce spočítá a naplní hodnotu ve flow recordu. Do této funkce se exportér dostane poté, co *validity funkce* nahlásí tento getter jako platný.

Vzhledem k tomu, že u těchto tří funkcí máme jako argument odkaz na privátní strukturu a známe offset pro data, není problém k nim přistoupit. Jednotlivé výsledné hodnoty tedy mohou být počítány těsně před exportem, což může vést k vyššímu výkonu, jelikož se nebudou muset počítat při každé změně dílčích hodnot. Toho je využito při počítání výsledné kvality, která je vyhodnocena právě v getteru. Pro počítání výsledné kvality byly implementovány vzorce z kapitoly 3.4.

5.1.5 Export toků

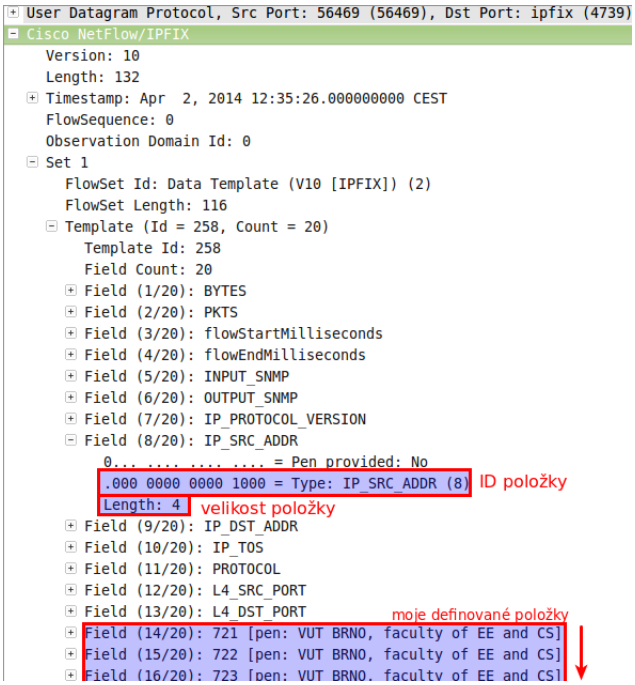
K exportu toků dochází při jedné z následujících podmínek:

- **Vypršení aktivního timeoutu**: Aktivní timeout značí maximální dobu, po kterou může být tok monitorován. Pokud je tedy aktivní timeout nastaven na 300 sekund, po uplynutí této doby (pokud nebyl z jiných důvodů již exportován) bude tok exportován.
- **Vypršení neaktivního timeoutu**: Neaktivní timeout je maximální doba prodlevy mezi dvěma příchozími pakety naležícími stejnému toku. Mějme tedy situaci, kdy je neaktivní timeout nastaven na 10 sekund. Pokud přijde 1000 paketů za sekundu, posléze za 10 sekund nepřijde žádný, tok bude exportován.
- **Povolení v bitové mapě**: Jedná se o nastavení určitých bitů v bitmapě. Na základě kombinace nastavených bitů může dojít k okamžitému exportu nebo k exportu může dojít až za určitý čas.
- **Před ukončením exportéru**: Všechny neukončené toky jsou před ukončením exportéru vyexportovány.

Aktivní a neaktivní timeout lze nastavit při spuštění exportéru pomocí parametrů.

5.2 Protokol IPFIX

Internet Protocol Flow Information Export (IPFIX) [5] je protokol pro sběr informací o jednotlivých síťových tocích. Někdy také bývá označován jako NetFlow v10. Tento protokol definuje jak řadit pakety do jednotlivých toků. Vzhledem k tomu, že IPFIX poskytuje pouze informace o toku jako celku, nepotřebuje zpracovávat data z aplikační vrstvy. Pakety se třídí do toků na základě následujících údajů [4]:



Obrázek 5.4: Zachycená IPFIX šablona poslaná z exportéru na kolektor.

- zdrojová a cílová IP adresa,
- zdrojový a cílový port,
- IP protokol,
- IP ToS (Type of Service),
- vstupní rozhraní.

Tento způsob zpracování paketů do jednotlivých toků je velmi šetrný k paměti a dokáže zpracovat velké množství síťového provozu. V praxi se NetFlow, popř. IPFIX používají například pro detekci útoků, měření přenesených dat, měření vytíženosti sítě apod.

