České vysoké učení technické

Fakulta elektrotechnická

Katedra telekomunikační techniky

Diplomová práce

Asterisk jako monitorovací a testovací nástroj pro rozsáhlé pobočkové sítě

Asterisk as a Monitoring and Testing Tool for Large PBX Networks

Diplomant: Bc. Simona Vránová

Vedoucí práce: Ing. Pavel Troller, CSc.

květen 2022

Čestné prohlášení

Prohlašuji, že jsem zadanou diplomovou práci zpracoval sám s přispěním vedoucího práce a konzultanta a používal jsem pouze literaturu v práci uvedenou. Dále prohlašuji, že nemám námitek proti půjčování nebo zveřejňování mé diplomové práce nebo její části se souhlasem katedry.

Datum: 20.5.2022

podpis diplomanta

Poděkování

V této formě bych chtěla poděkovat Ing. Pavlu Trollerovi, CSc. za vedení diplomové práce a poskytnutí potřebných zařízení k realizaci výsledného nástroje. Dále bych chtěla poděkovat společnosti TTC Marconi za podporu během tvorby diplomové práce.

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Vránová** Jméno: **Simona** Osobní číslo: **474259**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra telekomunikační techniky**

Studijní program: **Elektronika a komunikace**

Specializace: **Komunikační sítě a internet**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Asterisk jako monitorovací a testovací nástroj pro rozsáhlé pobočkové sítě.

Název diplomové práce anglicky:

Asterisk as a Monitoring and Testing Tool for Large PBX Networks

Pokyny pro vypracování:

Za pomoci otevřeného systému pobočkové ústředny Asterisk sestavte zařízení, které bude schopno provádět automatický monitoring, měření a kontrolu v rámci velké pobočkové sítě (dráha, energetika, vojsko...). Navrhněte koncepci řešení s použitím centrálních prvků a malých distribuovaných sond v různých částech sítě. Vyberte vhodné protokoly pro komunikaci mezi centrálním prvkem a sondami tak, aby bylo možno monitorovat síť a detekovat různé druhy mimořádných stavů, zejména: výpadek spojení (sonda ztracena z dohledu), sonda viditelná, ale nemožnost navázat telefonní hovor (chyba VoIP signalizace), nesprávně navázaná média (obousměrná či jednosměrná neslyšitelnost), nízká kvalita spojení (špatná hodnota MOS – přetížená nebo chybující síť), případné další stavy specifické pro konkrétní síť a specifikované zákazníkem. Stanovte technické požadavky na fyzickou realizaci centrálních prvků (zvažte možnost virtualizace) a sond. Mějte na zřeteli nízkou nákladnost, ale přitom zachování vysoké spolehlivosti řešení. Pokuste se v laboratorních podmínkách systém realizovat a ověřit jeho funkčnost.

Seznam doporučené literatury:

[1] Van Meggelen, J.; Bryant, R.: Asterisk: The Definitive Guide, 5th Edition. O'Reily, 2019. ISBN: 978-1-49203-160-4.
[2] Clyde F. Coombs, Catherine Coombs: Communications Network Test & Measurement Handbook. McGraw Hill Professional, 1997. ISBN: 0-07136-956-2.
[3] Dokumentace dostupná na https://wiki.asterisk.org/ a http://www.asteriskdocs.org/ [online]

Jméno a pracoviště vedoucí(ho) diplomové práce:

Ing. Pavel Troller, CSc. katedra telekomunikační techniky FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **09.02.2022** Termín odevzdání diplomové práce: **20.05.2022**

Platnost zadání diplomové práce: **30.09.2023**

Ing. Pavel Troller, CSc. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Diplomantka bere na vědomí, že je povinna vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

. | Datum převzetí zadání

Podpis studentky

Abstrakt

Cílem této diplomové práce bylo vytvořit nástroj pro monitoring a testování rozsáhlých pobočkových sítí za pomoci ústředny Asterisk. Tento nástroj sleduje síťovou dostupnost prvků v síti, možnost uskutečnit hovor a kvalitu provedených hovorů. Testy probíhají automaticky. Testovací hovor je možné vyvolat také manuálně. Výsledky měření jsou zobrazeny na webové stránce centrálního prvku monitorovacího nástroje. V rámci webového rozhraní je umožněna i samotná konfigurace systému včetně plánování měření.

Klíčová slova: Asterisk, pobočkové sítě, monitoring, SIP, RTCP, MOS

Summary

This diploma thesis focuses on creation of monitoring and testing tool for large PBX networks with use of Asterisk PBX. The tool monitors: network avaibility, ability to make calls and call quality. Meassurement calls are made automatically. Test calls can be started manually. The results are shown on website of central unit. Web interface enables configuration od system including setting test times.

Index Terms: Asterisk, PBX networks, monitoring, SIP, RTCP, MOS

Obsah

1 Úvod **1**
- 1.1 Cíle práce . . . 1
- 1.2 Členění práce . . . 1

2 Technologie Voice over IP **3**
- 2.1 Ovlivnění kvality . . . 3
- 2.2 Signalizační protokoly . . . 4
- 2.3 Transport médií . . . 5
- 2.4 Zabezpečené varianty protokolů SIP a RTP . . . 6

3 Měření kvality telefonního hovoru **9**
- 3.1 Kvalita služby (QoS) . . . 9
- 3.2 Quality of Experience (QoE) . . . 9
  - 3.2.1 E-model . . . 10
  - 3.2.2 Výpočet MOS . . . 10

4 Monitorovací a testovací nástroje pro IP sítě a VoIP **13**
- 4.1 Monitorovací nástroje pro IP sítě . . . 13
- 4.2 Monitorovací nástroje pro VoIP . . . 13

5 Systémová pobočková ústředna Asterisk **15**
- 5.1 Architektura Asterisku . . . 15
- 5.2 Konfigurace kanálů . . . 16
  - 5.2.1 Obecná konfigurace kanálů . . . 17
  - 5.2.2 Kanál PJSIP . . . 17
  - 5.2.3 Kanál IAX . . . 20
  - 5.2.4 Kanál DAHDI . . . 21
  - 5.2.5 Kanál Local . . . 22
- 5.3 Konfigurace Dialplanu . . . 22
  - 5.3.1 Aplikace dialplanu . . . 23
  - 5.3.2 Funkce dialplanu . . . 24

6 Webový server **25**
- 6.1 Protokol HTTP . . . 25
- 6.2 Jazyk HTML . . . 25
- 6.3 Django . . . 26
  - 6.3.1 Konfigurace web serveru pomocí Django . . . 26

7 Jednodeskové počítače SBC **27**
- 7.1 Zařízení vhodná pro Asterisk . . . 27
  - 7.1.1 Banana Pi M2 . . . 27
  - 7.1.2 Raspberry Pi 3 model B+ . . . 28
  - 7.1.3 Orange Pi 4 . . . 28

8 Rozbor zadání **29**
8.1 Základní předpoklady . . . 29
8.1.1 Využitá infrastruktura . . . 29
8.2 Prvky v síti . . . 30
8.2.1 Nároky a cena prvků . . . 30
8.3 Obecné chování nástroje . . . 31
8.3.1 Druhy a popis mimořádných stavů . . . 31
8.3.2 Reakce na mimořádné stavy . . . 32
8.4 Využité protokoly . . . 32
8.5 Výstup . . . 33

9 Realizace **35**
9.1 Konfigurace Asterisku . . . 35
9.1.1 Konfigurace koncových bodů . . . 35
9.1.2 Konfigurace 'dialplanu' . . . 36
9.1.3 Automatické generování hovoru . . . 38
9.2 Systém vyhodnocování . . . 39
9.2.1 Síťová dostupnost . . . 39
9.2.2 Uskutečnění hovoru . . . 40
9.2.3 Parametry hovoru . . . 40
9.2.4 Reakce na mimořádnou událost . . . 41
9.3 Další využité skripty . . . 42
9.3.1 Popis skriptu automove.py a testcall.py . . . 42
9.3.2 Popis skriptu sondaPing.py . . . 42
9.4 Instalace zařízení *Pi . . . 43
9.5 Konfigurace web serveru . . . 43
9.5.1 Nastavení postranního panelu a úvodní stránky . . . 44
9.5.2 Zobrazení výsledků měření a síťové dostupnosti . . . 45
9.5.3 Přidání nové sondy . . . 46
9.5.4 Zobrazení, úpravy a odstranění sond . . . 47
9.5.5 Plánování časů automatického měření . . . 47
9.5.6 Spuštění měření . . . 48
9.5.7 Restart aplikace Asterisk . . . 48
9.5.8 Stažení logů . . . 48
9.5.9 Role pro autorizaci pro správu . . . 49

10 Testovací provoz **51**
10.1 Otestování funkčnosti nástroje . . . 51
10.2 Otestování funkčnosti webových stránek . . . 52
10.2.1 Přidání, aktualizace a odstranění sond . . . 52
10.2.2 Vyvolání testovacího hovoru . . . 52
10.2.3 Restart aplikace Asterisk . . . 52

11 Možné rozšíření **53**
11.1 Připojení k technologii TDM . . . 53
11.1.1 Popis technologie TDM . . . 53
11.1.2 Způsob připojení k systému Asterisk . . . 53
11.1.3 Měření parametrů . . . 54

11.2 Ověření správnosti měřených dat systémem Asterisk 54
11.3 Rozšíření o e-mailovou službu 54

12 Závěr **55**

Seznam obrázků

1 Vrstvy a protokoly na nich využívané pro VoIP [3] . . . 3
2 Schéma přenosu hlasového signálu VoIP [16] . . . 4
3 SIP signalizační sestavení hovoru (převzato z [3] a upraveno) . . . 6
4 Výpis SDP zprávy obsahující atributy pro šifrování mediálního toku [26] . . . 7
5 Základní přehled architektury systému Asterisk . . . 16
6 Propojení typů sekcí v kanále pjsip [8] . . . 21
7 Django - základní zpracování HTTP žádosti [32] . . . 26
8 Zařízení Banana BPi-M2 [25] . . . 28
9 Konfigurace pro kontext dispečera v rámci diaplanu . . . 38
10 Postranní panel a úvodní stránka . . . 45
11 Formulář pro přidání nové sondy . . . 46
12 Webová stránka pro zobrazení sond . . . 47
13 Webové rozhraní pro stažení logů a průběh jejich získání . . . 48
14 Webová stránka pro přihlášení . . . 49
15 Administrátorské rozhraní pro projekt Django . . . 50
16 Záchyt hovoru mezi centrálním prvkem a sondou . . . 51
17 Výstupy při SIP nedostupnosti sondy . . . 52
18 Výstupy při špatné kvalitě spojení . . . 52
19 Aktualizace obsahu konfiguračního souboru `pjsip.conf` při přidání, aktualizaci a odstranění sondy . . . 52
20 Výpis z CLI Asterisku po vyvolání hovoru . . . 53
21 Zapojení prvků pro měření . . . 54

Seznam tabulek

1 Hodnoty parametrů pro QoS třídu 1 [22] . . . 9
2 Převod mezi parametrem R-faktor a MOS [21] . . . 10
3 Parametry sekce typu ’endpoint’ . . . 19
4 Parametry pro sekci typu ’IDENTIFY’ . . . 20

1 Úvod

Vysoká spolehlivost velké pobočkové ústředny organizací jako jsou dráhy, energetika a vojsko je velmi důležitá pro jejich chod. V dnešní době většina těchto organizací využívá technologie VoIP. Jednotlivé ústředny jsou rozmístěna na vetším geografickém území. Pro přenos hovoru je tedy využita IP síť a hovor je přenášen paketově. Tento krok přinesl snížení nákladů a zvýšení mobility a flexibility. Bylo to ovšem na úkor zaručení kvality.

Služba přenosu hlasu je velmi náchylná na špatné hodnoty parametrů přenosu jako jsou zpoždění, jiter a ztrátovost paketů. Kvalita je negativně ovlivněna i využitím kompresních algoritmů.

Pro zaručení kvality služby je VoIP aplikace nutné monitorovat a průběžně vyhodnocovat jejich kvalitu. Nejčastěji je pro vyhodnocení kvality hovoru využita metrika MOS. Pro měření a vyhodnocení kvality hovorové služby existuje řada doporučení od ITU-T.

Z tohoto důvodu byly vyvinuty mnohé monitorovací nástroje. Ve většině případů se jedná o nástroje měřící kvalitu probíhajících hovorů reálného provozu. Chyba (například nízká kvalita spojení) je tedy sice detekována ihned a je možné ji řešit. Jeden hovor byl však již negativně ovlivněn a neproběhl v zaručené kvalitě. Z tohoto důvodu by bylo vhodné síť testovat a monitorovat periodicky pomocí speciálních testovacích hovorů. Tímto způsobem lze předejít negativním vlivům vyskytnutých chyb v reálném provozu. Dále tento způsob umožňuje monitorovat samotnou dostupnost jednotlivých koncových bodů.

1.1 Cíle práce

Cílem této práce je vytvořit monitorovací systém pomocí systémové pobočkové ústředny Asterisk. Jedná se monitorovací systém pro velké firmy, které mají jednotlivé pobočky rozmístěny na větším geografickém území. Tento systém má za úkol monitorovat síťovou konektivitu mezi jednotlivými prvky v síti (sondy) s centrálním prvkem. Zároveň má systém za úkol měřit kvalitu spojení mezi sondou a centrálním prvkem z hlediska kvality hovoru. Tímto způsobem bude monitorována již existující pobočková síť. Spojení centrálního prvku se sondami je realizován na existující pobočkové síti a parametry tohoto spojení tedy odpovídají parametrům spojení pro reálný provoz. Kvalita hovoru bude udávána metrikou MOS.

Změřené výsledky jsou reprezentovány na webové stránce serveru centrálního prvku. Na tomto webu je dále možné modifikovat nastavení pro měření, parametry sond a prohlížet logy. Pro přístup na web je možné vytvořit několik uživatelů s různými úrovněmi oprávnění.

Nástroj disponuje škálou automatických reakcí na sadu předem definovaných mimořádných událostí jako je výpadek sítě, nemožnost navázat SIP spojení, nebo špatná kvalita spojení. Mezi tyto reakce spadá obeznámení správce funkcionality hovorem a textovou zprávou.

Posledním krokem je otestování nástroje v rámci laboratorního prostředí.

1.2 Členění práce

Celá práce je rozdělena do několika částí. V první části práce jsou popsány teoreticky technologie a další oblasti, které je nutné znát pro realizaci projektu. Jedná se o technologie VoIP, metodiku měření kvality hovoru, popis již existujících monitorovacích nástrojů, popis web serverů a jednodeskových počítačů.

Druhá část se zaobírá rozborem zadání, formováním základních předpokladů, podle kterých se bude následně systém sestavovat. Zde se nachází popis jednotlivých zařízení v síti, využitých protokolů a předpokládané infrastruktury.

Třetí část je zaměřená na praktickou realizaci konkrétního funkčního nástroje, tedy popis konfigurace a výsledného výstupu. Tato část popisuje vytvoření konfigurace pobočkové ústředny Asterisk, tvorbu vyhodnocovacích a pomocných skriptů a vytvoření webového rozhraní.

Poslední část popisuje možná další rozšíření nástroje. Například připojení ke klasické telefonii, nebo umožnění notifikací pomocí e-mailu.

2 Technologie Voice over IP

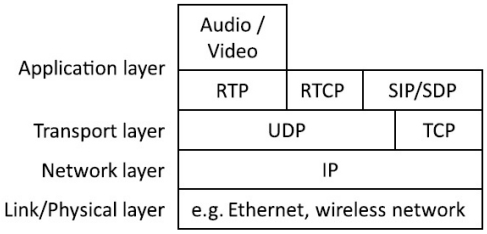
VoIP je technologie, která nahrazuje původní klasickou okruhovou telefonní síť PSTN. Primárně měl VoIP sloužit pro přenos hlasu (hlasových služeb) po paketové síti. V dnešní době nabízí škálu multimediálních služeb, například video hovory. Oproti klasické telefonní síti u VoIP nejsou vystavovány okruhy, ale hovor je posílán internetem/IP sítí na části v paketech.

Zavedení této technologie začalo původně v korporacích, kde byl VoIP levnější alternativou klasické telefonie. Zařízení potřebná pro volání pomocí VoIP jsou totiž o dost levnější. Softwarovou ústřednu je možné spustit téměř na jakémkoliv počítači, případně serveru. A jako koncová zařízení kromě IP telefonů slouží tzv. 'softphones' (softwarové telefony), tedy aplikace spustitelné na počítači. S tímto se telefonní síť stává mobilní, protože se správnou konfigurací systémové ústředny lze softwarový telefon zaregistrovat téměř odkudkoliv, kde je internetové připojení. Další výhodou je využití IP sítě, která je téměř všude přítomná. [4]

Softwarové nástroje pracující na technologii VoIP jsou například Microsoft Lync, Skype, Google Talk, Linphone a XLite. Většina těchto softwarů je bezplatná a umožňují běžnému uživateli využít VoIP pro komunikaci namísto klasické telefonie. [3]

Kromě koncových zařízení je potřeba ještě systémové ústředny, která bude hovory v síti směrovat a registrovat zařízení. Firma Cisco nabízí proprietární platformu zvanou CUCM (Cisco Unified Communications Manager), nebo je možnost využít open-source systémy jako Asterisk a Kamailio. Samozřejmě existuje celá řada dalších platforem (více v [15]).

Technologie VoIP by nebylo možné využít bez kompresních algoritmů pro hlas a video, transportních a signalizačních protokolů a monitorování kvality služby. [3]



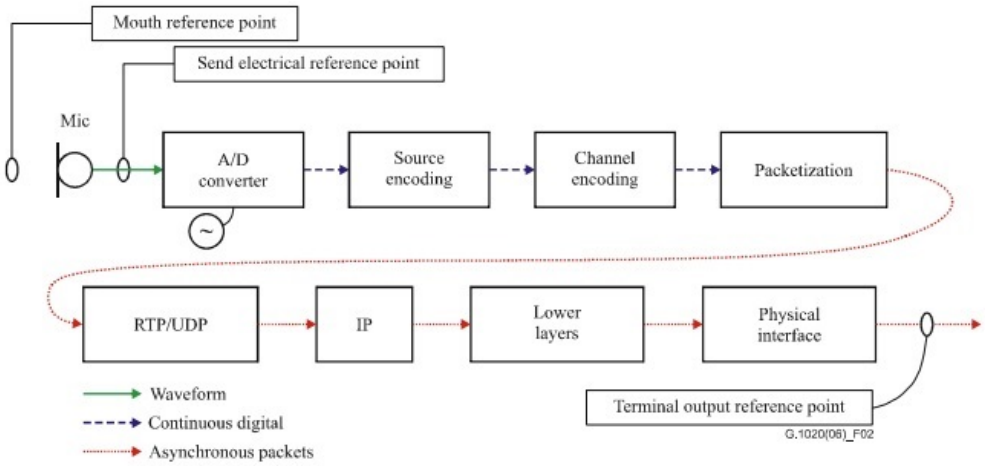
Obrázek 1: Vrstvy a protokoly na nich využívané pro VoIP [3]

VoIP funguje na fyzické vrstvě na technologiích Ethernet, WLAN a mobilních sítí. Na síťové vrstvě využívá IP protokol. Na transportní vrstvě lze využít protokoly UDP a TCP, primárně je využíván protokol UDP. Na aplikační vrstvě jsou dva druhy protokolů, pro přenos médií RTP a RTCP a pro přenos signalizace protokoly SIP a SDP. [3] Vyobrazení návazností jednotlivých vrstev a protokolů je na obrázku 1.