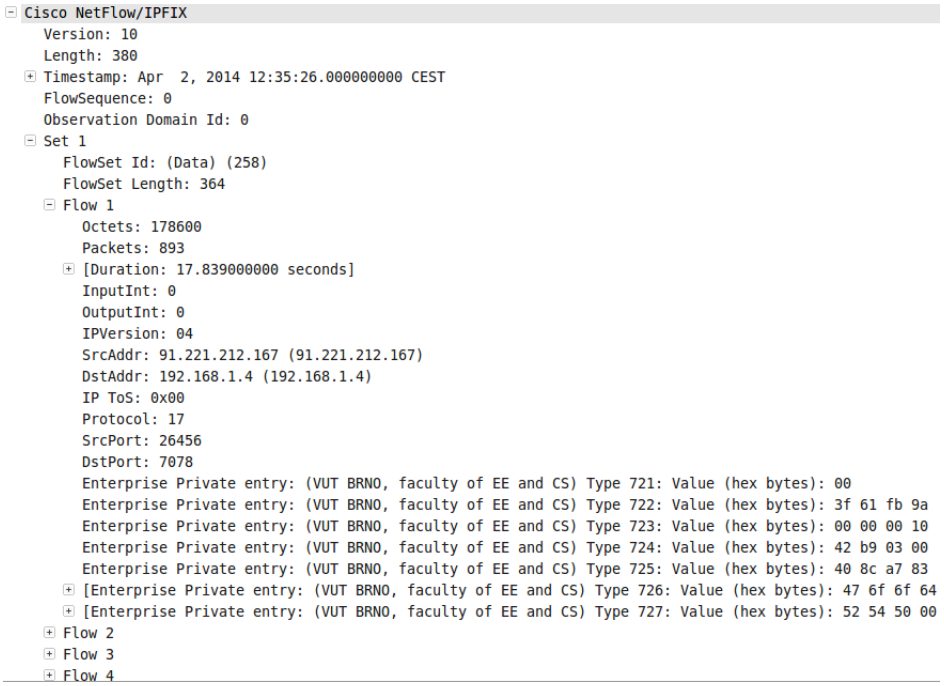
5.2.1 Přenos IPFIX dat

Pro přenos IPFIX se používá nespojovaná služba UDP. Exportér na sondě implicitně přenáší několik položek, ovšem je možné k těmto položkám připojit vlastní. Toho je využito pro přenos hodnot pro měřené statistiky. Pro lepší názornost byla na kolektoru zachycena ukázka dat přenášených z kolektoru na exportér.

Nejdříve se pošle IPFIX šablona (obrázek 5.4). V šabloně se posílají dvě stěžejní informace: *ID položky* a *velikost položky*. Identifikační číslo se skládá ze dvou hodnot, jednak *enterprise ID* (označující organizaci využívající toto ID) a také *ID*, které musí být v rámci organizace unikátní. Jednotlivá enterprise čísla jsou k nalezení na stránce *Private Enterprise Numbers*[4]. Standardizované položky lze najít na stránce *IPFIX Information Elements*[5].

V dalším jsou následně poslány hodnoty pro jednotlivé toky. V jednom paketu může být exportováno více toků. Ukázka je v příloze 5.5.

[4] http://www.iana.org/assignments/enterprise-numbers/enterprise-numbers
[5] http://www.iana.org/assignments/ipfix/ipfix.xhtml#ipfix-information-elements



Obrázek 5.5: Zachycená IPFIX data přenášená z exportéru na kolektor.

5.3 Kolektor

Kolektor je program jehož hlavní funkcí je spolupracovat s exportérem a sbírat data, která exportér sesbíral, spočítal a následně vyexportoval.

Práce s kolektorem

Před startem exportéru je třeba vědět číslo portu, na kterém bude kolektor naslouchat. Pokud nebude kolektor spuštěn, nemá exportér kam posílat data. Jakmile jsou nějaká data na kolektoru zachycena, jsou uložena v člověku nečitelné podobě. Pro čtení těchto dat používám program `fbitdump`, který umožňuje definovat různé šablony pro výstup. Výstup mojí práce je v textové podobě.

5.4 Shrnutí

Sonda je zařízení, pomocí něhož jsme schopni monitorovat síťový provoz. V této práci je využita pro monitorování VoIP hovorů. V praxi se exportér používá pro monitorování, jakmile má data připravena k odeslání, exportuje je na kolektor. Na kolektoru lze data zobrazit v člověku čitelné podobě a vyvodit z nich určité závěry.

Samotný exportér v této práci bylo potřeba přizpůsobit pro analýzu VoIP hovorů. Byly implementovány pluginy pro exportér v jazyce `C`, které jsou schopny rozpoznat VoIP hovor, monitorovat RTP/RTCP pakety, třídit je do toků, a počítat z nich statistiky. Následně je exportér schopen z těchto statistik určit výslednou kvalitu jednotlivých RTP toků.

Kapitola 6

Testování

Jednou z částí tohoto projektu je testování. Testování by mohlo probíhat ze tří rovin: paměťové nároky, časové nároky a správnost výsledků. V této práci jsem se zaměřil pouze na testování správnosti výsledků, což zahrnuje kontrolu vypočítaných hodnot. Každá naměřená hodnota (rozptyl, ztrátovost paketů a zpoždění) je testována jinak. V další části kapitoly se dozvíte jak se exportér chová v případě, kdy chybí nějaké zprávy protokolu SIP. Na závěr je v kapitole uvedeno srovnání výstupu z mého exportéru a ostatních programů.

Pracoval jsem s několika vzorky hovorů, které byly uloženy ve formátu pcap. Jelikož exportér dokáže pracovat s knihovnou *libpcap*, může data číst přímo ze souboru, nebo ze síťového rozhraní (pomocí programu *tcpreplay* lze posílat data na síťové rozhraní virtuální sondy).

Kontrola výpočtu rozptylu

Kontrola této položky je pravděpodobně nejjednodušší, jelikož naměřené hodnoty lze porovnávat s hodnotami, které naměřil program *Wireshark*. Podle dokumentu RTP Statistics[1] Wireshark počítá rozptyl podle dokumentu RFC 3550 [16], proto pro kontrolu mých naměřených hodnot rozptylu považuji tento program za referenční.

Kontrola výpočtu zpoždění

Jelikož měření zpoždění se provádí pouze u RTCP toků, nemusí být tato hodnota měřena vždy. Kontrola probíhala na náhodném vzorku dat. Jelikož zpoždění během běžného hovoru je počítánou pouze několikrát, není problém snadno ověřit naměřené hodnoty podle obrázku 4.2.

Kontrola výpočtu ztrátovosti

Pro simulaci ztrátovosti packetů byl použit program *editcap*. Tento program dokáže protřídit `pcap` soubor ze dvou pohledů: jednak dokáže nasimulovat celkovou ztrátovost, a také dokáže vymazat pouze určité pakety, což je výhodné pro testování různých situací s protokolem SIP. Nicméně nedokáže nasimulovat ztrátovost paketů tak, aby se to projevilo i v RTCP paketech.

[1] http://wiki.wireshark.org/RTP_statistics

6.1 Testování navazování spojení

Jak již bylo řečeno, program *editcap* dokáže vyjmout určité pakety z `pcap` souborů. Toho bylo využito pro odstranění určitých paketů s protokolem SIP a sledování chování exportéru.

1. Není zachycen SIP paket s žádostí INVITE, ovšem byla zachycena odpověď OK s protokolem SDP. Jelikož do tabulky hovorů se vkládá pouze na základě metody INVITE, je tento paket s odpovědí OK ignorován a k analýze hovoru nedojde.

2. Byl zachycen SIP paket s metodou INVITE, ovšem v určitém čase nebyla zachycena odpověď typu OK. V tomto případě již může probíhat hovor, ovšem jelikož nebyla zachycena odpověď s protokolem SDP, není možné znát jednu stranu hovoru (volajícího nebbo volaného) a tím pádem sonda tento hovor ignoruje. Aby se předcházelo situacím, kdy v paměti navždy zůstane položka v tabulce hovorů, byl implementován časovač, který se spustí po určité době. Pokud se hovor stále nachází ve fázi navazování spojení (po přijetí paketu SIP s metodou INVITE), vymaže jej z tabulky hovorů.

3. Ze stejného důvodu jako v předchozím bodě je nastaven časovač na následující intervaly:

   - Začátek měření hovoru a konec hovoru (interval mezi přijetím odpovědi OK a žádosti BYE). Tento časovač musí být nastaven na dostatečně velkou hodnotu, aby se nepřerušil měřený hovor.
   - Interval mezi přijetím žádosti typu BYE a odpovědi OK.