2.1 Ovlivnění kvality

Kvalita přenášeného hlasu IP sítí je negativně ovlivněna parametry sítě jako jsou ztrátovost paketů, zpoždění a jiter. Dále negativně kvalitu hlasu ovlivňují kompresní algoritmy.

Samotný přenos telefonního signálu se skládá z digitalizace analogového hlasového signálu. Následuje paketizace, tedy tvorba paketů z určitého počtu vzorků a přidání IP hlavičky. Sestavený



Obrázek 2: Schéma přenosu hlasového signálu VoIP [16]

paket prochází sítí ke koncovému účastníkovi. Na přijímací straně dochází k vyrovnání špatného pořadí příchodu paketů pomocí nástroje 'jitter buffer'. Tento nástroj nastavuje velikost okna a v rámci něj jsou pakety srovnány do správného pořadí. Pokud se paket nevejde do tohoto vyrovnávající okna je automaticky zahozen. Všechny tyto kroky přinášejí do přenosu zpoždění, které negativně ovlivňuje kvalitu hovoru. Z tohoto důvodu je důležité průběžně monitorovat kvalitu přenosu. [3]

2.2 Signalizační protokoly

Signalizační protokoly slouží k sestavení hovoru v paketových sítích. Signalizační protokoly jsou zodpovědné za sestavení, udržení a ukončení hovorového spojení. Mezi signalizační protokoly patří:

- H.323
- SIP
- IAX2
- SCCP
- MGCP
- proprietární protokoly moderních komunikačních platforem (Skype, WhatsApp ...) [17]

Nejrozšířenější z těchto signalizací je právě SIP, který je i standardem pro IMS sítě.

Signalizace SIP (Session Initiation Protocol) je definována v RFC 3261. Formálně vychází z protokolů HTTP, SMTP a dalších. Jedná se o textový protokol, který je velmi jednoduše analyzovatelný pomocí nástrojů tcpdump a Wireshark. Standardně se zprávy signalizace SIP posílají přes protokol UDP přes port 5060. V rámci SIP zpráv může být i protokol SDP (Session Description Protoco), který zastřešuje popis médií a vyjednávání kodeků.

SIP protokol je založen na komunikaci typu server-klient, kde se role serveru a klienta mohou měnit v rámci jednoho dialogu.

Jedna SIP relace je vytvořena a udržována v rámci dialogu. V rámci jednoho dialogu jsou uskutečňovány jednotlivé úkony (navázání spojení, ukončení spojení), které jsou označovány jako transakce (viz. obr. 3).

Klient komunikuje se serverem pomocí zpráv zvaných metody. Mezi tyto metody patří:

- INVITE
- REGISTER
- OPTIONS
- MESSAGE
- CANCEL
- ACK
- BYE

Server poté reaguje pomocí třímístného návratového kódu. Jelikož SIP vychází z HTTP jsou některé chybové kódy stejné, například '404 - Not Found'. [5]

Návratové kódy lze rozdělit do šesti kategorií podle první číslice:

- 1xx - dočasné
- 2xx - úspěšné provedení
- 3xx - přesměrování
- 4xx - chyba na straně klienta
- 5xx - chyba na straně serveru
- 6xx - obecná chyba

Na obrázku 3 je vidět klasické sestavení a ukončení hovoru. Klient posílá požadavek na vytvoření hovoru metodou 'INVITE', na kterou server odpovídá nejprve dočasnou odpovědí '180 - Ringing' a poté při sestavení spojení zprávou '200 - OK'. Pro ukončení dialogu posílá klient zprávu 'BYE' a od serveru dostává odpověď '200 - OK'.

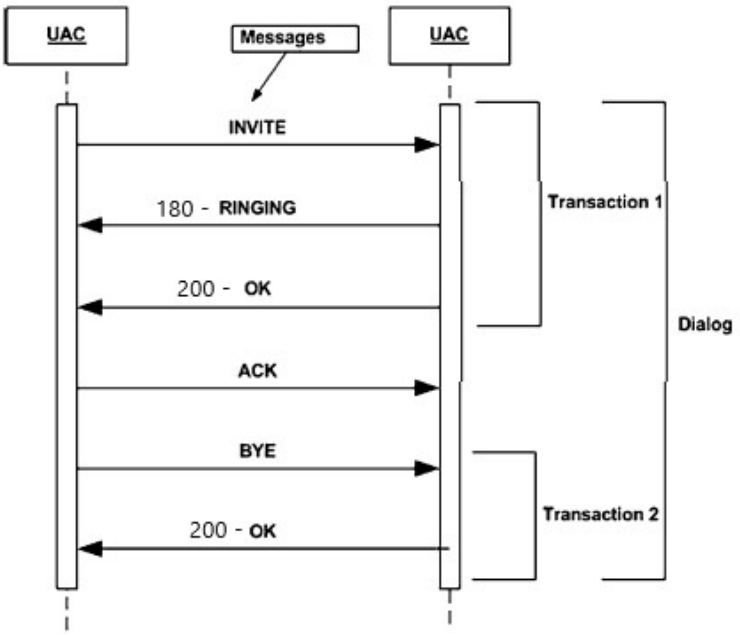
SDP protokol se používá pro definování formátu multimediální relace. Relace je mediální tok dat. Jedná se také o textově založený protokol. Je psán v řádkovém formátu, kde na každém řádku je definován typ parametru (například m pro média) a k němu je pomocí znaku '=' přiřazena hodnota. [3]

2.3 Transport médií

Transport médií se realizuje pomocí protokolu RTP a RTCP, který využívá pro transport protokol UDP. RTP protokol se stará o samotný přenos informace a RTCP protokol pomáhá s přenosem a řešením QoS/QoE. RTP přenos probíhá přes sudé porty, RTCP poté přes liché porty (port RTP + 1).

Protokol RTP je definován v dokumentu RFC 3550. Jedná se o protokol pro přenos dat v reálném čase. Velikost přenesených dat se odvíjí od velikosti hlavičky paketů a je celkem 160 bytů. Tato velikost vychází z velikosti rámce 200 bytů [22] a velikosti hlavičky paketů, která je 40 bytů.

Obrázek 3: SIP signalizační sestavení hovoru (převzato z [3] a upraveno)



Kompresní algoritmus G.729 obsahu 10 milisekund řečových vzorků v jednom řečovém rámci. Defaultně je do jednoho paketu obsaženo 20 milisekund. V rámci RTP hlavičky je definováno sekvenční číslo, podle kterého lze pakety po přijetí srovnat do správného pořadí. Pro zvětšení efektivity přenosu je možné využít komprimovanou verzi cRTP, kde je velikost hlavičky zmenšena z 40 bytů na 2-4 byty. [3]

Protokol RTCP přináší zpětnou vazbu o kvalitě přenosu. Je využíván monitorovacími nástroji pro kontrolu a management VoIP kvality. RTCP pakety jsou posílány periodicky a vyskytují se v 5 typech. V rámci RFC 3611 byl definován rozšiřující typ pro možnost hlubšího zkoumání QoS.

2.4 Zabezpečené varianty protokolů SIP a RTP

Síťová bezpečnost se stala jedním z největších problémů internetu. Řešení bezpečnostních rizik není pouze na úrovni protokolů, ale je řešen ve všech vrstvách sítě (zabezpečení fyzické infrastruktury, bezpečnost operačního systému). Protokoly SIP a RTP posílají data v otevřené formě a každý, kdo je zachytí je může lehce přečíst. Tímto způsobem lze se SIP zprávami manipulovat, nebo celý hovor přepojit jinam. Z tohoto důvodu byly vyvinuty zabezpečené varianty protokolů SIPS a SRTP. [27] [28]

SIPS je varianta SIP s rozšířením o TLS pro tvorbu zabezpečeného propojení ústředny a IP telefonu. Díky využití protokolu TLS je zaručena věrohodnost, autentizace a ochrana integrity při spojení. Přenos pomocí protokolu TLS probíhá ve dvou krocích. V prvním kroku se sestaví spojení, proběhne autentizace a vyjednaní klíče. Ve druhém kroku dochází k přenosu samotných dat šifrovaných pomocí algoritmů jako je AES. [26] [27]

SRTP využívá symetrického šifrování pomocí klíče. Tento klíč je nutné rozšířit mezi všechny účastníky využívající aktuálně službu. Klíč je mezi účastníky posílán v SIP paketech, konkrétně ve zprávách SDP. Pokud by nebyla použita šifrovaná varianta protokolu SIP, byly by klíče použité pro dešifrování informace přenášeny opět v otevřené formě. Na tomto způsobu pracuje klíčovací protokol SDES. Je tedy nezbytné při využití SRTP zašifrovat zprávy SIP. Příklad přenosu klíče pro

Obrázek 4: Výpis SDP zprávy obsahující atributy pro šifrování mediálního toku [26]

```
v=0
o=jdoe 2890844526 2890842807 IN IP4 10.47.16.5
s=SDP Seminar
i=A Seminar on the session description protocol
u=http://www.example.com/seminars/sdp.pdf
e=j.doe@example.com (Jane Doe)
c=IN IP4 161.44.17.12/127
t=2873397496 2873404696
m=video 51372 RTP/SAVP 31
a=crypto:1 AES_CM_128_HMAC_SHA1_80
inline:d0RmdmcmVCspeEc3QGZiNWpVLFJhQX1cfHAwJSoj|2^20|1:32
m=audio 49170 RTP/SAVP 0
a=crypto:1 AES_CM_128_HMAC_SHA1_32
inline:NzB4d1BINUAvLEw6UzF3WSJ+PSdFcGdUJShpX1Zj|2^20|1:32
m=application 32416 udp wb
a=orient:portrait
```

mediální komunikaci v rámci SDP zprávy pomocí parametru 'a' je vidět na obrázku 4. [26]

Dále je nutné, aby všechna zařízení podílející se na komunikaci podporovala tyto zabezpečené varianty protokolů. Pokud některé zařízení toto nepodporuje není možné sestavit bezpečné připojení. Je doporučené využívat SIPS a SRTP pokud jsou předpokládané útoky z vnějšího světa. [26] [28]

Nicméně při využívání služeb VoIP registrovaným operátorem není běžná forma zabezpečení využití šifrovaných signalizačních protokolů. Běžná forma zabezpečení je využití VPN (případně VLAN) s použitým šifrování dle standardu IPSec. Tímto způsobem je chráněna signalizace i samotná přenášená média. Operátoři jsou povinni umožnit zákonný odposlech. Ten je příslušným agenturách zpřístupněn přes speciální rozhraní. Tato data mohou být zpřístupněna nešifrovaná, nebo šifrované ve formě, aby ji druhá strana byla schopna dešifrovat.

Pokud se pohybuje v rámci pobočkové ústředny firmy je možné komunikaci zabezpečit využitím VPN. [28]

3 Měření kvality telefonního hovoru

Z výše uvedeného popisu technologie VoIP je zřejmé, že měření kvality hovoru je pro dlouhodobé uspokojení zákazníků nutné. V rámci telekomunikační techniky byly zavedeny dva pojmy pro lepší klasifikaci a stanovení kvality: kvalita služby a QoE (Quality of Experience, volný překlad Kvalita prožitku ze služby). S těmito metodami byly specifikovány i metody měření a hodnocení kvality. [3]

3.1 Kvalita služby (QoS)

Kvalita služby je spojena s parametry přenosové sítě. Při posuzování kvality služby není bráno v potaz ovlivnění kvality signálu koncovým zařízením. Měří se pouze kvalita přenosu paketu přes IP síť.

Doporučení ITI-T G.1010 zavádí tři základní metriky pro měření kvality služby a to zpoždění při přenosu, kolísání zpoždění a ztrátovost informace. Dalším kritériem pro posouzení kvality je přenosová rychlost. [3] [21]

Monitorování těchto metrik lze provádět intruzivní (aktivní) metodou a neintuzivní (pasivní) metodou. Mezi intruzivní metody patří využití protokolu ICMP pro nástroj PING, kde jsou posílány aktivně speciální pakety přímo za účelem změření jednotlivých metrik. Mezi neintruzivní metody patří záchyty síťového provozu například nástrojem Wireshark, kde jsou parametry sítě získávány přímo z provozu. [3]

Jednotlivé služby se dělí do základních 5 QoS tříd. Každá třída má specifikované hraniční hodnoty pro jednotlivé metriky. VoIP služby spadají do třídy 1 (třídy 0). Maximální hraniční hodnoty jsou shrnuty v tabulce 1. [22]

3.2 Quality of Experience (QoE)

QoE představuje celkovou kvalitu aplikace/služby vnímanou koncovým uživatelem. Tuto vnímanou kvalitu ovlivňují parametry sítě a aplikace. Mezi vlivy aplikace patří výběr kódovacích algoritmů, velikost vyrovnávací paměti pro jiter, šum na pozadí přenášeného signálu, nebo echo u analogových zařízení. [3]

Pro měření kvality aplikace jsou definovány následující metody:

- subjektivní
- objektivní
  - intruzivní
  - neitruzivní

Tabulka 1: Hodnoty parametrů pro QoS třídu 1 [22]

IPTD (zpoždění přenosu IP paketu)	IPDV (kolísání zpoždění IP paketu)	IPLR (ztrátovost IP paketu)
400 ms	50 ms	1 %

Tabulka 2: Převod mezi parametrem R-faktor a MOS [21]

Hodnota R-faktor [-]	Parametr MOS [-]	Hodnocení uživatele
90	4,3	Velmi spokojený
80	4,0	Spokojený
70	3,6	Někteří uživatelé nespokojení
60	3,1	Mnoho uživatelů nespokojeno
50	2,6	Téměř všichni uživatelé nespokojeni

Pomocí těchto metod získáváme 3 kategorie poslechové kvality: subjektivní, objektivní a odhadnutou. Subjektivní kvalita je získána statistickým vyhodnocením zaznamenané kvality hodnocené dostatečně velkým vzorkem koncových uživatelů. Objektivní metody jsou založeny na matematických modelech, které reprezentují lidský sluchový aparát. [21]

Objektivní poslechovou kvalitu lze změřit intruzivní metodou, kde je porovnáván referenční a degradovaný řečový vzorek. Pro určení kvality jsou využity nástroje pro výpočet SNR a spektrální analýza sidnálu. [3] [21]

Odhadnutá poslechová kvalita je získána neintruzivními metodami. Ty mohou být založené na matematickém modelu, který využívá parametry reprezentující jednotlivé komponenty ovlivňující kvalitu. Druhá metoda je založena na analýze degradovaného signálu. [3]

3.2.1 E-model

E-model je definován v doporučení ITU-T G.107. Jedná se o komplexní nástroj určující kvalitu hovoru mezi koncovými účastníky. E-model přiřazuje koeficient ke každému faktoru, který ovlivňuje celkovou kvalitu hovoru. Mezi tyto faktory spadá vliv šumu, hlasitosti, kvantizačního zkreslení, způsobu kódování, ozvěny a zpoždění. Je zohledněn tedy nejen telefonní kanál, ale také koncové telefonní zařízení. [21]

Výsledná hodnota reprezentující kvalitu je R-faktor a jedná se jednoduchý součet všech dílčích faktorů. Hodnoty R-faktoru se pohybují od 0 do 100. Minimální akceptovatelná hodnota pro kvalitu hovoru je stanovena na 50. [21]

3.2.2 Výpočet MOS

Metrika MOS je založena na subjektivní metodě hodnocení. Její hodnoty se pohybují na škále od 1 (špatná kvalita) do 5 (vynikající kvalita). V dnešní době se pro určení hodnoty MOS využito objektivních metod, které využívají matematické modely pro simulaci lidského požitku. [29]

Pro samotný výpočet je využit E-model a výpočet R-faktoru z naměřených metrik sítě (RTT, jiter a ztrátovost paketů). Mapování hodnot R-faktoru na MOS je vypsáno v tabulce 2.

R-faktor lze vypočítat z následujícího vztahů (převzato z [29] a [30]):

$$effectiveLatency = latency + jitter \cdot latencyImpact + compTime \tag{1}$$

$$R = 93 - \left(\frac{effectiveLatency}{factorLatencyBased} \right) \tag{2}$$

$$R = R - (lostPackets \cdot impact) \tag{3}$$

V rámci výpočtu se vyskytují 4 nastavitelné parametry udávající například dopad některých metrik na celkovou kvalitu. Tyto parametry jsou:

- latencyImpact = dopad jiteru v porovnání se zpožděním (obvyklá hodnota 2)
- compTime = výpočetní čas (obvyklá hodnota 10 ms)
- factorLatencyBased = faktor zpoždění (obvyklá hodnota 40 do zpoždění 160 ms)
- impact = dopad ztrátovosti paketů na kvalitu (obvyklá hodnota 2,5)

Tyto vztahy vychází ze standardů ITU-T P.861 a P.862. [31]
Pomocí převodního vztahu lze získat konečnou hodnotu MOS (převzato z [29] a [30]):

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

4 Monitorovací a testovací nástroje pro IP sítě a VoIP

Monitorovací nástroje pro IP sítě slouží k určení stavu sítě v reálném čase. Jsou využívány administrátory sítě pro jednoduší a efektivnější správu sítě a zařízení v ní. Monitorovací nástroje sbírají informace o síti a tyto informace následně vyhodnocují. Jedním z hlavních požadavků na monitorovací nástroj je automatická notifikace v případě výskytu chyby. [11] [12]

4.1 Monitorovací nástroje pro IP sítě

Monitorovací nástroje pro IP sítě zjišťují parametry jako jsou provoz, efektivita využití šířky pásma, doba provozuschopnosti, síťová dostupnost zařízení, přenosová rychlost a další. [11]

Monitorovací nástroje zlepšují udržení optimálního stavu sítě. Informují v reálném čase administrátora o aktuálním stavu jednotlivých zařízení a propojení. Mimo jiné pomáhají dříve detekovat bezpečnostní rizika a to zjištěním neobvyklé aktivity v síti. [11]

Využívané protokoly pro monitorování sítě jsou ICMP a SNMP. ICMP protokol se používá například pro zjištění dostupnosti zařízení. SNMP je standardní protokol pro monitorování zařízení v síti, je možné zjistit jejich dostupnost, určitá chybová hlášení a fyzické parametry jako teplota CPU. [12]

Hlavními požadavky na monitorovací nástroje jsou:

- indikátory doby provozuschopnosti
- sběr a vyhodnocování dat v reálném čase
- statistické vyhodnocování dat
- automatický výstražný systém
- automatické mapování topologie sítě
- jednoduché uživatelské rozhraní

Produktů pro monitorování sítí je mnoho. Každým rokem se musí přizpůsobovat měnícím se požadavkům na přenos a kvalitu sítě. Mezi známé a využívané platformy patří například: SolarWinds Network Performance Monitor, Auvig, Atera, Nagios a Zabbix. [13]

4.2 Monitorovací nástroje pro VoIP

Hlasové služby jsou velmi náročné na kvalitu přenosu sítě v porovnání s ostatními síťovými službami. Hlasová služba probíhá v reálném čase a je proto náchylná na zpoždění a ztrátovost paketů, které mohou negativně ovlivnit kvalitu audia. Z hlediska zaručení určité úrovně kvality služby jsou využívány monitorovací nástroje pro VoIP. Tyto nástroje sledují právě hodnoty ztrátovosti paketů, zpoždění a jiter.