6.2 Srovnání výsledků s jinými programy

Pro testování měřených hodnot a výsledné kvality hovorů bylo také využito některých jiných softwarových nástrojů. Konkrétně se jedná o volně dostupný program Wireshark[2], komerční program PacketScan[3] od společnosti GL Communications Inc. a program VoIPmonitor[4].

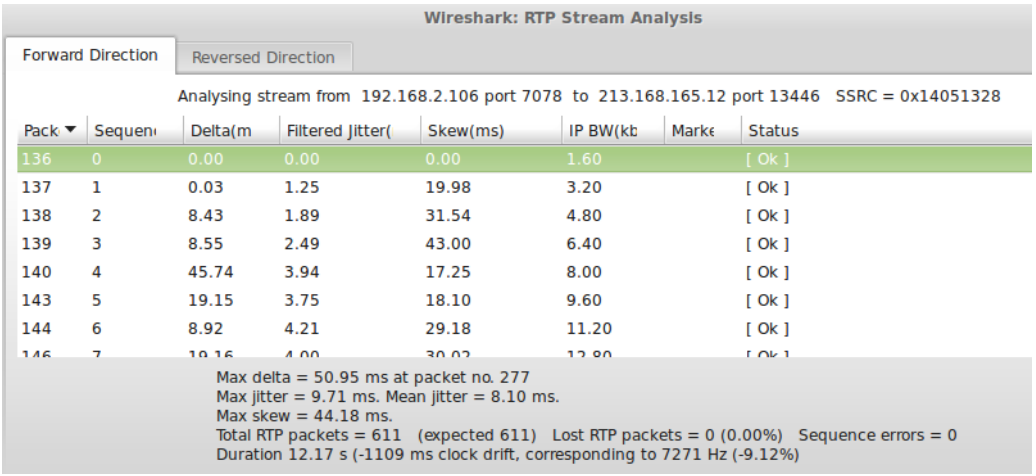
Wireshark

Na obrázku 6.1 je zachycena analýza jednoho z RTP toků ze stejného hovoru jako na obrázku 4.3. Wireshark nepočítá celkovou kvalitu RTP toku, nepočítá zpoždění ani statistiky z RTCP toku. Naopak ale obsahuje navíc ještě položky *skew* a *delta* (položka *Filtered jitter* obsahuje hodnoty pro klouzavý rozptyl). *Delta* značí hodnotu mezi příchodem aktuálního a předchozího paketu. *Skew* značí jak pozdě nebo jak předčasně přišel paket vzhledem k prvnímu paketu. Pokud máme například frekvenci 20 paketů za sekundu a 21. paket přijde v čase 60.01 s, tak *skew* bude -10 ms.

PacketScan

Výstup ze stejného hovoru jako je na obrázku 4.3 z programu PacketScan je na obrázku 6.2. Stejně jako exportér, i PacketScan dokáže ohodnotit kvalitu hovoru. PacketScan narozdíl od exportéru zobrazuje dvě hodnoty pro MOS: *Listening MOS* a *Conversational MOS*

[2] http://www.wireshark.org/
[3] http://www.gl.com/packetscan.html
[4] http://www.voipmonitor.org

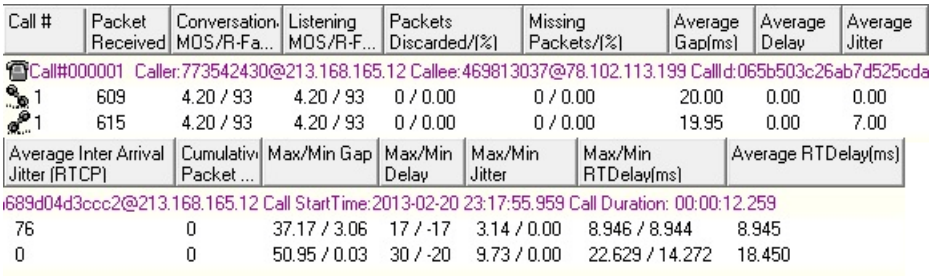


Obrázek 6.1: Analýza jednoho RTP toku ze stejného hovoru jako je na obrázku 4.3 v programu Wireshark.

(rozdíl mezi těmito hodnotami naleznete v kapitole 3.2.1). *Conversational-MOS* je počítán podle standardu ITU-T G.107[10]. Položka *Packets Discarded* značí počet paketů, které byly zahozeny kvůli omezenému jitter bufferu. *Average Gap* značí průměr hodnot, které obsahují časové rozdíly mezi příchody po sobě následujících paketů (stejný význam jako *delta* u Wiresharku).

PacketScan počítá zpoždění pro RTP tok (*delay*) jako rozdíl mezi časovou vzdáleností paketů u příjemce (*gap*) u desílatele. Pokud je např. *gap* 20,51 ms, a pakety se posílají s prodlevou 20 ms, zpoždění pro tento paket bude 0,51 ms. Jak lze vidět, PacketScan akceptuje i záporné zpoždění.

Odlišně je počítáno také RTD z RTCP paketů. Na rozdíl od exportéru, kde je RTD počítáno hlavně na základě obsahu RTCP paketů, zde je RTD mezi pakety 1 a 2 počítáno dle následující rovnice: $RTD = R2 - R1 - DLSR$. $R2$ a $R1$ značí okamžiky zachycení RTCP paketů a DLSR je položka zjištěná z paketu 2. Tento způsob ovšem nedovoluje abychom zachytili celkové RTD.



Obrázek 6.2: Analýza hovoru v programu Packetscan.

VoIPmonitor

VoIPmonitor je komplexní nástroj pro monitorování VoIP hovorů. Skládá se ze dvou komponent: analyzátoru paketů (tzv. *packet sniffer*) a webového uživatelského rozhraní. Umožňuje definovat různé senzory, kde každý je schopen monitorovat různá rozhraní. Dále je tento

program schopen nastavit určité akce vyvolané senzorem (např. upozornit pokud kvalita klesne pod určitou hodnotu).

Výstup z programu VoIPmonitor je zobrazen na obrázku 6.3. Ve výsledku jsou ukázány tři hodnoty určující MOS. *MOS fixed xx* značí kvalitu hovoru s pevně daným jitter bufferem, kde číslo *xx* označuje jeho velikost. Pokud je místo *fixed* uveden *adaptive*, jedná se o jitter buffer s proměnnou velikostí (maximální velikost je dána číslem *xx*).

	CALLER	CALLED
SIP IP	213.168.165.12 *DSCP N/A (6)*	192.168.2.106 *DSCP AF31 (26)*
SIP Port	5060	5060
RTP IP	213.168.165.12 *DSCP N/A (6)*	192.168.2.106 *DSCP CS0 (0)*
User Agent	CANISTEC-MATRIX-T100	Linphone/3.5.2 (eXosip2/3.6.0)
Codec	G.711a	G.711a
Received packets	607	610
Lost packets	0	0
MOS fixed 50	4.5	4.5
MOS fixed 200	4.5	4.5
MOS adaptive 500	4.5	4.5
Max RTP jitter	4	10
Avg RTP jitter	1	8
Max RTCP jitter	1	79
Avg RTCP jitter	0.3	76.1

Obrázek 6.3: Analýza hovoru v programu VoIPmonitor.

6.3 Nestandardní chování

Nestandardním chováním jsou myšleny situace, kdy není možné spočítat celkovou kvalitu. Ve výstupu se toto chování projeví tak, že ve sloupci obsahujícím slovní popis kvality bude řetězec „`Error`“ a výsledná hodnota MOS bude nulová. Výčet nestandardních situací:

- Pro audio byl zvolen neznámý kodek nebo jeho hodnoty pro I_e a B_{pl} nejsou k dispozici.
- Kodek byl v průběhu RTP toku změněn, v tom případě nejsem schopen zjistit jaké hodnoty pro I_e a B_{pl} použít.
- Pokud chybí opačný RTCP tok k danému RTCP toku, nejsem schopen spočítat zpoždění, a proto ani výslednou kvalitu.