Požadavky na monitorovací nástroj pro pobočkové sítě:

- měření VoIP metrik
  - jiter

  - zpoždění
  - ztrátovost paketů
- vyhodnocení naměřených dat - výpočet MOS
- automatický výstražný systém
- vysoká spolehlivost
- nízké náklady

Pro monitorování pobočkové sítě lze využít platformu SolarWinds VoIP and Network Quality Manager. Tento nástroj sleduje metriky jako jsou zpoždění, jiter, MOS a ztrátovost paketů, následně vyhodnocuje kvalitu hlasové služby. Dále je možné veškeré záznamy filtrovat podle mnoha kritérií. Nástroj má také jednoduché grafické rozhraní.

Další podobné nástroje jsou například PRTG Network Monitor, Site24x7 VoIP Monitoring, ThousandEyes, VoIP Spear a další. Ceny těchto nástrojů se pohybují od stovek dolarů po tisíce. [14]

5 Systémová pobočková ústředna Asterisk

Asterisk je open source projekt, který dokáže z obyčejného počítače vytvořit telefonní ústřednu. Jedná se o nástroj pro tvorbu komunikačních aplikací.

Asterisk sám o sobě je komunikační server, který má velkou škálu využití. Původně vytvořený jako telefonní systém pro malé podniky, dnes může fungovat jako daleko více. Primárně je Asterisk platforma pro VoIP komunikaci, díky instalaci dalších komponentů je možné propojit Asterisk s klasickými signalizacemi jako je například SS7.

Jak již bylo řečeno Asterisk je možné nainstalovat na běžný počítač nebo server. Podporované a testované distribuce operačního systému jsou Linux a Unix jsou CentOS, RHEL, Fedora, Ubuntu a Debian. Ústřednu lze plně virtualizovat.

Kromě systému Asterisk lze naistalovat verzi nazvanou FreePBX, která obsahuje grafické rozhraní pro konfiguraci z webu.

Asterisk podporuje celou řadu VoIP signalizací: SIP, H.323, IAX2, MGCP, SCCP. Dále je zaručena podpora velké škály audio, video, obrazových a textových kodeků. [1] [7]

5.1 Architektura Asterisku

Systém Asterisk je velmi komplexní program s mnoha komponenty. Jejich základní dělení a vztahy jsou vyobrazeny na obrázku 5. Jádro Asterisku interaguje s ostatními moduly a je zodpovědné za načtení konfiguračních souborů. Jednotlivé moduly zastřešují různé funkcionality systému. Například jeden modul (DAHDI) umožňuje systému komunikovat s analogovými telefony.

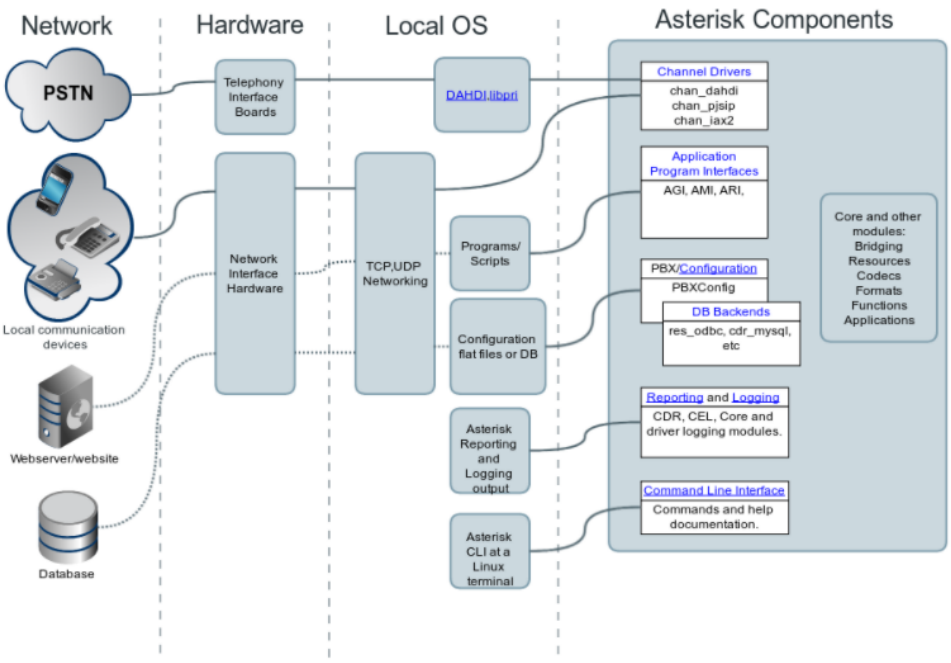
Základem systému Asterisk je jeho jádro. Jádro je zodpovědné za načítání dalších modulů a konfiguračních souborů včetně 'dialplanu'. Jádro nabízí základní funkcionalitu a pro její rozšíření je potřeba načíst další moduly. Všechny moduly jsou uloženy ve složce `/usr/lib/asterisk/modules`.

Většinu modulů je možné konfigurovat ze speciálních konfiguračních souborů, které jsou uloženy ve složce `/etc/asterisk`. Některé moduly umožňují načítání konfigurace z databáze.

Moduly se dělí na několik typů. Jeden z typů modulů je pro ovladače kanálů, pomocí kterých komunikuje jádro Asterisku s dalšími zařízeními, jak je vidět i na obrázku 5. [7]

Nejběžnější typy modulů jsou:

- moduly pro kanály ('Channel drivers')
- aplikace dialplanu
- funkce dialplanu
- zdroje - například hudba na pozadí
- kodeky
- přemosťovaní ovladače - propojení mezi signalizacemi ('bridge drivers') [7]



Obrázek 5: Základní přehled architektury systému Asterisk

Veškeré moduly, které jsou nainstalovány na systému Asterisk, lze nechat vypsat pomocí příkazu `module show` v příkazové řádce pro systém Asterisk. Konkrétní počet modulů u verze, kterou používám, je 350. [36]

Administrátor aplikace může ovlivnit, které moduly jsou načítány. Způsob načítání modulů lze ovlivnit pomocí konfigurace souboru `modules.conf`. Automatické načítání modulů je nastaveno následovně:

```
[modules]
autoload=yes
```

Nastavení pro nenačítání konkrétních modulů je umožněno pomocí příkazu `noload` a to následovně:

```
noload = chan_sip.so
```

Načítání modulů lze ovládat i z příkazové řádky pomocí příkazů:

```
CLI> module load | unload | reload <jméno modulu>
```

5.2 Konfigurace kanálů

Systém Asterisk obsahuje velké množství kanálů pro propojení různých platforem mezi sebou. Tyto kanály prezentují jednotlivé koncové body, se kterými je možné komunikovat. Jednotlivé druhy kanálů se liší způsobem komunikace, tedy signalizačním protokolem, které koncové zařízení podporuje.

Většina kanálů bývá dlouhodobě podporovaná. To neznamená, že se nevyhnou změnám. Některé kanály jsou nahrazovány novými variantami. Například sip byl nahrazen kanálem pjsip a zaptel byl nahrazen kanálem DAHDI. [9]

5.2.1 Obecná konfigurace kanálů

Většina kanálů je konfigurovatelná ze speciálních konfiguračních souborů. V těchto souborech se definují jednotlivé koncové body a trunk spojení, nikoli samotné chování při hovoru. To je konfigurováno v rámci 'dialplanu'.

Konfigurační soubory jsou psány v prostém textu a je možné je editovat v klasickém textovém editoru jako jsou Vim a Nano. Konfigurační soubory mají vždy stejnou strukturu. Název sekce je v hranatých závorkách, neměl by obsahovat mezery a rozlišují se velká a malá písmena. Pod názvem sekce jsou nastavovány hodnoty k parametrům ('settings'). Hodnoty jsou přiřazovány pomocí symbolu '='. [1]

```
[název sekce]
nastavitelný parametr=hodnota
```

Pro usnadnění konfigurace je možné využít šablony. Na začátku souboru se definuje šablona, která je poté použitá v rámci souboru. Šablona musí být vždy definována před jejím využitím. [7]

```
[název šablony](!)
Parametr šablony=hodnota
Parametr šablony2=hodnota

[název sekce](název šablony)
Další parametr=hodnota
```

Řádek je možné zakomentovat pomocí znaku '!'.

```
! toto je komentář
nastavitelný parametr=hodnota
```

5.2.2 Kanál PJSIP

PJSIP je samostatný projekt, který nespadá pod firmu Digium. Jedná se o knihovnu pro multimediální komunikaci psanou v jazyce C. Zastřešuje standardní protokoly SIP, SDP, RTP, STUN, TURN a ICE. Tato knihovna byla do projektu Asterisk přidána ve verzi 12 a v dnešní době již nahrazuje původní komunikační kanál pro SIP. Ve verzi Asterisk 18.7.1 je verze pjsip 2.10. [39]

Kanál PJSIP pracuje na signalizačním protokolu SIP a tedy na technologii VoIP. Jeho primárním zaměření je konfigurace pro softwarové telefony, IP telefony a SIP trunk (propojení dvou systémových ústředen).

Sipové kanály načítány z modulu chan_pjsip jsou konfigurovatelné v souboru pjsip.conf. Pro další funkce se musí načítat další moduly.

V rámci kanálu pjsip je konfigurace rozdělena do sekcí na rozdíl od původního sip kanálu. Typ sekce není nutné uvádět v názvu sekce, ale je nutné, aby byl specifikován v parametru 'type'. Názvy sekcí se mohou shodovat pokud je jejich typ rozdílný. V rámci každé sekce pak existuje sada parametrů, které je možno nastavit. Každá sekce musí mít definovaný parametr type.

Tyto typy jsou načítány z různých modulů. Typy jednotlivých modulů je možné nechat vypsat z příkazové řádky Asterisku pomocí příkazu `config show help modul` (například `config show help res_pjsip`).

Každá sekce má svoji funkci a sadu parametrů, které je možné nastavit. Níže jsou vypsány používané typy sekcí. Některé sekce jsou na sobě přímo závislé, tyto závislosti jsou také popsány níže. [7] [8]

Typ ENDPOINT

Typ 'endpoint' je primární konfigurační objekt. Tento typ není možné používat sám o sobě, aby se stal volatelným, je nutné nastavit ještě typ AoR ('Addresses of Record'). Pro definici nového objektu je nutné mít nastaveny dvě sekce typů 'endpoint' a 'aors'. Tento typ by se dal nazvat jako profil pro koncový bod jako je telefon a server.

Některé nastavitelné parametry jsou vypsány v tabulce 3 a seřazeny podle četnosti použití. Celkový počet nastavení je 135. Zároveň je v tabulce popis funkce jednotlivých parametrů a typ hodnoty, který může mít parametr přiřazen. Dále je v tabulce uvedena i defaultní hodnota pro parametry, kde je nastavena. [8]

Typ TRANSPORT

Tento typ definuje způsob přenosu zpráv. Nastavujeme protokol transportní vrstvy pro SIP provoz, IP adresu a port. Podporované protokoly jsou UDP, TCP, and WebSocket (WS). Lze nastavit i zabezpečenou komunikaci pomocí protokolu TLS. V rámci jedné konfigurace je možné nastavit více transportů, je nutné, aby se žádný neduplikoval ve všech parametrech. Zároveň musí být definovaná alespoň jedna sekce pro transport. Po změnách v tomto typu je nutné Asterisk restartovat, aby byly změny načteny.

V minimální konfiguraci specifikujeme:

- type=transport
- protocol= udp | tcp | tls | ws | wss | flow
- bind=W.X.Y.Z

Konfigurace pro transport přes protokol UDP vázaný na všechny IP adresy je vidět zde:

```
[transport-udp]
type=transport
protocol=udp
bind=0.0.0.0
```

Pro konfiguraci průchodu hovoru přes NAT je nutné specifikovat další parametry:

- local_net
- external_media_address
- external_signaling_address

Typ AUTH

Pokud je potřeba nastavit heslo pro přihlášení koncového bodu, je nutné nastavit typ sekce 'auth'. V rámci autorizace se vybírá způsob autentizace a kredence k dané autentizaci. Přiřazení dané sekce s autorizací k sekci 'endpoint' je nastaveno v sekci 'endpoint'.

Běžnou možností pro autentizaci je možnost 'userpass'. [8]

Tabulka 3: Parametry sekce typu ’endpoint’

Parametr	Popis
Potřebné parametry pro vytvoření a autorizaci objektu	
type	musí být nastaven na: endpoint
aors	název sekce AOR, která definuje daný objekt
auth	název sekce AUTH použité pro autorizaci objektu
context	odkaz na kontext v rámci dial planu v souboru `extensions.conf`
Parametry pro nastavení kodeků a médií	
disallow	zakázání určitých nebo všech kodeků
allow	povolení určitých kodeků
direct_media	nastavuje přímá média mezi koncovými body typ boolean, defaultní hodnota: ’yes’
direct_media_method	způsob jakým se nastaví přímá média mezi koncovými body možné hodnoty: ’invite’, ’reinvite’ a ’update’, defaultně: ’invite’
rtp_symmetric	RTP proud musí být symetrický typ boolean, defaultní hodnota: ’no’
rtp_keepalive	nastavuje časový interval pro posílání RTP zpráv typ: integer, defaultní hodnota: 0
rtp_timeout	maximální doba po kterou je udržován hovor bez RTP zpráv typ: integer, defaultní hodnota: 0
Parametry pro nastavení IP adres	
deny	seznam zakázaných IP adres
permit	seznam povolených IP adres
Další parametry	
100rel	podpora rozšíření pro 100rel možné hodnoty: ’yes’, ’no’ a ’required’, defaultně: ’yes’
callerID	informace o ID volajícího nutno zadávat ve formátu: jméno <číslo> nebo <číslo>
dtmf_mode	nastavení pro detekci DTMF volby dafaultní hodnota: ’rfc4733’
language	nastavení jazyka pro daný kanál
allow_tranfer	povolení pro předání hovoru pře SIP REFER typ: boolean, defaultní hodnota: ’yes’
ice_support	povoluje užití mechaniky ICE pro průchod přes NAT

Tabulka 4: Parametry pro sekci typu 'IDENTIFY'

Parametr	Popis
type	musí být nastaven na: identify
match	IP adresa anebo síť pro porovnání
svr_lookups	Provede SVR vyhledávání pro poskytnuté hostname Hodnota boolean, defaultně na: yes
match_header	Dvojice hlavička a hodnota pro porovnání ve tvaru: 'match_header = SIPHeader: value'
endpoint	Název sekce 'endpoint', kterou indentifikuje

Typ AOR

Typ 'aor' umožňuje Asterisku kontaktovat definovaný koncový bod. Zde se nastavuje, kde je daný objekt k nalezení. Zároveň umožňuje registraci více zařízení pod jednu registraci. Maximální počet zařízení je nastaven pomocí parametru 'max_contacts', který je v základním nastavení roven nule, je tedy nutné tento parametr nastavit na jinou hodnotu. [8]

Typ REGISTRATION

Tato sekce umožňuje registraci k dalším SIP serverům a je zcela oddělena od ostatních sekcí v konfiguraci. Pro minimální nastavení je potřeba nastavit 'server_uri' a 'client_uri'. [8]

Typ IDENTIFY

Tento typ sekce identifikuje koncové body na základě daných kritérií, nejčastěji IP adresy. Pokud je definováno více kritérií stačí, aby koncový bod splnil alespoň jedno. Sekce typu 'IDENTIFY' je přímo propojena s jednou sekcí typu 'endpoint'. [8]

Propojení jednotlivých typů sekcí je zobrazeno na obrázku 6. Pokud je čára na obrázku zakončena prázdným kolečkem znamená to, že tento typ sekce není potřeba pro to, aby se daná sekce nastavila. Naopak sekce typu 'endpoint' a 'aor' jsou na sobě závislé a pro funkční konfiguraci musí být nastaveny obě. Z obrázku je také patrné, že v rámci sekce 'identify' musíme odkazovat na sekci 'endpoint' stejně jako v sekci 'endpoint' specifikujeme sekce 'aor' a 'auth'. [8]

5.2.3 Kanál IAX

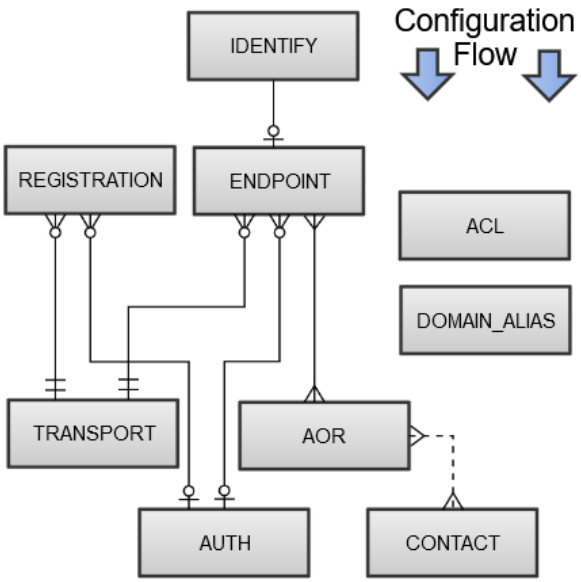
IXA2 (Inter-Asterisk eXchange protocol, version 2) je kanál založen pro VoIP komunikaci právě na protokolu IAX verze 2. Tento protokol byl vyvinut přímo pro Asterisk. Na rozdíl od ostatních VoIP kanálů se IAX kanál používá primárně pro trunkové spojení.

V dnešní době není již modul pro kanál vyvíjen, ale má stále aktivní podporu. Pokud by nastal problém bude vyřešen, ale samotný vývoj byl již zastaven. [7]

Mezi hlavní výhody využití IAX kanálu jsou:

- snadný průchod skrz NAT, PAT a firewall
- malá velikost hlavičky - 4 byty
- mezinárodní podpora - využití rodných jazyků uživatelů

Obrázek 6: Propojení typů sekcí v kanále pjsip [8]



- podpora autentizace pomocí MD5 a RSA
- multimediální protokol - podpora obrazu, videa, textu a dalších
- sběr statistických údajů o hovoru - například o zpoždění

.

Hlavní rozdíl od ostatních VoIP protokolů je, že se signalizační a mediální tok posílají přes jednu IP adresu a port pomocí UDP. Toto ulehčuje průchod NATem.

Pokud chceme propojit 2 Asterisk pobočkové ústředny, je nutné nakonfigurovat registraci na druhou ústřednu a vytvořit pro ni registraci.

5.2.4 Kanál DAHDI

DAHDI kanál slouží k připojení Asterisku do klasické telefonní sítě. V tomto případě je potřeba hardwarového rozšíření jako je E1 karta, kterou firma DAHDI nabízí. Kanál DAHDI nahradil kanál Zaptel. Pro použití je nutné při instalaci Asterisku nainstalovat ještě balíček DAHDI.

V rámci DAHDI kanálu je například možné využít knihovnu pro SS7 signalizaci 'libss7'. Tato knihovna již není v dnešní době vyvíjena.

Samotné kanály se nastavují v konfiguračním souboru `chan_dahdi.conf`.

V rámci tohoto kanálu lze využít následujících signalizací: E & M, FXS, FXO, PRI, BRI, E911, MFC/R2 a SS7. Signalizaci je možné specifikovat v rámci natavení kanálu. Defaultně je nastaven na auto.