6.4 Srovnání výsledků

Pro srovnání výsledků mého analyzátoru a ostatních programů byly použity dva hovory uložené ve formátu `pcap`. Každý soubor obsahoval jeden hovor, u kterého byly přítomny dva toky RTP a dva toky RTCP. Mnou vytvořený analyzátor, PacketScan a Wireshark umí analyzovat přímo `pcap` soubory. V programu VoIPmonitor byl vytvořen senzor, který

Hodnoty	**Wireshark**		**PacketScan**		**VoIPmonitor**		**Moje práce**	
RTP Jitter [ms]	8.10	0.61	7.00	0.00	8.00	1.00	7.91	0.60
RTCP Jitter [ms]	-	-	0.00	76.00	76.10	0.30	76.06	0.34
RTP Ztrátovost [%]	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
RTCP Ztrátovost [%]	-	-	0.00	0.00	0.00	0.00	0.00	0.00
RTP Zpoždění [ms]	-	-	0.00	0.00	-	-	7.00	0.00
RTCP One way delay [ms]	-	-	9.23	4.47	-	-	15.00	11.00
R-faktor z RTP toku	-	-	93.0	93.0	-	-	92.99	93.20
MOS z RTP toku	-	-	4.20	4.20	4.50	4.50	4.41	4.41
R-faktor z RTCP toku	-	-	-	-	-	-	90.09	92.90
MOS z RTCP toku	-	-	-	-	-	-	4.34	4.40

Tabulka 6.1: Srovnání výstupů jednotlivých programů. Každý sloupec je rozdělen na dvě části, kde každá představuje jeden směr RTP/RTCP toku.

Hodnoty	**Wireshark**		**PacketScan**		**VoIPmonitor**		**Moje práce**	
RTP Jitter [ms]	0.20	1.02	0.00	0.00	1.00	1.00	0.202	1.02
RTCP Jitter [ms]	-	-	1	3	3.5	2.2	3.65	2.1
RTP Ztrátovost [%]	1.50	1.20	1.50	1.20	-	-	1.48	1.21
RTCP Ztrátovost [%]	-	-	0.00	0.00	0.00	0.00	0.00	0.00
RTP Zpoždění [ms]	-	-	0.00	0.00	-	-	0.00	0.00
RTCP One way delay [ms]	-	-	2.65	0.61	-	-	41.00	1.00
R-faktor z RTP toku	-	-	93.0	93.0	-	-	80.96	82.95
MOS z RTP toku	-	-	4.20	4.20	4.10	4.20	4.06	4.13
R-faktor z RTCP toku	-	-	-	-	-	-	92.11	93.17
MOS z RTCP toku	-	-	-	-	-	-	4.39	4.41

Tabulka 6.2: Srovnání výstupů jednotlivých programů. Každý sloupec je rozdělen na dvě části, kde každá představuje jeden směr RTP/RTCP toku.

zachytával paket, jež jsem posílal na rozhraní pomocí programu `tcpreplay`. Velikost jitter bufferu v mém analyzátoru byla nastavena na 20 ms.

Výsledek tohoto srovnání je uveden v tabulkách 6.1 a 6.2. Jak lze vidět, v některých případech se výsledky liší nepatrně, jindy více.

U rozptylu je rozdílnost výsledků způsobená zaokrouhlovací chybou, ovšem takové rozdíly na výsledný R-faktor nemají skoro žádný vliv. U programu PacketScan lze pozorovat, že hodnoty pro RTP a RTCP rozptyl jsou přehozené, což je u mého analyzátoru a u programu VoIPmonitor spraveno.

Hodnoty pro RTCP zpoždění jsou u mého analyzátoru větší z toho důvodu, jelikož jej měřím celý a pracuji s hodnotami z RTCP paketů (DLSR a LSR). PacketScan pracuje s časy zachycení paketů a proto nemůže naměřit celé zpoždění mezi zařízeními, pouze zpoždění mezi sondou a koncovým zařízením.

U srovnání v tabulce 6.2 byl použit program `editcap`, který dokáže z `pcap` souborů odstranit určité pakety. Díky tomu mohla být nasimulována ztrátovost RTP paketů. Nelze pomocí něj ale ovlivnit RTCP pakety, proto stále ukazují nulovou ztrátovost. Díky tomu bude i kvalita u RTP toků nižší než u RTCP.

6.5 Shrnutí

Výstup z kolektoru byl porovnán se třemi dalšími programy. Snažil jsem se o řešení, které se dokáže vyrovnat těmto programům. Program nepracuje se statistikami z RTCP toků ani nepočítá výslednou kvalitu hovorů, dokáže pouze spočítat rozptyl a ztrátovost z RTP toků. Programy VoIPmonitor a PacketScan jsou z pohledu hodnocení kvality mnohem komplexnější programy. Mimo textový výstup dokáží kvalitu hovorů zobrazit také v grafické podobě.

Určit maximální odchylku o kterou se můj analyzátor liší od ostatních je velice složité, protože ostatní programy mají taktéž různé výsledky. Z tabulek 6.1 a 6.2 byla na stupnici MOS největší naměřená odchylka vůči programu PacketScan 4,7 % a oproti programu VoIPmonitor 2 %.

Kapitola 7

Závěr

Práci bych zhodnotil ze dvou rovin: teoretické a praktické. V teoretické části jsem se seznámil s principy VoIP technologie na takové úrovni, abych byl schopen porozumět jevům, které ovlivňují kvalitu hovorů. Prostudoval jsem několik standardů určující kvalitu hovorů a nakonec jsem vybral jednu z metod, která bere v potaz všechny faktory, které jsem schopen změřit. Kvalita byla hodnocena na základě statistik změřených ze síťového provozu, proto může docházet k situacím, kdy bude analyzátor ukazovat vysokou kvalitu, ovšem např. vlivem echa nebo příliš velkého šumu bude kvalita hovoru výrazně nižší. Je třeba si uvědomit, že objektivní metody vrací pouze odhad na stupnici MOS, kde jsou hodnoty nastaveny dle subjektivního měření.

V praktické části byly implementovány dva pluginy pro sondu FlowMon. Díky těmto pluginům jsem byl schopen monitorovat síťový provoz a tím pádem i kvalitu hovorů VoIP. Tuto část bych rozdělil na dva stěžejní body: filtrace jednotlivých hovorů a implementace vzorců pro počítání statistik.

V kapitole 6 proběhlo srovnání výsledků mého analyzátoru a ostatních programů. Programy VoIPmonitor a PacketScan ukazovaly pro stejný hovor různou kvalitu. Lze tedy vidět, že každý program využívá různé metody a hodnocení kvality hovorů VoIP na základě přenosových parametrů není jednotná záležitost. Maximální odchylka mezi mým analyzátorem a ostatními programy na stupnici MOS nepřesáhla při testování 5 %.

Dalo by se pracovat na několika zajímavých rozšířeních zpřesňujících výsledek. Nejdůležitějším rozšířením by bylo zlepšit výpočet pro zpoždění koncových bodů u RTP paketů. Dále by se dala vylepšit práce s protokolem SIP a SDP. Pokud by se ztratila například odpověď s přijetím hovoru nesoucí informace o adrese a portu pro RTP tok na vzdálené straně, mohl by analyzátor pracovat stejně jako Wireshark a rozpoznávat RTP pakety pouze na základě rozeznání jedné strany (adresy a portu).

Při reálném provozu sondy je vhodné mít na paměti umístění sondy. Pokud bude sonda příliš blízko jednomu koncovému uzlu, je pravděpodobné, že výsledky z RTP toků budou zkreslené. Jelikož hodnoty pro RTCP pakety jsou počítány přímo na koncových bodech, jsou statistiky z těchto toků obvykle přesnější.

Literatura

[1] Andersen, S.; Duric, A.; Astrom, H.; aj.: *Internet Low Bit Rate Codec (iLBC)*. RFC 3951, 2004.

[2] Baugher, M.; McGrew, D.; Naslund, M.; aj.: *The Secure Real-time Transport Protocol (SRTP)*. RFC 3711, 2004.

[3] Cisco: Quality of Service for Voice over IP [online]. 2001-04-16 [cit. 2014-04-15]. URL http://www.cisco.com/c/en/us/td/docs/ios/solutions_docs/qos_solutions/QoSVoIP/QoSVoIP.html

[4] Cisco: NetFlow Services Solutions Guide [online]. 2001-09-05 [cit. 2014-05-16]. URL http://www.cisco.com/c/en/us/td/docs/ios/solutions_docs/netflow/nfwhite.html

[5] Claise, B.; Trammell, B.; Aitken, P.: *Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information*. RFC 7011, 2013.

[6] Handley, M.; Jacobson, V.; Perkins, C.: *SDP: Session Description Protocol*. RFC 4566, 2006.

[7] ITU-T: *Methods for subjective determination of transmission quality*. P.800, 1996.

[8] ITU-T: *Mean Opinion Score (MOS) terminology*. P.800.1, 2003.

[9] ITU-T: *Single-ended method for objective speech quality assessment in narrow-band telephony applications*. P.563, 2004.