Základní konfigurace by mohla vypadat následovně:

```
[channels]
context=public
signalling=auto
usecallerid=yes
callwaiting=yes
echocancel=yes
group=1
callgroup=1
pickupgroup=1
```

Každá signalizace má vlastní specifické parametry k nastavení. Jejich bližší popis lze získat ze vzorového souboru vytvořeného při instalaci Asterisku. [9]

5.2.5 Kanál Local

Lokální kanál přináší možnost volat zpět na Asterisk. Vytváří takovou smyčku zpět do 'dialplanu'. Na rozdíl od ostatních kanálů tedy nepropojuje Asterisk s jinými zařízením pomocí daného protokolu a neexistuje pro něj konfigurační soubor.

Od verze Asterisku 12 je kanál Local součástí jádra Asterisku a je načítán ihned při spouštění Asterisku.

Dalším rozdílem od ostatních kanálů je, že není konfigurován v rámci souboru, ale je rovnou využíván v rámci 'dialplanu'.

Příklad využití lokálního kanálu může být postupné prozvánění telefonů. Například po 30 sekundách postupně obvolávat všechny telefony infolinky. [1]

5.3 Konfigurace Dialplanu

V rámci konfiguračních souborů pro komunikační kanály lze nastavit pouze koncové body na systému Asterisk nikoliv však jejich chování při vytváření hovoru a průběhu. Za celou logikou směrování v rámci Asterisku stojí tzv. 'dialplan'. Je konfigurovatelný ze souboru `extensions.conf`, kde je definováno mimo jiné chování pro příchozí a odchozí hovory.

Konfigurační soubor pro 'dialplan' se jmenuje `extension.conf` a nachází se ve složce s ostatními konfiguračními soubory `/etc/asterisk`. Je možné ho psát v zabudovaném skriptovacím jazyce, ve formátech AEL nebo LUA.

Konfigurační soubor je rozdělen do několika sekcí. První dvě sekce definují proměnné. V rámci sekce `[General]` se nastavují předem dané parametry například: 'static', 'writeprotect', 'autofallthrough'. V rámci sekce `[Global]` lze definovat proměnné, které lze využít dále v rámci 'dialplanu'. Globální proměnné lze nastavit i v jiných sekcích pomocí aplikace 'Global()'. Další sekce zastřešují jednotlivé číslovací plány (tzv. 'extensions') se nazývají kontexty. Je zvyklostí si pomocí těchto kontextů rozdělovat typy provozů například na lokální a příchozí. Jednotlivé kontexty je možno přidávat do jiných pomocí příkazu `include`.

Jednoduchá konfigurace pak může vypadat následovně:

```
[General]
static=yes
writeprotect=no

[local]
exten => 1234,1,Aplikace()
```

```
[incoming]
include => local
```

Samotné 'extensions' reprezentuje volané jméno nebo číslo. Může obsahovat čísla i písmena. Velmi často je využíváno číselných vzorů, které vychází z regulérních výrazů. Pro rozeznávaní vzorů od ostatních čísel je na začátek čísla přidán znak podtržítka (_).

Pro hledání shody vzoru je použito následujících výrazů:

- X - reprezentuje libovolné číslo v rozmezí 0-9
- Z - reprezentuje libovolné číslo v rozmezí 1-9
- N - reprezentuje libovolné číslo v rozmezí 2-9
- `[1247-9]` - reprezentuje jedno z čísel v závorce
- `[a-z]` - reprezentuje jakékoliv malé písmeno abecedy
- `[A-Z]` - reprezentuje jakékoliv velké písmeno abecedy
- · - reprezentuje libovolné jedno nebo více čísel

Dále existuje sada speciálních znaků, které mají speciální určení - i, h, t. Znak 'h' značí úkony, které jsou vykonány po ukončení hovoru.

V rámci jedné 'extension' lze definovat několik po sobě jdoucích kroků, jak hovor zpracovat. Pořadí kroků je dané pomocí priority `exten => 100,priorita,Aplikace()`. V následujícím příkladu přichází hovor na číslo 100. Hovor je vyzvednut, čeká se 1 sekundu a přehraje se zvuková stopa 'hello' a hovor je ukončen. 'Extension' musí vždy začínat číslování priorit od 1.

```
[local]
exten => 100,1,Answer()
same => n,Wait(1)
same => n,Playback(hello)
same => n,Hangup()
```

5.3.1 Aplikace dialplanu

Aplikace umožňují vyvolávat určité akce na hovor definovaný v `extension.conf`. Jednotlivé aplikace je možné rozřadit podle jejich zaměření, tedy podle toho, co dělají a ovlivňují.

Aplikace pro správu hovoru přímo ovlivňují, jak se bude hovor chovat. Jedná se o aplikace pro přijetí, přepojení a ukončení hovoru.

Aplikace 'Answer()' vyzvedává hovor na straně Asterisku. Aplikace 'Hangup()' naopak hovor ukončuje. Dále jsou zde aplikace 'Busy(x)' a 'Congestion(x)' pro indikaci určitého stavu kanálu.

Asi nejdůležitější aplikace pro využití Asterisku jako PBX je aplikace 'Dial'. Tato aplikace vytváří kanál na určité číslo přes určitou technologii a tím spojuje samotný hovor. V rámci této aplikace je možné vytvořit vícero hovorů pomocí znaku '&'. Samotná aplikace obsahuje další sadu nastavení například pro maximální dobu hovoru nebo pro předání hovoru. Samotná syntaxe vypadá následovně:

```
Dial(technologie/číslo & technologi/číslo|timeout|nastavení)
```

Mezi obecné aplikace patří například aplikace 'Wait()', která nastavuje dobu čekání v rámci 'dialplanu'.

V rámci instalace se stahuje i sada zvukových souborů, které je možno nechat přehrát pomocí aplikace 'Playback(zvukový soubor)'.

Aplikace 'NoOp(text)' je využívaná pro odladění programu, vypisuje nastavený text do příkazové řádky Asterisku.

Aplikace 'AGI' umožňuje v rámci 'dialplanu' spustit externí program, například Python skript. Skript lze spustit se sadou vstupních parametrů, které byli získány například pomocí funkce 'dialplanu'. Základní syntax aplikace je následovný:

```
AGI(command,arg1,[arg2[,...]])
```

Aplikace 'MessageSend' se stará o posílání textových zpráv koncovým bodům. Tato aplikace je svázána s funkcí 'MESSAGE'. Pomocí funkce 'MESSAGE(Body)' lze nastavit nebo předat text zprávy. Dalšími parametry funkce jsou příjemce a odesílatel. Tyto dva parametry lze specifikovat i jako argumenty aplikace. U příjemce musí být specifikovaná technologie, kterou používá (například pjsip).

```
MessageSend(destination,[from,[to]])
```

5.3.2 Funkce dialplanu

Funkce 'dialplanu' jsou podobné jeho aplikacím. Na rozdíl od aplikací nemanipulují přímo s vytvořeným kanálem, ale s poskytnutými daty. Funkce jsou v rámci konfiguračního souboru psány velkými písmeny.

Existuje například funkce 'VERSION', která poskytuje informaci o konkrétní verzi Asterisku (číslo verze, OS na kterém Asterisk běží ...). Toto lze využít například při tvorbě logů. Další důležitou funkcí pro logování je 'STRFTIME(epoch,timezone,format)', která získává aktuální čas serveru ve specifikovaném formátu. Pomocí funkce 'TIMEOUT' lze nastavit časový odpočet pro odpověď, nebo zadávání čísel. Další funkce 'CALLERID' umožňuje získat nebo nastavit ID volajícího.

Informace o proběhnutém hovoru jsou dosažitelné pomocí funkce 'CDR(argument)'. Zadáním argumentu lze získat informace o délce hovoru ('billsec'), název kanálu ('channel') a konečný stav hovoru ('disposition').

Nejvíce informací o kanálu lze získat pomocí funkce 'CHANNEL(parametr)'. Tento kanál obsahuje sadu argumentů, které jsou dostupné pro všechny technologie. Poté existuje sada specifických argumentů pro konkrétní technologie.

Pro kanál PJSIP existuje celá sada argumentů pomocí, kterých lze získat parametry přenosu (RTT, ztrátovost paketů a jiter). Lze například zjistit adresy odkud kam proudily RTP pakety.

6 Webový server

Pojem webový server zahrnuje dva významy. V první řadě je to počítač (hardware), na kterém jsou uloženy soubory tvořící webovou stránku (HTML soubory, obrázky, CSS soubory, skripty). Zároveň na tomto hardwaru běží software webový server. Webový server jako software (například HTTP server) zpracovává požadavky klienta a posílá zpět odpovědi s obsahem webové stránky. [19]

Podle druhu obsahu webových stránek dělíme webové servery na:

- statický
- dynamický

Pokud webový server obsahuje pouze neměnné webové stránky je nazýván statickým. Pokud je obsah stránky aktualizován například skriptem je tento server nazýván dynamický. V případě dynamického webu je nejčastěji použita HTML šablona, která je vyplněna daty z databáze. [19]

Server s klientem komunikují pomocí protokolu HTTP. Samotný obsah webových stránek je psán v jazyce HTML.

6.1 Protokol HTTP

Protokol HTTP je internetový protokol využitý pro komunikaci mezi klientem a serverem. Klientem je ve většině případů internetový prohlížeč.

Komunikace je založena na principu dotaz-odpověď. Klient posílá dotaz na server a serve odpovídá. Součástí odpovědi je i tzv. stavový kód. Ten má podobu třímístného čísla, které udává v jakém stavu požadavek skončil.

V dnešní době je standardem zabezpečená verze protokolu HTTPS. [41]

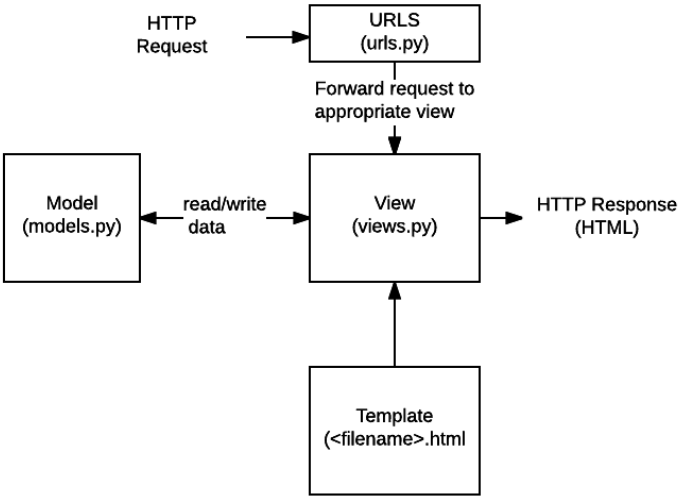
6.2 Jazyk HTML

Pro tvorbu webových stránek je využívaný jazyk HTML. HTML soubor tvoří nejzákladnější stavební prvek webových stránek. V rámci souboru je nutné specifikovat typ dokumentu. Dále se dokument dělí na hlavičky a tělo stránky. HTML je značkovací jazyk a jednotlivé značky určují, která část těla je nadpis, paragraf, tabulka atd.

Značka pro hypertextový odkaz je `<a href="odkaz» </a>`.

Základní struktura HTML dokumentu je následující:

```
<!DOCTYPE HTML>
<html>
    <head>
        ...
    </head>
    <body>
       ...
    </body>
</html>
```



Obrázek 7: Django - základní zpracování HTTP žádosti [32]

V rámci HTML dokumentu je specifikován i odkaz na soubory `css`, které definují styl stránky. [40]

6.3 Django

Django je webový framework psaný v jazyce Python. Představuje nástroj pro tvorbu webových aplikací. Zajišťuje rychlý vývoj webových aplikací, bez nutnosti definovat základy. Celý projekt je open source a zdarma. Mnoho velkých webových stránek pracuje právě na platformě Django (Instagram, Mozzila).

Mezi hlavní výhody Djanga je univerzálnost (možnost využití mnoha formátů HTML, JSON, XML), zabezpečení koncové aplikace, škálovatelnost a přenosnost (Django běží na mnoha platformách - Linux, Windows, MacOS).

Základní princip zpracování HTTP žádosti je zobrazen na obrázku 7. Po přijetí žádosti je požadované URL namapováno na konkrétní obraz ('View'). Tento obraz je seskládán z konkrétních modelů a šablony a je vrácen jako odpověď'.

Modely jsou Python objekty, které definují strukturu aplikačních dat a přináší metody pro jejich správu. Šablona je soubor definující rozložení stránky. [32]

6.3.1 Konfigurace web serveru pomocí Django

V rámci Django projektu lze definovat několik aplikací pro rozlišení jednotlivých projektů. V rámci aplikací jsou definovány modely, které reprezentují objekty potřebné v aplikaci. Django pracuje se systémem databází a jejich obsluha je umožněna pomocí python skriptů. Jednotlivé databáze jsou reprezentovány právě modely.

Samotné stránky jsou definovány ve třech souborech: 'view.py', 'urls.py' a html šablona.

Pro grafickou úpravu je možné použít sadu nástrojů nabízené z nástroje Bootstrap. Bootstrap je nástroj pro využití CSS a html šablon. Tento nástroj je zdarma a open-source.

7 Jednodeskové počítače SBC

Jednodeskový počítat poskytuje veškeré možnosti klasického počítače integrované na jedné desce. Jejich malá váha a velikost s nízkou cenou je činní perfektní volbou pro monitorovací zařízení a pro jednoduché kontrolní procesy. [20]

Mezi nejrozšířenější jednodeskové počítače patří zařízení Raspberry Pi nebo Arduino. Do stejné rodiny zařízení spadá i Banana Pi a Orange Pi.

Zařízení Raspberry Pi jsou velmi rozšířená i v rámci telekomunikační techniky. Tato zařízení nemají problém fungovat jako pobočkové ústředny Asterisk. Podle [10] je vidět, že zařízení nemělo problém s obsluhou několika simultánních hovorů. Další testování pořízené v [23] ukazuje, že novější zařízení dokáží se správnou volbou kodeků obsloužit i 150 současně vedených hovorů. Lze tedy konstatovat, že samotné zařízení může fungovat jako pobočková ústředna pro menší podniky.

Operační systém zařízení je tzv. Raspbian, což je odlehčená verze Debianu. Existuje projekt, který kombinuje operační systém Raspbian s instalací Asterisk a FreePBX.

Pro větší integraci do komunikačního světa byli vyrobeny moduly, které rozšiřují klasické Raspberry Pi o porty pro připojení do sítě PSTN. Jedno z řešení nabízí 4 porty FXO a 4 porty FXS. Ceny těchto modulů se pohybují v rozmezí 100-350 dolarů v závislosti na počtu portů a přidání modulu pro potlačení echa. [24]

7.1 Zařízení vhodná pro Asterisk

V této části budou představeny různé modely zařízení Pi, které jsou vhodná pro spuštění aplikace Asterisk.

Nároky pro malý systém (do 5 koncových zařízení) jsou následující [9]:

- Procesor: 400 MHz x86
- RAM: 256 MB
- ethernetový konektor

Pro základní konfiguraci je potřeba také výstup pro USB a display port.

Maximální paměť je dána vloženou microSD kartou a tedy i maximální podporovanou velikostí pro microSD karty, obvykle se jedná o hodnotu 64 GB.

Ceny těchto zařízení se pohybují v jednotkách tisíc korun.

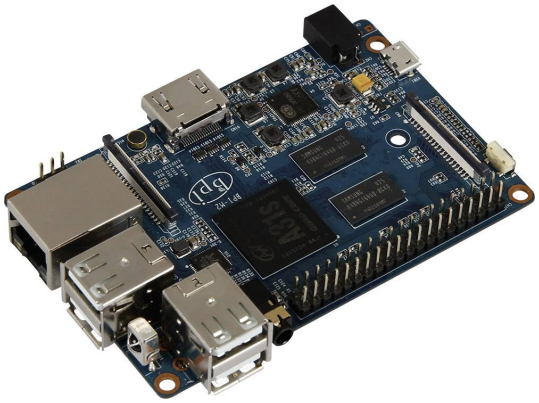
Tyto požadavky splňují modely všech značek Raspberry Pi, Orange Pi i Banana Pi. Níže je popsán jeden model každé značky.

7.1.1 Banana Pi M2

Banana Pi je jednodeskový minipočítač, který podporuje všechny funkce stejně jako Raspberry Pi. Samotné rozložení Banana Pi BPi-M2 je téměř totožné s Raspberry 2.

Tento konkrétní model disponuje čtyřjádrovým procesorem 4,1 GHz a 1 GB operační paměti.

Pro připojení k síti lze využít ethernetový konektor nebo modul Wi-Fi 802.11 b/g/n. Další konektory na desce jsou: 4x USB, HDMI a audio výstup 3,5 mm Jack.



Obrázek 8: Zařízení Banana BPi-M2 [25]

Ukládání dat a načítání operačního systému probíhá přes microSD kartu. Podporované operační systémy jsou Lubuntu, Debian Linux, Ubuntu Linux, Raspbian a Android.

Zařízení je napájeno přes microUSB konektor. [25]

7.1.2 Raspberry Pi 3 model B+

Modely Raspberry Pi jsou nejvíce využívané pro VoIP technologie.

Tento model obsahuje čtyřjádrový procesor 1.4 GHz a 1 GB RAM. pro komunikaci obsahuje moduly WiFi 802.11AC, Bluetooth 4.2 BLE, 300 Mb/s Ethernet a PoE konektor. Obsahuje video a zvukové vstupy a výstupy, 4 USB konektory.

Podporuje microSDHC, tedy vnitřní paměť se může dostat až na 32 GB.

7.1.3 Orange Pi 4

Orange Pi opět poskytuje obdobné funkce jako předchozí zařízení. Jeho velikost a hmotnost je taktéž obdobná.

Tento model disponuje šestijádrovým procesorem 2 GHz a RAM paměť 4 GB.

Pro bezdrátovou komunikaci jsou k dispozici moduly pro wi-fi a bluetooth. Dále disponuje zařízení klasickým ethernetovým portem.

Zařízení podporuje následující operační systémy: Debian, Ubuntu a Android.

Na rozdíl od předchozích zařízení obsahuje většina Orange Pi zařízení Wi-fi anténu, vstupy a výstupy pro audio a video. Tyto modely jsou spíše využívány v odvětví IoT. [33]

8 Rozbor zadání

V této části bude blíže popsán nástroj pro monitorování. Budou specifikována jednotlivá zařízení potřebná pro fungování systému. Obecně bude popsáno chování systému, postup měření a vyhodnocování dat. Také budou specifikována omezení na konkrétní implementaci řešení monitorovacího zařízení.

8.1 Základní předpoklady

V této části jsou zformulovány základní předpoklady, podle kterých bylo vypracováno konečné řešení. Pokud by některý z těchto předpokladů nebyl splněn, muselo by se upravit celé řešení.

V rámci pobočkové sítě počítám pouze s telefonním provozem na bázi technologii VoIP.

Technologie TDM nebyla zpracována z ohledem na čas dokončení práce a místo ní byl vytvořeno webové rozhraní pro zobrazování naměřených dat a konfiguraci. Webový server byl vybrán jako přednější kvůli zjednodušení obsluhy systému koncovému uživateli.

Dále počítám se symetrickým průchodem paketů sítí. Doba cesty paketu sonda-centrála bude stejná jako centrála-sonda.

Adresa všech prvků v síti bude statická, nebude využito DHCP.