[10] ITU-T: *The E-model: a computational model for use in transmission planning*. G.107, 2011.

[11] ITU-T: *Perceptual evaluation of speech quality (PESQ): An objective method for end-to-end speech quality assessment of narrow-band telephone networks and speech codecs*. P.862, 2011.

[12] ITU-T Study Group 12: *Estimates of Ie and Bpl parameters for a range of CODEC types*. 2003.

[13] Kováč, A.; Halás, M.; Orgoň, M.; aj.: E-model MOS Estimate Improvement through Jitter Buffer Packet Loss Modelling. *Advances in Electrical and Electronic Engineering*, ročník 9, č. 5, 2011.

[14] Pennock, S.: Accuracy of the Perceptual Evaluation of Speech Quality (PESQ) algorithm.

[15] Rosenberg, J.; Schulzrinne, H.; Camarillo, G.; aj.: *SIP: Session Initiation Protocol*. RFC 3261, 2002.

[16] Schulzrinne, H.; Casner, S.; Frederick, R.; aj.: *RTP: A Transport Protocol for Real-Time Applications*. RFC 3550, 2003.

[17] Vozňák, M.: *Spojovací systémy*. Vysoká škola báňská - Technická univerzita Ostrava, 2009.

[18] Vozňák, M.: Non-intrusive speech quality assessment in simplified e-model. *WSEAS TRANSACTIONS on SYSTEMS*, ročník 11, č. 8, 2012.

Příloha A

Obsah CD

- **Adresář** `src` obsahuje adresáře `input` a `process` se zdrojovými kódy ke vstupnímu a procesnímu pluginu. Dále obsahuje adresář `lib`, kde jsou umístěny knihovny `uthash` a `utlist`.
- **Adresář** `templates` obsahuje šablony pro protokol IPFIX a program `fbitdump`.
- **Adresář** `doc` obsahuje zdrojové kódy textové části bakalářské práce pro LaTeX.
- **Soubor** `BP_Martin_Basel_2014.pdf` obsahuje textovou část bakalářské práce.
- **Soubor** `README.txt` obsahuje návod na instalaci a ukázky spuštění exportéru a kolektoru.

Příloha B

Manuál

Pro překlad a spuštění je třeba sonda FlowMon od firmy INVEA-TECH (virtuální verzi lze spustit např. ve volně dostupném programu `VMware Player`). Dále je pro překlad nutná knihovna `uthash` a pro správné zobrazení výsledků šablony pro protokol `IPFIX` a program `fbitdump`.

1. Nejdříve je třeba dodefinovat šablony pro protokol `IPFIX` a program `fbitdump`. Tyto šablony jsou umístěny v adresáři `/templates/`. Těmito šablonami je třeba nahradit stávající stávající šablony na sondě FlowMon:

   - `/etc/ipfixcol/ipfix-elements.xml`
   - `/usr/share/fbitdump/fbitdump.xml`
   - `/etc/flowmon/ipfix-template-file.txt`

2. Před startem exportéru doporučuji spustit kolektor ve *verbose režimu* příkazem:

   ```
   ipfixcol -v2
   ```

3. Poté je třeba přeložit pluginy pro exportér. V adresáři `src` je nachystán `Makefile` pro překlad. V tuto chvíli lze spustit exportér. Pro správný chod exportéru je vhodné zadat přepínač `-t`, který udává *aktivní timeout:neaktivní timeout*.

   ```
   sudo flowmonexp -t 300:60 -X process/voip_analyzator-process.so
   -I:input-voip:pcap_file=src.pcap -P:process-voip:jitter_buffer=20
   -X /home/flowmon/ipfix_export/flowmon-export-ipfix.so
   -E ipfixx:host=localhost,port=4739
   ```

   V tomto případě bude exportér číst zdrojová data ze souboru `src.pcap`. Pro čtení paketů ze síťového rozhraní (např. eth0) je třeba změnit zdroj na `pcap_if=eth0`. Dále je nutné zadat parametr `jitter_buffer` u procesního pluginu, který označuje velikost jitter bufferu.

4. Ve složce `/data/ipfixcol/` se vytvoří nový záznam. Ten se dá přečíst pomocí programu `fbitdump`. K zobrazení informací o kvalitě je třeba využít šablonu `voip`.

   ```
   fbitdump -R /data/ipfixcol/zaznam/ -o voip
   ```