Jelikož předpokládáme, že síť je chráněna protokolem IPSec a je vytvořena VLAN/VPN a přenášená data jsou pouze výsledky měření, není potřeba využívat zabezpečené varianty VoIP protokolů.

Bude zajištěna transparentnost sítě pro protokol ICMP.

Je předpokládaná stoprocentní spolehlivost centrálního prvku a spojení na dispečera/správce technologie.

8.1.1 Využitá infrastruktura

Veškerá infrastruktura bude ve vlastnictví a správě zákazníka. Toto může způsobovat určité limity. Bude nutné se se správcem sítě dohodnout na konkrétním zapojení a parametrech komunikace.

Z bezpečnostního rizika by mohl nastat případ, že bychom byli komunikačně v síti omezeni jen na určité protokoly. V tomto případě je potřeba vyřešit předávání informací mezi sondami a centrálním prvkem. Pro samotné fungování aplikace je nutnost povolení protokolů SIP a RTP/RTCP. Protokol SIP sice podporuje přidání dodatečných informací pomocí speciální hlavičky 'P'. Pro přenos naměřených dat lze použít protokol RTCP, který obsahuje naměřená data obou stran.

Jedním z řešení je využití LTE modemu, který by byl součástí zařízení. Tímto způsobem bychom získali kontrolu nad projektem. Otevírá to ovšem možnost mobilních útoků na síť klienta. Samozřejmě by měla být síť klientů kritické infrastruktury proti těmto útokům ošetřena, přesto by toto řešení nemuselo být v souladu s bezpečnostní politikou správce sítě klienta.

Další variantou by bylo vytvoření samostatné, izolované VLAN v rámci sítě klienta přímo pro tento projekt. S touto variantou se počítá v dalším řešení projektu.

Pro sledování parametrů existující pobočkové sítě je potřeba řídit se její směrovací politikou. Je tedy nutné, aby jednotlivé prvky byli zapojeny přímo do ústředen v daných lokalitách. Pokud by tomu tak nebylo byla by sledována pouze vhodnost dané IP sítě pro VoIP technologii.

8.2 Prvky v síti

Obecně se v celém monitorovacím systému budou vyskytovat dva typy prvků: sondy a centrální prvek. Centrální prvek by se měl vyskytovat v hlavní pobočce firmy. Sondy by se poté měly zapojit do sítě v monitorovaných pobočkách.

Centrální prvek má za úkol shromažďovat a vyhodnocovat naměřená data. Budou v něm uloženy informace o všech sondách. Centrální prvek je schopen vyvolávat a přijímat hovory na/z sondy.

Hovory se budou vyvolávat pomocí systémové pobočkové ústředny Asterisk, která bude na centrálním prvku nainstalována. Systém Asterisk je možné nainstalovat na fyzický nebo virtuální server s podporovaným operačním systémem. V tomto případě Linux, Debian 11.

Sondy jsou zařízení, která dokážou hovory přijímat i vyvolávat. Je vyžadována funkce automatického volání, proto bude pro sondy využit obdobný systém jako pro centrální prvek. Sonda bude zařízení, na kterém poběží Debian 11 se systémem Asterisk.

8.2.1 Nároky a cena prvků

Centrální prvek je možno plně virtualizovat a využít některý ze serverů, které by již zákazník vlastnil. Nároky na prostředky virtuálního PC se budou odvíjet od nároků na fungování OS Debian 11 a od nároků Asterisku.

Nároky Debian 11:

- RAM: minimum 512MB, doporučeno 2GB
- paměť: 10GB
- procesor: 1GHz

Nároky na provoz Asterisku závisí na počtu koncových bodů, které chceme udržovat zároveň. Tedy bude záležet na počtu sond, které budou k centrálnímu prvku připojeny. Dále záleží na počtu simultánních hovorů a jejich průměrné délce. Doba hovoru je pevně nastavena pro odchozí i příchozí hovory, a to 100 sekund a 30 sekund.

Oficiální soupis požadavků je uveden v [9], tento údaj je ovšem z roku 2007, proto je potřeba ho brát pouze orientačně. Zde jsou nastaveny nároky pro malou firemní síť (do 25 zařízení) 3 GHz procesor a 1GB RAM. Pro větší systémy je poté doporučeno využít duálního jádra procesoru.

Při výpočtu zátěže vytvořeného v roce 2015 bylo počítáno se systémem 2.67GHz Intel Xeon CPU s 2GB RAM. Z matematických modelů vyplývá, že ztrátovost je nulová i pro 122 kanálů s délkou hovoru 120 sekund [6]. Z toho můžeme uvážit, že použití výše vypsaných parametrů by mělo být dostatečné.

Pro centrální prvek by bylo možné specifikovat nároky pro obsluhu 25 sond na:

- RAM: 2-4GB
- procesor: 3GHz
- paměť: 20GB

Ideální by bylo využít server přímo u zákazníka a vypůjčit si jeho výpočetní prostředky. Pokud bych měla odvodit cenu, můžeme vzít například zařízení Raspberry Pi 4 model B - 8 GB, procesor 1,5GHz čtyř jádrový. Toto zařízení by mělo 25 sond zvládnout. Cena tohoto zařízení je 2249,- Kč. K této ceně se musí započítat ještě cena SD karty, která je nutná pro chod zařízení.

Sonda má nároky menší než centrální prvek. Pokud by byla sonda virtualizována na serveru, byly by hlavní nároky na minimu pro rozběhnutí OS Debian 11 a systému Asterisk. Na sondě bude v danou chvíli probíhat vždy pouze jeden hovor. Další možností je využití mini počítače, například zařízení Raspberry Pi nebo Banana Pi. Podle [10] je patrné, že pro nenáročnou telefonní aplikaci stačí RAM 512MB. V dnešní době je cena zařízení Raspberry Pi 1 Model B+ 512MB RAM rovna 769,- Kč (https://rpishop.cz/raspberry-pi-1b/74-raspberry-pi.html).

8.3 Obecné chování nástroje

Nástroj primárně monitoruje spojení a jeho kvalitu mezi centrálním prvkem a sondami. Spojení je periodicky kontrolováno na pozadí bez nastavení od uživatele.

Uživatel definuje sondy a dále nastavuje pravidelnost kontrolních hovorů. Během těchto hovorů se získají parametry pro následovný výpočet MOS pro určení kvality. Centrální prvek MOS vyhodnotí a v případě špatné kvality obeznámí správce sítě.

Po ukončení hovoru automaticky centrální prvek čeká na hovor zpět od sondy. Opět je vypočten MOS a hovor je zaznamenán.

Všechny hovory jsou zaznamenávány včetně naměřených hodnot pro ztrátovost paketů, zpoždění a jiter.

Centrální prvek reaguje a vyhodnocuje několik mimořádných stavů, popsaných níže.

8.3.1 Druhy a popis mimořádných stavů

V rámci monitorovací činnosti bude systém detekovat a následně reagovat na sadu mimořádných stavů. Tyto stavy budou detekovány na centrálním prvku, který bude zodpovědný za jejich nahlášení.

Seznam těchto stavů je následující:

- výpadek spojení
- nemožnost navázání telefonního hovoru
- obousměrná či jednosměrná slyšitelnost
- nízká kvalita spojení

Pod výpadkem spojení je myšlena síťová dostupnost sondy pro transparentní sítě, kde bude umožněn průchod pro PING. Dále je automaticky systémem Asterisk kontrolován stav registrace sondy. Po připojení k centrálnímu prvku by sonda měla udržovat spojení. Pokud se sonda odregistruje z jiného důvodu než výpadku spojení, je to zjištěno až při vyvolání hovoru. Síťová dostupnost bude indikována i na webové stránce centrálního prvku.

V rámci sytému budou pravidelně vyvolávány hovory směrem z centrálního prvku na sondy. Pokud tento hovor nebude možné sestavit, bude opět kontaktován dispečer. Po úspěšném hovoru směrem na sondu, je automaticky očekáván hovor ze sondy, aby byla podchycena situace, kdy by bylo možné sestavovat hovory pouze jedním směrem. V tomto případě může nastat situace, že při nefunkčnosti centrálního prvku, nebudou hovory ze sondy vyvolávány vůbec.

Nicméně v tomto případě bude již nahlášena porucha, kvůli nefunkčnímu hovoru z centrálního prvku a není potřeba nic více speciálně řešit.

Tyto zpětné hovory by bylo možné vyvolávat méně často v určité časové okamžiky a centrála by je v tuto dobu očekávala. Tento koncept by ovšem vyžadoval časovou synchronizaci centrálního prvku a sond, nejspíš pomocí protokolu NTP. Přidala by se možnost chybně nastaveného času, který by mohl ovlivnit monitorovací činnost. Navíc by musel existovat kontrolní mechanismus, který by zajišťoval, aby se hovory nevyvolali náhodou zároveň. Zároveň by přibyla další nutná konfigurace pro uživatele, pokud by i zpětné hovory byly v režii koncového zákazníka.

Pod pojmem slyšitelnosti je rozuměno správné navázání médií, tedy odesílání a přijímání RTP paketů z/na správnou adresu.

Kvalita spojení bude odvozena z parametru MOS, který bude vypočten z naměřených hodnot během hovoru. Tyto hodnoty jsou přenášeny v rámci paketů protokolu RTCP. Podle standardů bude určena hraniční hodnota a hovor s jakoukoliv vypočtenou hodnotou nižší bude hlášen jako málo kvalitní. Toto snížení kvality může být způsobeno například přetížením sítě. Naopak bude potřeba sledovat i pokud by hodnota MOS byla moc vysoká. Podle maximální možné hodnoty pro MOS budou tedy vyhodnocovány situace, kdy také hovor nebude probíhat standardně.

8.3.2 Reakce na mimořádné stavy

V případě detekce jednoho z výše popsaných mimořádných stavů bude systém reagovat jedním, nebo více z následujících možností:

- kontaktování dispečera
  - hovorem
  - textovou zprávou
- indikace mimořádné situace na webové stránce centrálního prvku
- výpis do seznamu mimořádných událostí

Ve všech případech mimořádných stavů budou tyto situace zaznamenány do výpisu mimořádných událostí, který bude možno stáhnout z webové stránky.

Na závažné poruchy bude dispečer upozorňován přímo pomocí hovoru, nebo textové zprávy.

Na webových stránkách budou vidět záznamy o všech uskutečněných hovorech, včetně výpisu hodnoty MOS. Dále bude možno v seznamu sond vidět indikaci síťové konektivity.

Tyto reakce mohou být dále přizpůsobeny konkrétním požadavkům zákazníků, jednalo by se ovšem již o zásah do konfigurace.

8.4 Využité protokoly

Pro ověřování síťové dostupnosti je využit nástroj PING, který používá protokol ICMP.

Pro transport médií budou využity protokoly RTP a RTCP. Protokol RTCP bude využit i pro získávání kvalitativních parametrů hovorů.

Jelikož je v dnešní době nejvíce rozšířen pro VoIP komunikaci protokol SIP, bude v rámci konfigurace Asterisku využit kanál pro SIP, konkrétně pak projekt PJSIP. Případnou alternativou by byl protokol IAX2.

NTP protokol by bylo potřeba využít v případě, kdy by hovory ze sondy nebyly vyvolávány ihned po hovoru z centrálního prvku, ale v domluvený čas. V tomto případě by bylo potřeba synchronizovat čas právě pomocí protokolu NTP. Centrální prvek by sloužil jako NTP server a jednotlivé sondy by si podle něj čas synchronizovaly.

8.5 Výstup

Primárním výstupem budou dva textové soubory, do těchto souborů jsou zapisovány výsledky měření. První soubor bude obsahovat soupis všech sond s jejich příslušnými IP adresami. V tomto souboru bude také u každé sondy indikace zda je síťově dostupná. Druhý soubor bude obsahovat historii měření. Tento soubor bude ve formátu:

ČAS DÉLKA_HOVORU NÁZEV_SONDY TYP_HOVORU MOS ZTRÁTOVOST_PAKETŮ RTT JITER

Oba tyto soubory budou využity pro zobrazení dat na webové stránce. Webová stránka slouží k prohlížení naměřených hodnot i k prohlédnutí seznamu sond.

Dalším souborem bude soupis mimořádných událostí. Veškeré mimořádné události jsou zaznamenávány do souboru, tento soubor bude ke stažení z webové stránky.

9 Realizace

Postup realizace bude popsán v následujících kapitolách. U postupu bude odlišena konfigurace centrálního prvku od konfigurace sond. Samotná konfigurace sond vychází z centrálního prvku, ale má některá specifika, která budou popsána.

Nejprve bude popsána konfigurace Asterisku pro tvorbu a přijímání hovorů. V této části bude popsán způsob automatického generování hovorů.

Další část se věnuje samotnému vyhodnocovacímu skriptu a tvorbě výstupních souborů. Dále budou popsány další pomocné skripty pro zajištění správné funkce aplikace.

Samostatná podkapitola se věnuje instalaci a nastavení zařízení Raspberry Pi. Následuje podkapitola o konfiguraci webového serveru pro zobrazení naměřených dat a konfiguraci nástroje.

Poslední část se věnuje samotnému otestování systému v laboratorním prostření.

Všechny konfigurační soubory budou přiloženy jako přílohy.

9.1 Konfigurace Asterisku

Tento nástroj využívá software Asterisk verze 18.8. [36]

Asterisk je spouštěný jako 'service' a to z důvodu, aby byla aplikace sama znovu spuštěna například při pádu, nebo při spuštění systému. K zapnutí, vypnutí a získání status o servise využívám syntaxi příkazů:

- `service asterisk start`
- `service asterisk stop`
- `service asterisk status`

Konfigurace Asterisku se dělí na dvě části:

- konfigurace koncových bodů
- konfigurace 'dialplanu'

9.1.1 Konfigurace koncových bodů

V konfiguračním souboru `pjsip.conf` jsou definována jednotlivá koncová zařízení. Dále musí být v tomto souboru definovaná registrace k centrálnímu prvku. Jelikož chceme, aby se dalo dovolat i na centrální prvek, je nutné vytvořit registrace i na straně centrály. Asterisk v tuto chvíli neslouží jen jako ústředna, ale i jako koncové zařízení.

Příklad konfigurace pro jednu sondu na centrálním prvku je následovný:

```
[sonda](endpoint-sonda)
outbound_auth=sonda
aors=sonda

[sonda](auth-userpass)
password=sonda
```

```
username=sonda

[sonda](aors-sonda)

[sonda](identify-sonda)
endpoint=sonda
match=10.48.67.47

[sonda]
type=registration
server_uri=sip:10.48.67.47
client_uri=sip:sonda@10.48.67.47
contact_user=sonda
outbound_auth=sonda
```

Na sondě je konfigurace obdobná.

V tuto chvíli je nutné uvést, že zde došlo ke značnému zjednodušení situace oproti reálné síti. Ve skutečnosti by registrace probíhaly na ústředny pobočkové sítě dané lokality, jak bylo specifikováno v předpokladech o využité infrastruktuře.

Pro větší přehlednost jsou na centrálním prvku pro každou sondu vytvořeny zvláštní konfigurační soubory, které jsou zahrnuty v hlavním konfiguračním souboru pomocí příkazu 'include'. Pro 3 sondy by tedy souborová struktura vypadala následovně:

- pjsip.conf
- sonda_jmeno1.conf
- sonda_jmeno2.conf
- sonda_jmeno3.conf

V souboru `pjsip.conf` je vytvořena registrace pro správce sítě. Pokud by byl umožněn přístup do sítě, dalo by se připojením k poskytovateli hovorové služby směrovat telefonní hovory přímo na číslo správce, případně propojením trunkem k pobočkové ústředně.

9.1.2 Konfigurace 'dialplanu'

Konfigurace 'dialplanu' je odlišná pro centrální prvek a sondy. Nejprve popíšu chování 'dialplanu' centrálního prvku.

Centrální prvek

V rámci centrálního prvku jsou definovány 3 kontexty. V rámci jednoho jsou obsluhovány sondy, hovory a textové zprávy pro správce sítě, poslední kontext je pomocný pro vyvolání kontrolního skriptu.

U vyvolávání automatického hovoru jsem narazila na problém, že pokud není koncový bod registrován a hovor není vůbec proveden a není tedy zaznamenán do CDR. Tedy není možné zkontrolovat zpětně zda hovor proběhl, jednou z možností by bylo kontrolovat v průběhu testu, zda byl kanál na sondu otevřený. Tedy v rámci skriptu, který vyvolává hovor, by byla část, která čeká určitou dobu a poté si nechá vypsat z CLI Asterisku všechny otevřené kanály. Následně zkontroluje jestli je součástí tohoto výpisu název sondy.

Druhá metoda, kterou jsem využila, byla vytvořit hovor na lokální kanál pomocí 'call' souboru a poté v rámci 'dialplanu' zkusit zavolat na sondu. V tomto případě i pokud je koncový bod neregistrovaný bude zaznamenán v CDR.

V tuhle chvíli je problém, že po vyvolání hovoru přes lokální kanál, je hovor okamžitě ukončen a není možné dokončit zbytek kroků a tedy není možné shromáždit parametry pro výpočet MOS.

Lokální kanál je využitý pro kontrolu, zda se dá na sondu dovolat. Pokud je sonda dostupná, je na ní vyvolán testovací hovor.

V rámci kontextu pro sondy jsou definovány čísla pro jednotlivé sondy pro vyzvednutí hovoru, jedno číslo pro vytvoření hovorů na sondy a jedno číslo pro lokální kanál, který otestovává dostupnost sond.

Celkové rozložení konfiguračního souboru je následující:

- internal - kontext pro sondy
  - 123 - číslo lokálního kanálu pro kontrolu dostupnosti sondy
  - 100 - číslo pro vytvoření hovoru na sondu a změření kvality
  - jednotlivá čísla na sondy podle názvů sond
  - h - číslo pro uskutečnění akcí po ukončení hovoru
- dispecer - kontext pro hovory a textové zprávy pro správce sítě
  - 8902 - číslo lokálního kanálu
  - 100 - automatický hovor na dispečera
  - 300 - automatická textová zpráva na dispečera
- testhovor - pomocný kontext
  - 200 - číslo pro vyvolání skriptu pokud byl test dostupnosti úspěšný

Číslo 123 je číslo lokálního kanálu. V rámci jeho činnosti je otestována dostupnost sondy pomocí aplikace 'Dial()'. Tato aplikace je ukončena v jednom z 4 stavů, které je možné získat jako proměnnou pomocí funkce 'CDR()'. Tato proměnná s názvem sondy je předána do skriptu, který vyhodnotí, zda je možné uskutečnit hovor pro měření.

Jednotlivá čísla sond slouží k vyzvednutí hovorů ze sond.

`Extension` 100 vykonává úkony na kanálu otevřený na jednotlivé sondy. V rámci jeho činnosti je spuštěna testovací nahrávka s přesnou délkou. Po ukončení nahrávky jsou nasbírány hodnoty RTT, ztrátovost paketů a jiter. Tyto hodnoty jsou získány pomocí funkce `CHANNEL(rtcp, parametr)`. Tato funkce vyčítá jednotlivé parametry ze zpráv protokolu RTCP. Hodnota RTT je přiřazena proměnné v rámci 'dialplanu' následovně:

```
same => n,Set(RTCP_rtt=${CHANNEL(rtcp,rtt)})
```

Při ukončení hovoru v rámci kontextů 'internal' nastávají dvě akce. Nejprve je zapsána délka hovoru do proměnné. Druhým úkonem je vyvolat vyhodnocovací skript 'MOS.py' pomocí aplikace 'AGI' se všemi změřenými hodnotami jako vstupními parametry.

V rámci hovoru na dispečera je pouze přehrána hláška o výskytu chyby v monitorované síti. Více informací o vyskytnuté chybě je předáno pomocí textové zprávy, tělo textové zprávy se mění

```
[dispecer]

exten => 8902,1,Answer()

exten => 100,1,Noop(Chyba - hovor na dispecera)
        same => n,Playback(chyba)
        same => n,Hangup()

exten => 300,1,Noop(Sending sms to dispecer)
        same => n,Set(MESSAGE(body)="[${STRFTIME(${EPOCH},,%d%m%Y-%H:%M:%S)}] ${text}")
        same => n,Set(ACTUALFROM=pjsip:dispecer)
        same => n,MessageSend(${ACTUALFROM},CentralniPrvek)
```

Obrázek 9: Konfigurace pro kontext dispečera v rámci diaplanu

v návaznosti na detekované mimořádné události. Vždy je uveden čas a název sondy, který je v součástí textu. Tento text je předán z vyhodnocovacího skriptu. (viz. obrázek 9)

Pomocný kontext jsem definovala spíše pro zvýšení přehlednosti celého souboru, jeho obsah by mohl být napsán i do kontextu 'internal'.

Konfigurace sond

'Dialplan' sond je zjednodušený. Sonda nemusí vyvolávat hovory na správce sítě a dále nedochází ke sběru parametrů pro výpočet MOS. V souboru `extensions.conf` je definován pouze jeden kontext a v rámci něj jsou definována dvě čísla:

- 100
- nazev_sondy

Pomocí čísla 100 se řídí hovory automaticky vyvolané ze sondy. Číslo dané názvem sondy slouží pro vyzvedávání hovorů, které přicházejí z centrálního prvku. V rámci obou čísel je vyvolán testovací zvukový záznam.

9.1.3 Automatické generování hovoru

Automatické generování hovorů je možné několika způsoby. Jeden ze způsobů je vyvolávat hovory přes správcovské rozhraní AMI. V tomto případě se lze přes telnet připojit k Asterisku a pomocí série příkazů vyvolat hovor. Dalším způsobem je využití 'call' souborů. Zde je v rámci souboru definováno chování a po umístění do složky `/var/spool/asterisk/outgoing` jsou příkazy v souboru realizovány. Další možností je využití skriptu, který bude obsahovat příkaz Asterisk příkazové řádky pro vytvoření hovoru. Tento příkaz vypadá následovně: `originate PJSIP/sonda extension exten@context`.

V rámci tohoto projektu jsem zvolila metodu generování 'call' souborů. Tímto způsobem jsou generovány testovací hovory i hovory na správce sítě (případně textové zprávy).

Obecný syntax tohoto souboru vypadá následovně:

```
Channel:název_technologi/kanál
Context:název_kontextu_v_rámci_dialplanu
Extension:číslo_v_rámci_kontextu
CallerID:jméno_které_je_zobrazeno_protější_straně
```

Konkrétní soubor `dispecer.call` pak vypadá následovně:

```
Channel:PJSIP/dispecer
Context:dispecer
Extension:100
CallerID:centrala
```

Některé soubory jsou statické a pevně vytvořené v adresáři `/root/AutoDial/` a jiné jsou dynamicky vytvářeny pomocí skriptů, například při vyhodnocení mimořádné události. Celkem je v rámci nástroje využito 4 druhů souborů generující automatické volání.

- testovací hovor na sondu pomocí lokálního kanálu
- testovací hovor na sondu měřící kvalitu
- hovor na dispečera
- poslání textové zprávy na dispečera

9.2 Systém vyhodnocování

V rámci celého systému dochází k vyhodnocování mimořádných událostí ve třech částech. Vyhodnocování probíhá v rámci skriptů. Pro tvoru skriptů jsem zvolila jazyk Python.

První je vyhodnocování síťové konektivity. To probíhá na pozadí aplikace. Každou minutu je postupně pomocí protokolu ICMP a nástroje PING otestována síťová konektivita. V případě, že je zjištěn výpadek, je zaznamenán čas výpadku a zavolán správce sítě. Po ukončení hovoru je správci sítě poslána textová zpráva s informací o výpadku.

Druhé vyhodnocování se týká možnosti uskutečnit hovor na sondu. Pokud není možné hovor spojit, je zaznamenán čas pokusu uskutečnit hovor, jméno sondy a důvod neproběhnutí hovoru.

Třetí vyhodnocení přichází po ukončení hovoru při výpočtu kvality. Pokud se kvalita hovoru nenachází v předem definovaných mezích je upozorněn dispečer textovou zprávou. V tomto vyhodnocení dochází i k vyhodnocování jednostranné či oboustranné neslyšitelnosti.

9.2.1 Síťová dostupnost

K vyhodnocení síťové dostupnosti dochází ve skriptu `sondaPing.py`. V rámci vyhodnocování bylo potřeba brát i minulý stav síťové dostupnosti. Sondy se tedy nachází v jednom z následujících stavů:

1. Sonda je dostupná a byla dostupná
2. Sonda je dostupná a nebyla dostupná = obnova spojení
3. Sonda je nedostupná a byla dostupná = ztráta spojení
4. Sonda je nedostupná a byla nedostupná

Na každý z těchto stavů je jiná reakce. Na první stav se nijak nereaguje. Při znovu obnovení spojení je na správce poslaná textová zpráva s upozorněním, která sonda byla opět připojena. Při ztrátě spojení je vyvolán hovor a poslána textová zpráva. Pokud chybový stav přetrvává není už nijak oznamováno. Při masivním výpadku by mohlo dojít k přehlcení správce technologie, pokud by periodicky dostával zprávy o výpadku sítě z vícero sond.

Stavy 2., 3. a 4. jsou logovány do souboru `aplikace.log`.

9.2.2 Uskutečnění hovoru

Zda je možné uskutečnit testovací hovor je vyhodnocováno ve skriptu `testcall.py`. Tento skript je vyvolán v rámci 'dialplanu' po vytvoření hovoru na sondu přes lokální kanál. Výsledný stav a název sondy jsou vstupní parametry skriptu.

Výsledný stav je získán z funkce `CRD(disposition)`. Návratová hodnota této funkce nabývá následující hodnoty:

- NO ANSWER (bez odpovědi)
- FAILED (chyba)
- BUSY (osazeno)
- ANSWERED (odpovězen)

Testovací hovor je možné vyvolat pouze pokud byla sonda dostupná a na hovor odpověděla. Všechny ostatní hodnoty jsou vyhodnocené jako chybové a jsou hlášeny pomocí hovoru, textové zprávy a logů.

V těle textové zprávy je předáván název sondy a návratová hodnota funkce `CDR` pro větší přehlednost situace.

Soubory pro automatické vyvolání hovorů na správce sítě jsou dynamicky vytvářeny v rámci skriptu.

9.2.3 Parametry hovoru

Všechny kvalitativní parametry jsou vyhodnocovány pomocí skriptu `MOS.py`. Veškeré vyhodnocování vychází ze vstupních parametrů předaných z 'dialplanu' Asterisku. Tyto vstupní parametry jsou následující:

- čas hovoru
- délka hovoru
- název sondy
- ztrátovost paketů
- RTT
- jiter
- typ hovoru (příchozí/odchozí)
- cílová adresa RTP toku
- zdrojová adresa RTP toku

Z parametrů sítě je vypočtena hodnota R-faktoru. R-faktor je podle převodního vztahu přepočten na MOS. Tyto parametry musí být převedeny na formát čísla z textového řetězce. Dále jsou výsledné hodnoty zaokrouhlovány na 3-4 desetinná místa.

V první řadě je vyhodnocena kvalita. V tomto kroku je ověřeno, že R-faktor nepřesáhl hodnotu 100 a nedošlo k chybě ve výpočtu. Následně je ověřena hodnota MOS. Pokud je hodnota menší než 4, je obeznámen správce technologie textovou zprávou.

Navázání medií je kontrolováno podle IP adres cíle a zdroje RTP toku dat. V testovacím případě je multimediální tok nastaven přímo mezi centrálním prvkem a sondou. V reálné síti, kde je nastaven NAT a firewally, by komunikace probíhala přes ústřednu, ke které by byla centrála připojena.

9.2.4 Reakce na mimořádnou událost

Jak bylo výše popsáno, na mimořádné události reaguje systém třemi způsoby. V následující části jsou popsány části skriptů vyvolávající dané reakce. Tyto skripty se vyskytují v rámci všech vyhodnocení.

Vyvolání hovoru na dispečera

Pro vyvolání hovoru je nutné vytvořit `.call` soubor a tento soubor přesunout do složky `/var/spool/asterisk/outgoing`.

Tvorba `.call` souboru vypadá následovně:

```
fileDispecer = open(pathDispecer, 'w+')
fileDispecer.write("""
Channel:PJSIP/dispecer
Context:dispecer
Extension:100
CallerID:"CentralniPrvek"
""")
fileDispecer.close()
```

Jedná se o jednoduchou manipulaci s textovým souborem. Soubory lze přesouvat pomocí knihovny `shutil`, konkrétní syntaxe je: `shutil.move(zdroj, cíl)`.

Poslání textové zprávy na dispečera

Vyvolání textové zprávy je obdobné jako pro hovor. Jediným rozdílem je využití lokálního kanálu pro odeslání textové zprávy. Pokud by byl v 'call' souboru nastaven přímo dispečer, byl by na tento kanál nejprve vyvolán hovor. Navíc je pouze nastavená proměnná 'text', která obsahuje chybovou hlášku, která je v 'dialplanu' stane tělem zprávy.

Předání proměnné do 'dialplanu' lze udělat pomocí příkazu `Set` v rámci `.call` souboru.

```
Set:promenna=hodnota/text
```

Zápis mimořádné události do logů

V případě logů se opět jedná o jednoduchou manipulaci se souborem. Ve složce `/var/log` byla vytvořena složka `aplikace` se souborem `aplikace.log`, do tohoto souboru budu ukládány všechny definované mimořádné události.

Samotné vložení logu je uskutečněno následovně:

```
fileLog=open('/var/log/aplikace/aplikace.log','a+')
fileLog.write(newline)
fileLog.close()
```

Při otvírání souboru je využitý parametr 'a+'. Zápis bude pokračovat na posledním řádku a pokud soubor neexistuje bude vytvořen.

9.3 Další využité skripty

V rámci nástroje bylo využito více skriptů. Kromě vyhodnocovacích byl hlavním skriptem skript pro přesouvání souboru na tvorbu hovoru do správného adresáře. Tento skript je pravidelně vyvoláván pomocí nástroje `crontab`.

Dále jsem vytvořila pomocné skripty na restart aplikace Asterisk a přidání nové sondy. Tyto skripty byly následně využity při tvorbě webových stránek.

Pro kontrolu síťové konektivity byl vytvořen skript využívající nástroj PING. V rámci tohoto skriptu jsou postupně otestovány všechny sondy ze seznamu sond a je zapsán jejich stav, zároveň dochází k samotnému vyhodnocení síťové konektivity.

Seznam všech využitých skriptů na centrálním prvku (vyjma skriptů pro tvorbu webu):

- `sondaPing.py`
- `MOS.py`
- `testcall.py`
- `autoMove.py`

Seznam všech využitých skriptů na sondě:

- `autoMoveSonda.py`

9.3.1 Popis skriptu automove.py a testcall.py

Tyto skripty mají za úkol dle vstupního parametru vytvořit soubor pro vytvoření automatického hovoru. Tento soubor je poté v rámci skriptu přesunut do složky odchozích hovorů pro Asterisk.

V rámci skriptu `automove.py` je vytvořen hovor přes lokální kanál. Vstupním parametrem tohoto skriptu je název sondy, na kterou je snaha se dovolat. Tento název a výsledek volání jsou předány jako vstupní parametry skriptu `testcall.py`. V rámci tohoto skriptu je provedena kontrola, zda je možné vyvolat hovor na sondu. Pokud je sonda dostupná je vytvořen soubor pro automatický hovor na danou sondu. V opačném případě je obeznámen dispečer a tento stav je zaznamenán do logů.

Skript `automoveSonda.py` na sondě vyhodnocuje zda na něj přišel jen hovor testující dostupnost a nebo testovací hovor. Toto je vyhodnoceno podle délky hovoru. Při testování dostupnosti je délka hovoru 0 sekund. Pokud se nejedná o test dostupnosti, je vyvolán hovor zpět na centrální prvek.

9.3.2 Popis skriptu sondaPing.py

Základem skriptu byla definice nástroje PING, který byl využitý pro kontrolu síťové konektivity.

```
def myping(host):
    parametr = '-c'
    command = ['ping', parametr, '1', host]
    response = subprocess.call(command)

    if response == 0:
```

```
        return True
    else:
        return False
```

Největší problém byl získat skutečný stav. V rámci nástroje PING je totiž odpověď `Destination host unreachable` (host nedostupný) brána jako pozitivní. Je-li jedna z odpovědí na PING tato, je celkově PING brán jako úspěšný, proto jsem definovala jiný způsob vyhodnocení. Tento způsob spočívá v poslání 4 zpráv ICMP za sebou. Všechny tyto zprávy musí mít kladnou odpověď. Pokud nebyly doručeny všechny 4 odpovědi, je prvek vyhodnocen jako síťově nedostupný.

```
result = [True]*4
for i in range(4):
    result[i]=myping(hostname)
endresult=all(result)
```

Tímto způsobem bude vyhodnocena jako nedostupnost i situace pokud je například půlka paketů poslána. Tato monitorovací aplikace je pro technologii VoIP, která na ztrátovost paketů reaguje negativně. Tedy je žádoucí upozornit i na špatnou síťovou konektivitu.

IP adresa cíle je vyčtena ze seznamu sond v souboru `seznamSonda.txt`. Tímto způsobem je postupně otestována jedna sonda podruhé.

Tímto způsobem je kontrolována síťová dostupnost sondy pouze v rámci IP sítě. Samotná pobočková síť může využívat jiné směrování a tedy je možné, že sonda nebude pro hovor dostupná. Z tohoto důvodu je nutné pravidelně vyvolávat hovorové testy, kde je komunikace navázána přes pobočkovou síť. Tento test sleduje stav IP sítě.

9.4 Instalace zařízení *Pi

Pro sondu bylo využito zařízení Raspberry Pi 3 model B+. Zařízení Raspberry Pi jsou využívána pro telefonní aplikace, proto byl vybrán tento model s ohledem na nároky aplikace.

Prvním krokem při konfiguraci zařízení *Pi bylo zvolit operační systém a vytvořit jeho obraz na microSD kartu. Jako operační systém jsem zvolila Raspbian. Obraz operačního systému lze vytvořit pomocí aplikace `Raspberry Pi Imager`, který je dostupný na oficiálních stránkách Raspberry Pi.

Při prvním zapojení zařízení proběhne instalace a vytvoření uživatele. Dále bylo potřeba aktualizovat programy a aplikace operačního systému. Poté bylo potřeba doinstalovat další programy, které nejsou součástí základní instalace (textový editor Vim, tcpdump).

V tuto chvíli je na zařízení funkční verze debianu. Zdrojové soubory pro Asterisk je možné stáhnout pomocí příkazu `wget` s odkazem na stažitelný soubor dané verze Asterisku. Zbytek instalace je obdobný jako pro centrální prvek.

Dalším krokem je vytvoření konfigurace Asterisku (`pjsip.conf` a `extention.conf`) a překopírování dalších souborů (`autoMove.py` a `autocall.call`).

Posledním krokem je instalace zařízení do testovací sítě a korektní nastavení IP adresy, která je statická.

9.5 Konfigurace web serveru

V této podkapitole bude popsána základní konfigurace a fungovaní webové rozhraní pro monitorovací nástroj. Celý kód je součástí přílohy, v rámci textu budou zmíněny jen některé úryvky.

Při tvorbě webového rozhraní byl využit následující projekt: 'Django tutorial'[38] a následující nástroje: Django [35], Bootstrap [37]. Nástroj Django jsem vybrala z důvodu možnosti tvorby dynamických stránek v jazyce Python, který byl již využit v rámci vyhodnocovacích skriptů.

Webový server byl vytvořen pomocí nástroje Django. Nejprve byl vytvořen nový projekt *mywebserver* a v rámci tohoto projektu aplikace *mereni*.

V aplikaci byli vytvořeny 3 modely pro správu a zobrazovaní hodnot ze sond. První model reprezentuje samotné sondy a obsahuje proměnné pro název, heslo, IP adresu a dva číselné údaje pro nastavování frekvence měření v minutách a hodinách. Další dva modely slouží k zobrazení výsledků uložených v textových souborech. Tyto modely jsou definovány v souboru `models.py`.

Některá pole v rámci modelu mají vyplněné další parametry. Například název sondy a IP adresa musí být vždy unikátní.

```
class Sonda(models.Model):
    Nazev = models.CharField(max_length=30, unique=True,
       help_text='Zadejte název sondy')
    ...
    mereni_minuty = models.IntegerField('', default=0)

    def __str__(self):
        return F'{self.Nazev}, {self.ip_adresa}'
```

Pro spuštění webového serveru v rámci projektu Django se používá příkaz:
`python3 manage.py runserver 0.0.0.0:8000`.

Soubor `manage.py` se nachází v hlavním adresáři projektu. Při změnách v rámci modelů je nutné provést migrace pomocí dvojice příkazů:

```
python3 manage.py makemigrations
python3 manage.py migrate
```

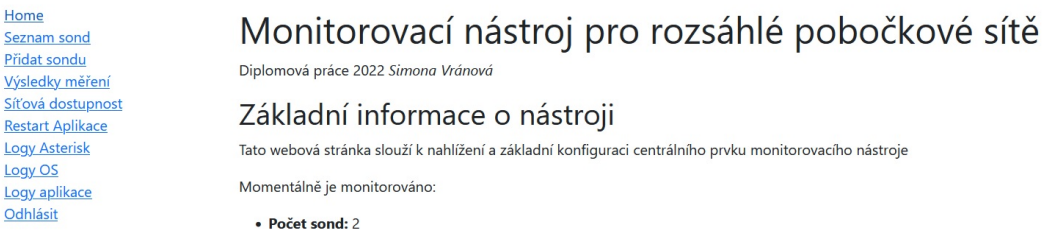
9.5.1 Nastavení postranního panelu a úvodní stránky

Postranní panel byl vytvořen jako HTML šablona, která je přidána na veškeré stránky obsahující navigační panel. V rámci postranního panelu jsou odkazy na jednotlivé části webových stránek. V rámci této šablony jsou také odkazy na knihovnu Bootstrap a na statické soubory CSS nastavující grafické rozložení stránek.

Po přihlášení je uživatel přesměrován na úvodní stránku, kde je v krátkosti popsáno webové rozhraní. Momentálně je na této stránce také vypsán aktuální počet monitorovaných (definovaných) sond. Toto je dynamicky generovaný parametr v rámci statické HTML šablony. Tato stránka by do budoucna šla rozšířit o další statistické údaje.

Úvodní stránka je tvořena třemi soubory: `urls.py, views.py a index.html`. V rámci souboru `urls.py` jsou definovány názvy odkazů pro každou stránku. Jsou to názvy pro psaný za lomítko webového odkazu, název definice v rámci souboru `views.py` a hypertextový odkaz využívaný v rámci HTML šablon.

```
urlpatterns = [
path('login', views.login_user, name='login'),
path('update_sonda/<Sonda_id>', views.update_sonda, name='update_sonda'),
...
]
```



Obrázek 10: Postranní panel a úvodní stránka

Soubor `views.py` zastřešuje samotné chování webové stránky a předání dynamického obsahu na HTML šablonu.

Soubor `index.html` definuje strukturu webové stránky. Grafická nastavení jsou součástí CSS souboru.

9.5.2 Zobrazení výsledků měření a síťové dostupnosti

Výsledky měření jsou z ukládány do textových souborů. V rámci funkce webových stránek jsou tyto hodnoty uloženy do databáze a tato databáze je následně předána pro zobrazení na webové stránce.

Aby bylo možné využít databáze, byly vytvořeny 2 speciální modely pouze pro zobrazování hodnot z textových souborů.

V rámci HTML šablon (`historie.html` a `sonda_view.html`) byly vytvořeny tabulky pro naměřené hodnoty. Syntaxe pro ně byla převzata z dokumentace pro Bootstrap.

Vyčítání dat z textových souborů a jejich ukládání je definováno v souboru `views.py`. V prvním kroku je původní databáze vymazána, aby nedocházelo k duplikaci dat. Následně je otevřen textový soubor a v rámci cyklu jsou po řádkách vyčteny jednotlivé parametry a ty jsou poté uloženy jako řádek databáze. Tato část kódu vypadá následovně:

```
def sonda_view(request):
    SondaView.objects.all().delete()
    with open('/root/vystup/seznamSond.txt') as file:
        for data in file:
            string_list = data.split()
            nazev = string_list[0]
            ...
            row = SondaView(
            nazev = nazev,
            ...
            )
            row.save()
    sonda_list = SondaView.objects.all()
    return render(request, 'sonda_view.html', {'sonda_list': sonda_list})
```

U historie měření bylo potřeba pro větší přehlednost ještě rozdělit výpis výsledků na stránky. Stránkování Django řeší pomocí knihovny `paginator`. Stránkování je nutné nastavit i na HTML šabloně. Je potřeba přidat odkazy na stránky. Samotné nastavení v souboru `views.py` pro zobrazení 40 objektů na stránku je následující:

Obrázek 11: Formulář pro přidání nové sondy

```
p=Paginator(History.objects.all().order_by('id'),40)
page=request.GET.get('page')
history_list=p.get_page(page)
```

9.5.3 Přidání nové sondy

Přidání nového prvku je v Django řešeno využitím formulářů. Formuláře jsou vytvořeny v souboru `forms.py`. Jednotlivé formuláře jsou poté importovány v rámci konfigurace ve `views.py`.

V rámci logiky vyhodnocování je zahrnuto zda byl formulář korektně vyplněn, nebo byla stránka poprvé zavolána. V prvním případě je do šablony předána informace o provedení a na webové stránce se objeví informační hláška o úspěšném vložení sondy. V druhém je na stránce zobrazen formulář pro vyplnění (obrázek 11).

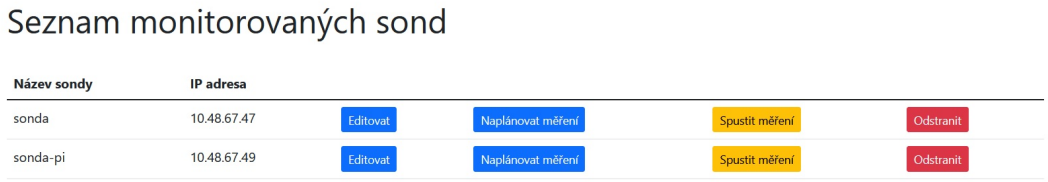
Po poslání správně vyplněného formuláře je vytvořen záznam v databázi pro sondy. Zároveň jsou vytvořeny konfigurační soubory Asterisku pro `extentions.conf` a `pjsip.conf`. Do těchto souborů jsou přidány odkazy na soubory pro konkrétní sondu svázané s ní podle jejího názvu: `nazev_sondy.conf` a `dial_nazev_sondy.conf`. Tyto soubory jsou také nově vytvořeny. Také je přidán záznam do textového souboru se seznamem sond.

Vyhodnocení poslání formuláře je provedeno následovně:

```
def add_sonda(request):
    submitted = False
    if request.method == "POST":
        form = SondaForm(request.POST)
        if form.is_valid():
         ...
            form.save()
            return HttpResponseRedirect('add_sonda?submitted=True')
    else:
        form = SondaForm
        if 'submitted' in request.GET:
            submitted = True
    return render(request, 'add_sonda.html', {'form':form,
      'submitted':submitted})
```

9.5.4 Zobrazení, úpravy a odstranění sond

Zobrazení sond je webová stránka vypisující seznam všech sond do tabulky. V rámci tabulky jsou u každé sondy také tlačítka pro úpravu a odstranění sond, spuštění měření a plánování měření (obrázek 12).



Obrázek 12: Webová stránka pro zobrazení sond

Každé tlačítko je funkční a obsahuje sérii úkonů a přesměrování na webovou stránku. V rámci odkazu se předává také ID jednotlivých sond do webového odkazu.

Parametry sondy jsou upravovány opět v rámci formuláře. Tento formulář se v tomto případě zobrazuje již předvyplněný stávajícími hodnotami pro sondu. Po poslání formuláře jsou změněny parametry v databázi. Zároveň je potřeba pozměnit konfigurační soubory pro Asterisk a textový soubor se seznamem sond. Toto je uskutečněno pomocí Python nástrojů pro správu souborů (například nahrazení textového řetězce, smazání souboru). Také je potřeba aktualizovat práci v `crontab`.

Odstranění sondy je uskutečněno pomocí načtení instance sondy a poté její vymazání. Je také nutné smazat konfigurační soubory a odstranit příslušné řádky v souborech odkazujících na tuto konkrétní sondu. Po smazání objektu sondy je uživatel přesměrován zpět na stránku se seznamem sond.

Syntaxe pro smazání objektu je následující:

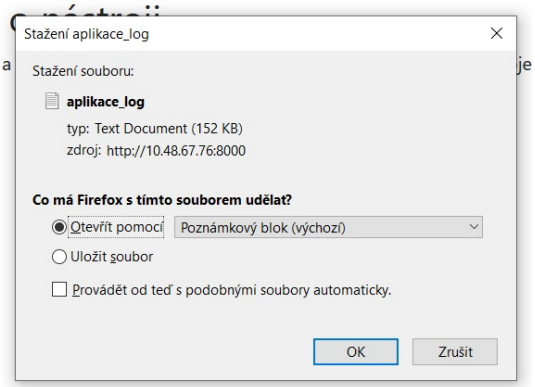
```
def delete_sonda(request, Sonda_id):
    sonda=Sonda.objects.get(pk=Sonda_id)
    sonda.delete()
    return redirect('sonda_seznam')
```

Veškeré změny musí být aplikovány. Systém Asterisk se sám automaticky neaktualizuje. K tomu slouží tlačítko pod tabulkou, které zajistí znovu načtení konfiguračních souborů. Toto znovu načtení nenaruší aktuálně probíhající hovory.

9.5.5 Plánování časů automatického měření

Měření jsou plánované pomocí nástroje `crontab`. V rámci webových stránek je možné nastavit frekvenci měření v řádu hodin a minut. Správa `cronu` pomocí skriptu v jazyce Python je následující:

```
with CronTab(user='root') as cron:
    cron.remove_all(comment=comment)
    job=cron.new(command,comment)
    job.hour.every(new_hour)
    job.minute.every(new_min)
```



Obrázek 13: Webové rozhraní pro stažení logů a průběh jejich získání

9.5.6 Spuštění měření

Měření je spuštěno pomocí stejného skriptu `autoMove.py` jako jsou vytvářeny automatické hovory. Jako vstupní argument je nastaven název sondy. Skript je vyvolán funkcí `os.system(command)`. Po spuštění měření je vypsána na webové stránce zpráva o spuštění měření. Pokud je sonda dostupná, objeví se po ukončení testu jeho výsledek v historii měření. Když sonda není dostupná, je vytvořen hovor na dispečera a odeslána textová zpráva.

9.5.7 Restart aplikace Asterisk

Restart aplikace je vyvolán pomocí tlačítka ze stránky restartu. Tento restart je vyvolán pomocí příkazu z CLI. Samotný příkaz Asterisku `core restart gracefully` znamená, že restart aplikace bude proveden až po ukončení všech probíhajících hovorů.

```
def restart_done(request):
    command="asterisk -rx 'core restart gracefully'"
    os.system(command)
    return render(request,'restart_done.html')
```

9.5.8 Stažení logů

V rámci aplikace jsou důležité 3 log soubory - Asterisk logy, Debian logy a logy samotné aplikace. Tyto logy jsou přístupné ke stažení přímo z webové stránky z navigačního panelu. Logy jsou stahovány jako textové přílohy (obrázek 13).

Předání přílohy textového souboru je uskutečněno v rámci Django následovně:

```
def asterisk_log(request):
    response=HttpResponse(content_type='text/plain')
    response['Content-Disposition']= 'attachment; name=asterisk.txt'
    ....
    return response
```

Obrázek 14: Webová stránka pro přihlášení

9.5.9 Role pro autorizaci pro správu

V rámci webových stránek se vyskytují 3 úrovně uživatelů: běžný uživatel, správce funkcionality a správce platformy. Jednotlivé role mají následující pravomoci:

- běžný uživatel
  - zobrazení seznamu sond
  - spuštění měření
  - zobrazení výsledků měření
- správce funkcionality
  - plánování měření
  - přidávání, aktualizace a ubíraní sond
- správce platformy
  - prohlížení logů
  - restart aplikace Asterisk
  - přidání nových uživatelů a změny jejich role

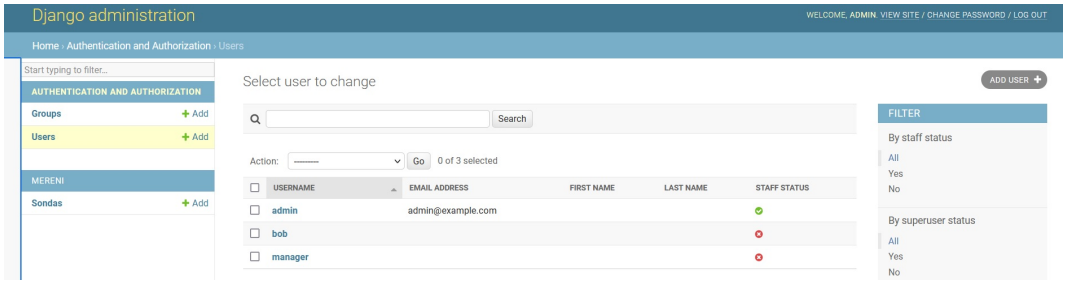
Role jsou plně hierarchické. Správce platformy má pravomoce správce funkcionality i běžného uživatele.

Pro samotný přístup k obsahu webových stránek je potřeba být přihlášen v jakékoli roli. V rámci webových stránek je plně funkční přihlašovací a odhlašovací systém. Přihlášení probíhá přes přihlašovací stránku (obrázek 14) dostupnou přes odkaz 'http://10.48.67.76:8000/admin/'. Uživatel se odhlašuje přes odkaz v rámci navigačního panelu, který uživatele odhlásí a přesměruje na stránku s přihlášením se zprávou o jeho úspěšném odhlášení.

Nový uživatelé jsou přidáváni a modifikováni z administrátorského rozhraní aplikace Django (obrázek 15). Pravomoci jsou uživatelům přidělovány pomocí skupin ('groups'). Přístup do tohoto rozhraní mají pouze uživatele s rolí správce platformy. V rámci administrátorské stránky je nutno nastavit uživatele jako 'superuser'.

Pro omezení přístupu k webovým stránkách bez přihlášení lze použít následují podmínku v rámci HTML šablony.

```
{% if user.is_authenticated %}
    ...
{% endif %}
```



Obrázek 15: Administrátorské rozhraní pro projekt Django

Některé stránky jsou poté ještě omezeny podle povolení. K daným stránkám mají poté přístup pouze uživatelé, kteří náleží do skupiny s náležitými právy. Toto je nastaveno v rámci HTML šablon pomocí následující podmínky:

```
{% if perms.admin.view_log %}
...
{% endif %}
```

10 Testovací provoz

V rámci laboratorního prostředí byla otestována funkčnost nástroje. Postupně byli uměle vyvolány všechny mimořádné stavy a ověřeny reakce nástroje.

10.1 Otestování funkčnosti nástroje

Nejprve bylo potřeba ověřit, že samotné hovory jsou korektně navázány. Tato kontrola proběhla síťovým záchytem pomocí nástroje `tcpdump`. Byly kontrolovány SIP zprávy (například pro ukončení hovoru) a správné navázání médií.

Pro některé mimořádné události budou v této části ukázány reakce systému v případě zápisu do logů a textových zpráv na správce.

První byla otestována kontrola síťové dostupnosti. Pro tuto kontrolu byla přidána nová sonda s nedostupnou IP adresou. Síťová nedostupnost je pokaždé vypsána do log souboru. Při prvotním zjištění je vyvolán hovor na dispečera a poslána textová zpráva ve formátu:

```
[08052022-16:35: 41] 'ERROR - nedostupná sonda': test.
```

Logy jsou ve formátu:

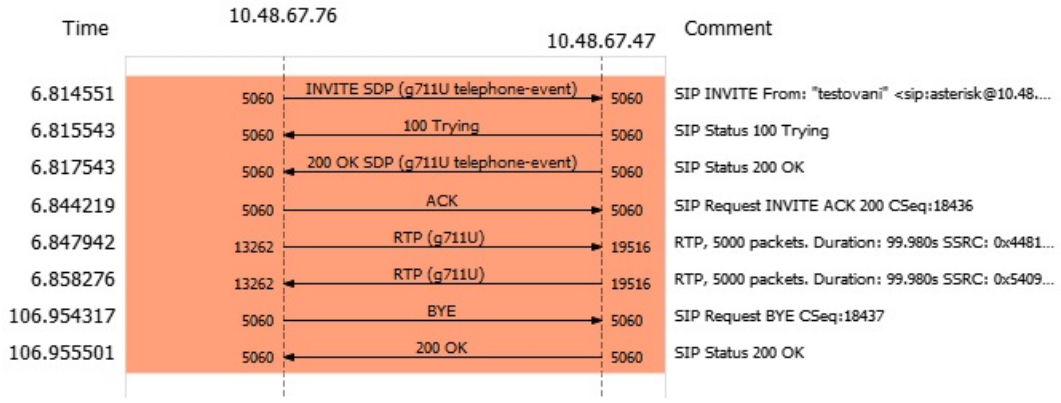
```
20:00:01 ERROR sonda síťově nedostupná: test.
```

Po obnovení dostupnosti sondy je opět zaslána textová zpráva dispečerovi.

Dále bylo otestováno přidání plánovaného měření. Měření bylo nastaveno na nedostupnou sondu. V tomto případě hovor nebylo možné vyvolat a dispečer byl na neúspěšný hovor upozorněn v nastaveném čase pro měření. Tento stav byl otestován i v případě síťově dostupné sondy, kde byl nuceně vypnut Asterisk. V obou případech se chová nástroj stejně.

Po obnovení dostupnosti byli otestovány reakce na různé hodnoty délek hovoru a kvality hovoru. Systém opět reagoval podle stanovených předpokladů. Délka hovoru byla upravena v rámci 'dialplanu' změnou testovací nahrávky na jinou s rozdílnou délkou záznamu. Celková kvalita hovoru byla nastavena na nepřijatelnou hodnotu v rámci vyhodnocovacího skriptu. Obdobným způsobem byla otestována jednosměrná neslyšitelnost.

Veškeré mimořádné stavy byly navozeny a funkčnost vyhodnocovacích skriptů byla ověřena.



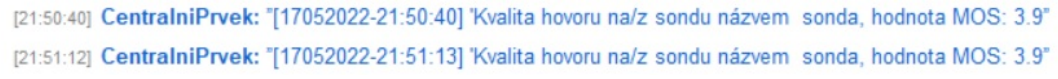
Obrázek 16: Záchyt hovoru mezi centrálním prvkem a sondou

(a) Textová zpráva

```
13:20:01 ERROR - neprovedený hovor na sondu sondas koncovým stavem: NO ANSWER
```

(b) Výpis do logů

Obrázek 17: Výstupy při SIP nedostupnosti sondy



(a) Textová zpráva

```
21:50:40 ERROR - nedostatečná kvalita hovoru, hovor na  sonda, hodnota MOS: 3.9
21:51:13 ERROR - nedostatečná kvalita hovoru, hovor na  sonda, hodnota MOS: 3.9
```

(b) Výpis do logů

Obrázek 18: Výstupy při špatné kvalitě spojení

10.2 Otestování funkčnosti webových stránek

Některé funkcionality webové stránky a jejich funkčnost byla ukázána již v předchozí kapitole, proto budou tyto funkce zde přeskočeny. Jedná se o přihlášení a odhlášení, zobrazení historie měření, zobrazení síťové dostupnosti a stahování logů.

10.2.1 Přidání, aktualizace a odstranění sond

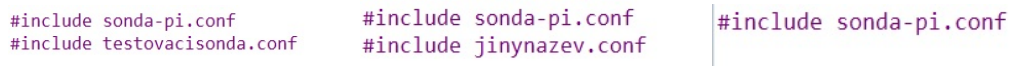
Přidání nové sondy je umožněno přes webovou stránku obsahující formulář pro zadaní názvu, IP adresy a hesla. Po vyplnění jsou do konfiguračních souborů přidány náležité řádky a jsou vytvořeny nové konfigurační soubory. Na obrázku 19 je znázorněn postupný vývoj jednoho řádku v rámci souboru `pjsip.conf` při přidání, aktualizaci a následovném smazání sondy. Soubor `extension.conf` je modifikován obdobně.

10.2.2 Vyvolání testovacího hovoru

Při spuštění testovacího hovoru dochází k vytvoření hovoru. Toto bylo ověřeno výpisem v rámci CLI Asterisku, kde jsou zaznamenávány jednotlivé úkony systému (obrázek 20).

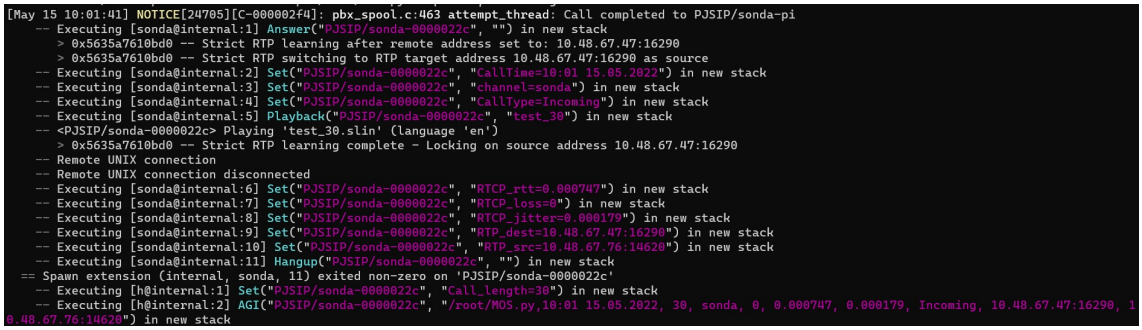
10.2.3 Restart aplikace Asterisk

Restart aplikace byl ověřen opět v rámci CLI Asterisku. Pokud je příkazová řádka otevřena, je po spuštění restartu ukončena. Zároveň je možné ověřit dobu běhu aplikace pomocí příkazu `core show uptime`, tato doba by měla být vynulovaná.



(a) Přidání (b) Aktualizace (c) Odstranění

Obrázek 19: Aktualizace obsahu konfiguračního souboru `pjsip.conf` při přidání, aktualizaci a odstranění sondy



Obrázek 20: Výpis z CLI Asterisku po vyvolání hovoru

11 Možné rozšíření

Tato kapitola shrnuje další možná rozšíření systému a jejich možnou implementaci.

11.1 Připojení k technologii TDM

Ačkoliv je klasická technologie přepínaných okruhů nahrazována technologií VoIP, je stále přítomné v rámci pobočkových sítí. Některé tyto technologie mohou být i součástí interních nařízení a není možné je jednoduše nahradit. Z tohoto důvodu by bylo vhodné do budoucna začlenit i měření kvality klasické telefonie. Tato práce upřednostnila tvorbu konfiguračního webového rozhraní.

11.1.1 Popis technologie TDM

Technologie digitálních přepínaných okruhů klasické telefonie je od VoIP technologie odlišná způsobem navázání spojení a použitých protokolů pro signalizaci. Na rozdíl od VoIP nejsou hovory posílány paketově, ale po vystavěných okruzích, tedy po fyzických obvodech sloužících pouze pro telefonní službu. Oproti VoIP technologii zaručuje TDM vysokou kvalitu přenosu. Zároveň jsou kladeny jiné požadavky, například hodnota maximálního zpoždění byla stanovena na 450 μs [2].

11.1.2 Způsob připojení k systému Asterisk

V tuto chvíli by se muselo opustit od plně virtualizovaného centrálního prvku. Pro komunikaci s klasickou telefonní síti je nutné využít E1 kartu, kterou je nutné připojit k fyzickému serveru.

K TDM ústředně by bylo možné se připojit využitím protokolu QSig. Tento protokol se konfiguruje v rámci kanálu DAHDI. Konfigurace by mohla vypadat následovně:

```
[channels]
language=cs
switchtype=qsig
signalling=xxx
....
```

Obrázek 21: Zapojení prvků pro měření

Další parametry by byly nastaveny podle druhu ústředny.

11.1.3 Měření parametrů

Na rozdíl od VoIP technologie není možné využít protokol RTCP, kde jsou veškeré parametry již změřeny. Samotný signál by bylo nejspíš vhodné analyzovat pomocí frekvenční analýzy. Jelikož by byl posílána testovací nahrávka bylo by možné využít Fourierovu transformaci na referenční a naměření signál. Na tomto principu funguje například algoritmus PESQ vytvořený ITU-T v rámci doporučení P.862.

Pro zaznamenání přijímaného hovoru je možné využít aplikaci 'dialplanu' 'Monitor()'. Nahrávání příchozí a odchozí zvukové stopy je odlišen pomocí parametru.

11.2 Ověření správnosti měřených dat systémem Asterisk

Pro ověření naměřených hodnot a také ověření výpočtu kvalitu by šel využít nástroj `Netem`, který je součástí operačního systému Debian. Tento nástroj dokáže do sítě zavést konkrétní zpoždění a ztrátovost paketů. Je také možnost nastavit změnu pořadí paketů.

V laboratorním prostředí by se využilo virtuální zařízení, které by simulovalo trasu. Konkrétní zapojení je znázorněno na obrázku 21. Na centrálním prvku a sondách by byla nastavena statická routa na virtuální prvek, na kterém je simulována trasa.

Nastavení nástroje `Netem` je umožněno příkazem `tc qdisc`. Nastavení pevného zpoždění 100 ms na rozhraní 'eth0' vypadá následovně [34]:

```
# tc qdisc add dev eth0 root netem delay 100ms
```

Postupně by byly nastavovány hodnoty pro zpoždění, ztrátovost paketů a jiter a byl by sledován jejich vliv na měřené hodnoty.

11.3 Rozšíření o e-mailovou službu

Další možností pro rozšíření je vytvoření e-mailových notifikací na správce technologie. Tyto e-mailové zprávy by mohly být posílány v případě méně vážných mimořádných událostí. Další využití by mohlo být i více statistického charakteru. Například by se na konci měsíce vytvořilo vyhodnocení četnosti chyb, případně by byly poslány log soubory.

Pro posílání e-mailových zpráv je nutné správně nastavit SMTP server. Tento server je možné nastavit pomocí platformy `Sendmail`.

12 Závěr

Cílem této diplomové práce bylo vytvořit monitorovací a testovací nástroj pro rozsáhlé pobočkové sítě. Pomocí systému Asterisk byl vytvořen nástroj pro automatické monitorování stavu a kvality existující pobočkové sítě. Hlavním záměrem byla detekce předem definovaných mimořádných stavů a vytvoření adekvátní reakce pro lepší správu celé sítě.

Tento testovací nástroj je založen na komunikaci dvou typů prvků v síti: centrálního prvku a sond. Komunikace mezi sondami a centrálním prvkem má emulovat komunikaci v rámci pobočkové sítě. Cílem tohoto nástroje je detekovat mimořádné stavy za využití externích zařízení, bez nutnosti analyzovat hovory reálného provozu.

V první části práce byly teoreticky popsány veškeré technologie, které byly při tvorbě nástroje využity. Velký důraz byl kladen na popis konfigurace systému Asterisk.

Další část se zabývala rozborem zadání a definování základních předpokladů. Jedním z předpokladů bylo využití pouze VoIP technologie. Dále bylo specifikováno omezení vzhledem k využití omezeného testovacího prostředí. Například využití přímé registrace sond s centrálním prvkem.

V posledních částech byla uvedena konfigurace jednotlivých částí nástroje: Asterisk a web server vytvořený nástrojem Django. Nástroj byl otestován a všechny definované mimořádné stavy jsou detekovány a je na ně korektně reagováno.

Finální nástroj detekuje mimořádné stavy: síťová nedostupnost, nemožnost navázat telefonní hovor, obousměrná/jednosměrná neslyšitelnost, nízká kvalita spojení. Nemožnost navázat telefonní spojení oběma směry je také kontrolována, nicméně tento mimořádný stav není automaticky detekován, je možné hovory zkontrolovat z webového rozhraní. Kvalitativní parametry byly získávány ze zpráv posílaných v rámci protokolu RTCP. Vyhodnocování probíhalo v rámci skriptů psaných v jazyce Python. Jako signalizační protokol byl využit protokol SIP. Konfigurace v rámci Asterisku byla vytvořena v rámci kanálu pjsip a lokálního kanálu.

Centrální prvek je možné plně virtualizovat. Nároky na centrální prvek byly stanoveny na procesor 3GHz, RAM 2-4 GB a paměť 20 GB. Sondy mohou být například zařízení Raspberry Pi. V této práci bylo využito zařízení Raspberry Pi3 model B+. Celková cena systému by se odvíjela podle počtu sond.

Pro lepší prezentaci výsledků měření bylo vytvořeno webové rozhraní. Původně byl záměr vytvořit web pouze na prezentaci výsledků. Nakonec bylo rozhodnuto, že pro lepší obsluhu bude webové rozhraní obsahovat i možnosti pro konfiguraci. Veškerá konfigurace systému je nyní možná z webových stránek. Jedná se o přidávání, aktualizace a odstranění sond, plánování měření, spuštění měření, stahování logů, restart aplikace Asterisk. Webová aplikace byla vytvořena pomocí nástroje Django. Pro grafickou úpravu byl využit nástroj Bootstrap.

Tento nástroj by do budoucna bylo možné rozšířit o připojení k TDM síti, případně o přidání e-mailové notifikace.

Zdroje

[1] MEGGELEN, Jim Van, Russell BRYANT a Leif MADSEN. *Asterisk: The Definitive Guide*. 5th Edition. Beijing: O'Reilly Media, ©2019. ISBN 978-1-492-03160-4.

[2] COOMBS, Clyde F., ed. *Communications network test and measurement handbook*. New York: Osborne-McGraw-Hill, ©1998. xvii, 813 s. ISBN 0-07-012617-8.

[3] SUN, Lingfen et al. *Guide to Voice and Video over IP: For Fixed and Mobile Networks* [online].1st 2013. London: Springer London, 2013. ISBN 978-1-4471-4905-7.

[4] DINARDI, Gaetano. *What Is VoIP & How Does It Work?* [online blog]. ©2020 [cit. 2022-04-24]. Dostupné z: https://www.nextiva.com/blog/what-is-voip.html

[5] ROSENBERG, J. et. al. *SIP: Session Initiation Protocol* [online]. 2002 [cit. 2022-04-24]. Request For Comments (RFC) 3261. Dostupný z: https://datatracker.ietf.org/doc/html/rfc3261

[6] RODRIGUES COSTA, Lucas et al. *Asterisk PBX capacity evaluation* [online]. IEEE, 2015. 519-524 s. ISBN 978-1-4673-7684-6/15.

[7] *Asterisk Wiki* [online]. Asterisk Project. [cit. 2022-04-25]. Dostupné z: https://wiki.asterisk.org/

[8] *PJSIP Configuration Sections and Relationships* [online]. Asterisk Project. 2018 [cit. 2022-04-25]. Dostupné z: https://wiki.asterisk.org/wiki/display/AST/PJSIP+Configuration+Sections+and+Relationships

[9] MEGGELEN, Jim Van, Jared SMITH a Leif MADSEN. *Asterisk: the future of telephony* [online]. 2nd Edition. Beijing: O'Reilly, ©2007. ISBN 978-0-5965-1048-0.

[10] TIPANTUNA, Christopher et al. *Performance Analysis of a Raspberry Pi Based IP Telephony Platform* [online]. 2015, vol. 36 [cit. 2022-04-24]. Dostupné z: https://www.researchgate.net/publication/328687559_Performance_Analysis_of_a_Raspberry_Pi_Based_IP_Telephony_Platform

[11] *What Is Network Monitoring?* [online]. Cisco Systems, Inc.. [cit. 2022-04-25]. Dostupné z: https://www.cisco.com/c/en/us/solutions/automation/what-is-network-monitoring.html

[12] PECHA, Petr. *What are network monitoring tools?* [online]. 2021 [cit. 2022-04-25]. Dostupné z: https://www.flowmon.com/en/blog/what-are-network-monitoring-tools

[13] WILSON, Mark. *Best Network Monitoring Tools & Software* [online]. 2022 [cit. 2022-04-26]. Dostupné z: https://www.pcwdld.com/best-network-monitoring-tools-and-software#wbounce-modal

[14] KEARY, Tim. *10 Best VoIP Monitoring Tools & Software for 2022* [online]. 2021 [cit. 2022-04-26]. Dostupné z: https://www.comparitech.com/net-admin/best-voip-monitoring-tools/

[15] *Virtual PBX providers* [online]. VOIP-Info.org LLC. [cit. 2022-04-28]. Dostupné z: https://www.voip-info.org/virtual-pbx-providers/

[16] ITU-T Recommendation G.1020. *Performance parameter definitions for quality of speech and other voiceband applications utilizing IP networks*. Geneva: UTU-T. 2007

[17] ANTONIOUS, Stelios. *VoIP Signaling Protocols* [online]. 2010 [cit. 2022-04-28]. Dostupné z: https://www.pluralsight.com/blog/it-ops/voip-signaling-protocols

[18] ROSENBERG, et. al. *SIP: Session Initiation Protocol*. Request for Comments 3261. Internet Engineering Task Force, červen 2002

[19] *What is a web server?* [online]. MDN contributors. 2022 [cit. 2022-04-28]. Dostupné z: https://developer.mozilla.org/en-US/docs/Learn/Common_questions/What_is_a_web_server

[20] *Single-Board Computer (SBC)* .Janalta Interactive.2017 [cit. 2022-05-04]Dostupné z: https://www.techopedia.com/definition/9266/single-board-computer-sbc

[21] VODRÁŽKA, Jiří, Ivan PRAVDA a České vysoké učení technické v Praze. Elektrotechnická fakulta. *Principy telekomunikačních systémů*. Praha : Česká technika - nakladatelství ČVUT, 2006.

[22] ITU-T Recommendation Y.1541. *Network performance objectives for IP-based services*. Geneva: UTU-T. 2011

[23] TESAŘÍK, Lukáš a Martin WARZESZKA. *ASTERISK NA RASPBERRY PI* [online]. 2016 [cit. 2022-04-30]. Dostupné z: http://skola.tlukas.eu/voip/#software

[24] *Welcome To SwitchPi*. SwitchPi. 2022 [cit. 2022-05-10] Dostupné z: https://switchpi.com/

[25] *Banana PI BPi-M2*. Conrad Electronic Česká republika, s.r.o.. 2021 [cit. 2022-05-04]. Dostupné z: https://www.conrad.cz/p/banana-pi-bpi-m2-banana-pi-bpi-m2-1-gb-4-x-10-ghz-2142748

[26] JOHNSTON, Alan B., Daniel SCHUTZER a Inc BOOKS24X7. *SIP: Understanding the Session Initiation Protocol, Third Edition* [online].3rd. Norwood: Artech House, 2009. ISBN 9781607839958;1607839954;.

[27] Bates, Regis J. Jr (Bud). *Securing VoIP: Keeping Your VoIP Network Safe* [online].Rockland, MA: Elsevier Science & Technology Books, 2014;2015;. ISBN 9780124170391;0124170390;0124171222;9780124171220;.

[28] *What is SIPS and SRTP?* [online]. Dostupné z: https://askozia.com/voip/what-is-sips-and-srtp/

[29] *Call Quality Defining Using R-Factor And MOS* [online]. Dostupné z: https://www.voicehost.co.uk/help/call-quality-r-factor-and-mos

[30] VENKAT. *Mean Opinion Score (MOS) Calculation of Asterisk VoIP calls* [online]. 2017 [cit. 2022-05-01]. Dostupné z: https://sillycodes.com/mean-opinion-score-or-mos-calculation-asterisk-voip-calls/

[31] *VoIP Performance Over WiFi* [online]. Alethea Communications Technologies Pvt Ltd. [cit. 2022-05-01]. Dostupné z: https://alethea.in/voip-performance-wifi/

[32] MDN contributors. *Django introduction* [online]. 2022 [cit. 2022-05-01]. Dostupné z: https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/Introduction

[33] *What's Orange Pi 4?*. Xunlong Software CO. 2016 [cit. 2022-05-04].Dostupné z: http://www.orangepi.org/Orange%20Pi%204/

[34] *Netem* [online]. CC Attribution-Noncommercial-Share Alike 4.0 International 2021 [cit. 2022-05-16]. Dostupné z: https://wiki.linuxfoundation.org/networking/netem

[35] DJANGO SOFTWARE FOUNDATION. *Django project* [software]. 2021 [cit. 2022-05-16].Dostupné z: https://www.djangoproject.com/download/

[36] SANGOMA TECHNOLOGIES. *Asterisk* [software]. 2021 [cit. 2022-05-16]. Dostupné z: https://www.asterisk.org/downloads/

[37] *Bootstrap* [software]. [cit. 2022-05-16]. Dostupné z: https://getbootstrap.com/docs/5.2/getting-started/download/

[38] ELDER, John. *Django-Club-Youtube-Playlist* [online]. 2022 [cit. 2022-05-16]. Dostupné z: https://github.com/flatplanet/Django-Club-Youtube-Playlist

[39] *About PJSIP* [online]. [cit. 2022-05-17]. Dostupné z: https://www.pjsip.org/about.htm

[40] *Jazyk HTML*. ReJ. 2016 [cit. 2022-05-19]. Dostupné z: http://www.owebu.org/cze/html/html_tagy.php

[41] ŠTRÁFELDA, Jan. *HTTP*. [cit. 2022-05-19]. Dostupné z: https://www.strafelda.cz/http

SEZNAM ZKRATEK

OS	Operating system
GUI	Graphical user interface
VoIP	Voice over IP
MOS	Mean opinion score
ITU-T	International Telecommunication Union - Telecommunication Standardization Sector
SIP	Session Initiation Protocol
PSTN	Public Switched Telephone Network
IP	Internet Protocol
CUCM	Cisco Unified Communications Manager
WLAN	Wireless LAN
LAN	Local area network
UDP	User Datagram Protocol
TCP	Transmission Control Protocol
RTP	Real-time Transport Protocol
RTCP	RTP Control Protocol
SDP	Session Description Protocol
IAX2	Inter-Asterisk eXchange version 2
SCCP	Skinny Client Control Protocol
MGCP	Media Gateway Control Protocol
IMS	IP Multimedia Subsystem
HTTP	Hypertext Transfer Protocol
SMTP	Simple Mail Transfer Protocol
RFC	Request for Comments
QoS	Quality of Service
QoE	Quality of Experience
cRTP	compressed RTP
SIPS	SIP Secure
SRTP	Secure RTP
TLS	Transport Layer Security
AES	Advanced Encryption Standard
SDES	Session Description Protocol Security Descriptions
VPN	Virtual private network
VLAN	Virtual local area network
IPSec	Internet Protocol Security

ICMP	Internet Control Message Protocol
PING	Packet InterNet Groper
IPTD	IP packet transfer delay
IPDV	IP packet delay variation
IPLR	IP packet Loss Ratio
SNR	Signal-to-noise ratio
RTT	Round-trip Time
SNMP	Simple Network Management Protocol
CPU	Central processing unit
DAHDI	Digium Asterisk Hardware Device Interface
STUN	Simple Traversal of User Datagram Protocol Through NAT
TURN	Traversal Using Relays around NAT
ICE	Interactive Connectivity Establishment
AoR	Address of Records
NAT	Network Address Translators
PAT	ort Address Translation
MD5	Message-digest algorithm
RSA	Rivest–Shamir–Adleman
FXS	Foreign eXchange Subscriber
FXO	Foreign eXchange Office
PRI	Primary Rate Interface
BRI	Basic Rate Interface
MFC/R2	Multi-Frequency Compelled R2
SS7	Signalling System No. 7
PBX	Private Branch Exchange
CSS	Cascading Style Sheets
HTML	HyperText Markup Language
JSON	JavaScript Object Notation
XML	Extensible Markup Language
PHP	Hypertext Preprocessor
RAM	Random-access memory
Wi-Fi	Wireless Fidelity
USB	Universal serial bus
HDMI	High-Definition Multimedia Interface
PoE	Power Over Ethernet
DHCP	Dynamic Host Configuration Protocol
LTE	Long Term Evolution

NTP	Network Time Protocol
CLI	command-line interface
CDR	Call detail record
AMI	Asterisk Manager Interface
TDM	Time-division multiplexing
PESQ	Perceptual evaluation of speech quality

Seznam příloh

Veškeré konfigurační soubory a skripty jsou přiloženy jako přílohy, pro papírovou verzi jsou soubory obsaženy na datovém médiu přiloženém u práce.

Konfigurační soubory Asterisk

- centrální prvek
  - pjsip.conf
  - sonda.conf
  - extensions.conf
  - dialsonda.conf
- sonda
  - pjsip.conf
  - extensions.conf

Skripty

- centrální prvek
  - automove.py
  - testcall.py
  - sondaPing.py
  - MOS.py
- sonda
  - automove.py

Konfigurační soubory webového rozhraní

Zvukové testovací soubory:

- test.wav
- test_30.wav

Výstupní soubory:

- seznamSond.txt
- vystupMOS.txt
- aplikace.log

Snímky obrazovky webových stránek